Problèmes d'ordonnancement et de moyens de transport des systèmes de production : prise en compte de la qualité de service

Matthieu Gondran

Université Clermont Auvergne

École doctorale des
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse

présentée par

Matthieu GONDRAN

pour obtenir le grade de

Docteur d'Université

Spécialité : Informatique

Problèmes d'ordonnancement et de moyens de transport des systèmes de production : prise en compte de la qualité de service

Soutenue publiquement le 04 octobre 2019 devant le jury composé de :

Rapporteurs :	
Caroline PRODHON	Maître de Conférences, HDR, Université de Technologie de Troyes
Marc SEVAUX	Professeur des Universités, Université de Bretagne Sud
Examinateurs :	
Éric BOURREAU	Maître de Conférences, Université de Montpellier
Thierry GARAIX	Maître de Conférences, École des Mines de Saint-Étienne
Marie-José HUGUET	Professeur des Universités, l'INSA de Toulouse
Directeurs de thèse :	
Philippe LACOMME	Maître de Conférences, HDR, Université Clermont Auvergne
Alain QUILLIOT	Professeur des Universités, Université Clermont Auvergne
Nikolay TCHERNEV	Professeur des Universités, Université Clermont Auvergne

À Rafaële,
À Gabriel.

Remerciements

Je tiens à remercier chaleureusement mes trois directeurs de thèse : Philippe Lacomme, Alain Quilliot et Nikolay Tchernev, d'abord pour leur recherche de financement qui a permis de réaliser ce projet, mais surtout pour la confiance qu'ils m'ont accordée, pour leur disponibilité, leur patience et leur dévouement. Leurs immenses connaissances ont été des atouts précieux tout au long de ces trois années de recherche.

Merci à chacun des membres du jury pour l'intérêt porté à ces travaux, pour leur relecture attentive du manuscrit et leurs suggestions. Je remercie tout particulièrement Caroline Prodhon, Maître de Conférence HDR à l'Université Technologique de Troyes, et Marc Sevaux, Professeur à l'Université de Bretagne Sud, d'avoir accepté d'être rapporteurs.

Je veux également remercier Éric Bourreau, Thierry Garaix et Marie-José Huguet pour les collaborations que nous avons eues ensemble, pour l'accueil qu'ils m'ont réservé dans leurs établissements respectifs, et pour tout ce qu'ils m'ont transmis. Chacun a partagé avec moi une part de ses connaissances, me permettant de découvrir des pans entiers de la Recherche Opérationnelle.

Je remercie toutes les personnes du LIMOS avec qui j'ai eu la chance de discuter, et qui ont ainsi apporté leur pierre à l'édifice de ces travaux de recherche.

Merci à tous les responsables et directeurs de l'ISIMA et de Polytech qui m'ont offert l'opportunité d'enseigner dans leurs écoles durant ces trois années. Merci de m'avoir donné la possibilité de transmettre une partie de mes connaissances, et – bien plus ! – de communiquer une passion.

Merci infiniment à tous mes amis du LIMOS, qui ont fait de ces trois années une joie. Ils sont nombreux, mais méritent tous d'être cités : Benjamin Bergougnoux et ses égarements politiques ; David Brevet et son sens du service ; Rafael Colares et son soutien de chaque jour, dans le bureau que nous avons partagé durant trois ans ; Benjamin Dalmas et sa force Pépouze® (sans oublier son Saint-Môret® étalé au carré sur le pain de mie !) ; Théo Ducros et sa collaboration dans les cours ; José-Luis Figueroa et ses discussions passionnantes sur les cultures du monde ; Damien Lamy et ses questions existentielles ; Arnaud Laurent et son scepticisme ; Henri Perret du Cray et ses « soirées de folies » ; Rui Shibasaki et sa passion pour les vieux chanteurs français ; Benjamin Vincent, « l'être de lumière » ; et enfin Marina Vinot et ses excellents conseils.

Je remercie toute ma famille – ma belle-famille comprise – qui, pendant trois ans, se sont révélés être mes plus grands « supporters » ! Ils ont été un solide appui, tant sur le plan moral que pratique (notamment pour leur minutieux travail de relecture).

Enfin, je dis merci à Gabriel, petit ange et rayon de soleil dans nos vies.

Et merci à Rafaële, qui en plus d'être la mère géniale de notre fils, est une femme merveilleuse. Ce qu'elle m'a apporté gratuitement est immense et inexprimable. Sa présence à mes côtés mériterait bien plus que quelques lignes de remerciements. J'ai voué beaucoup d'amour à ces travaux de recherche, et je lui en voue mille fois plus. Merci.

Ce travail de recherche est cofinancé par l'Union européenne dans le cadre du Fonds Européen de Développement Régional (FEDER).

Résumé

Ce manuscrit aborde des problèmes d'ordonnancement et de transport avec une modélisation explicite du transport. De tels problèmes se modélisent communément sous forme de graphes qui sont évalués afin d'obtenir les dates de début des opérations.

Les évaluations classiques des graphes sont effectuées au moyen d'algorithmes de plus long chemin permettant d'obtenir une solution semi-active, où toutes les dates des opérations sont au plus tôt. Néanmoins, ces évaluations permettent généralement de ne prendre en compte que des critères de temps ou de distance à minimiser. Les travaux présentés dans ce manuscrit proposent de tenir compte de critères de qualité de service dans la fonction objectif. Cette prise en considération nécessite de nouvelles fonctions d'évaluation du graphe afin d'obtenir des solutions non nécessairement semi-actives permettant de maximiser la qualité de service. En effet, une solution semi-active propose rarement une qualité de service optimale. Les critères de qualité de service adoptés portent sur les ordonnancements et sur le transport.

Trois problèmes intégrés sont successivement traités. Le premier problème est un problème de Job-shop avec transport et qualité de service, appelé Job-shop Scheduling Problem with Routing (JSPR). Des pièces, définies par une succession d'opérations, sont à fabriquer sur différentes machines, et entre deux opérations, la pièce doit être transportée de machine en machine. Le critère de qualité de service dans ce problème est dépendant des délais entre, d'une part les différentes opérations sur les machines, et d'autre part entre les différentes opérations de transport. Les gammes opératoires et les opérations de transport sont dépendantes les unes des autres.

Le second problème est un problème de Workforce Scheduling and Routing Problem (WSRP), assimilable à un problème de planification de visites à domicile par un ensemble d'employés, et où le transport est pris en compte. Pour ce problème, le critère de qualité de service dépend des dates de début des visites. Les tournées sont indépendantes les unes des autres.

Le troisième problème est le Generalised Workforce Scheduling and Routing Problem (GWSRP), qui prend en compte des contraintes de coordination entre les employés. Les tournées de ces derniers sont dépendantes les unes des autres. Elles nécessitent d'être toutes considérées simultanément pour évaluer les dates des visites respectant les contraintes de coordination et maximisant la qualité de service.

Pour chaque problème, une nouvelle fonction d'évaluation est proposée. Pour le JSPR, cette fonction est basée sur l'algorithme de (Cordeau and Laporte, 2003) qui est initialement prévu pour le Dial-A-Ride Problem, ainsi que sur l'insertion de time-lags dans le graphe disjonctif du JSPR. Cette évaluation est incluse dans une métaheuristique. Pour le WSRP, la fonction d'évaluation est basée sur un algorithme de calcul du plus court chemin avec un algorithme de type programmation dynamique à labels. Elle est généralisée pour être utilisée dans une génération de colonnes. Et enfin, pour le GWSRP, l'évaluation est effectuée par un modèle PPC qui combiné à une génération de colonnes définissent tous deux un schéma d'optimisation global.

Abstract

This manuscript addresses scheduling and transport problems where the transport is explicitly taken into account. Such problems are commonly modelled by graphs that are evaluated to obtain the starting times of operations.

Classic graph evaluations are performed using longer path algorithms to obtain a semi-active solution, where all operations are left shifted. Nevertheless, these evaluations generally allow only time or distance criteria to be taken into account. The work presented in this thesis propose to take the quality of service criteria into account in the objective function. These considerations require new graph evaluation functions in order to obtain non-semi-active solutions that maximise the quality of service. Indeed, a semi-active solution rarely offers maximum quality of service.

Three integrated problems are successively addressed. The first problem is a Job-shop Scheduling Problem with transport and quality of service, referred to as Job-shop Scheduling Problem with Routing (JSPR). Jobs, defined by a succession of operations, are to be performed on different machines, and between two operations, the job must be transported from a machine to another machine. The quality of service criterion in this problem depends on the delay between, on the one hand, the different operations belonging to the same job, and on the other hand, between the different transport operations. Machine-operations and transport-operations are dependent.

The second problem is a Workforce Scheduling and Routing Problem (WSRP), which is similar to a problem of planning home services by a set of employees, and where transport is taken into account. For this problem, the quality of service criterion depends on the starting times of the visits. The trips of employees are independent.

The third problem is the Generalised Workforce Scheduling and Routing Problem (GWSRP), which takes coordination constraints between employees into account. The trips are dependent on each other. The evaluation function of the starting times must consider simultaneously all trips in order to respect all coordination constraints and to maximise the service quality.

For each problem, a new evaluation function is proposed. For the JSPR, this function is based on the algorithm of (Cordeau and Laporte, 2003) which is introduced first for the Dial-A-Ride Problem. The evaluation function, for the JSPR, is based on the insertion of time-lags in the disjunctive graph. This evaluation is included in a metaheuristic. For the WSRP, the evaluation function is based on the dynamic labelling algorithm used for an Elementary Shortest Path Problem With Resource Constraints. This function is generalised in order to be included in a column generation scheme. Finally, for the GWSRP, the evaluation is performed by a PPC model combined with a generation of columns and both define an overall optimisation scheme.

Sommaire

Introduction générale

L'objectif des systèmes de production et de transport intégré est d'ordonnancer des opérations de production de biens, de planifier les opérations de transport de ces produits, et d'affecter des véhicules aux opérations de transport. Le transport peut être associé à un problème de tournées de véhicules, tandis que la production peut être apparentée à un problème d'ordonnancement. L'optimisation de tels systèmes est un enjeu majeur pour de nombreux secteurs, allant de la chaîne logistique en milieu industriel aux services à domicile.

Les problèmes de production et de transport intégré couvrent un large spectre de problèmes théoriques à visée « industrielle » ou de « service ». Dans le premier cas, les produits correspondent à des biens matériels qui nécessitent d'être transportés entre des machines. Les opérations de transport sont effectuées par des AGVs (Automated Guided Vehicles) ou des robots. Le Job-shop Scheduling Problem with Routing est un problème classique des problèmes de production et de transport à visée industrielle : des pièces sont à fabriquer sur différentes machines et des AGVs doivent les transporter entre les machines. Dans le second cas, les produits sont des services requis par des clients, et qui doivent être réalisés par des employés. Les opérations de transport correspondent aux déplacements des employés pour aller d'un client à un autre. Le Home Health Care Problem est une application typique des problèmes de production et de transport à visée de service : un ensemble d'aides-soignants doit visiter et procurer des soins à des patients à domicile.

Les critères de performances « classiques » des systèmes de production et de transport sont : de manière non exhaustive, la minimisation des coûts de production, la minimisation de la distance parcourue par les véhicules (ou les AGVs), la minimisation du makespan (date de fin de la dernière opération), la maximisation du nombre de pièces (ou de visites) réalisées. Depuis quelques années, d'autres objectifs apparaissent dans la littérature et sont souvent articulés autour de critères de qualité de service, notion très large et pouvant englober de nombreux éléments. Elle peut être définie aussi bien pour la production que pour le transport. La qualité de service pour la production (donc concernant les clients ou les pièces) intègre régulièrement : la préférence du client pour l'employé effectuant le service, l'horaire où l'employé arrive pour réaliser le service, le temps d'attente de la pièce sur la machine avant son usinage, le temps de transfert des pièces entre les machines, etc. La qualité de service pour la partie transport (portant donc sur les employés ou les machines) admet généralement : les temps d'attentes des employés, la durée totale de la tournée, les dates de début et de fin de tournée, les zones géographiques à couvrir, les pauses, l'équilibrage des lignes de production, etc.

En plus des objectifs précédemment présentés, les systèmes de production et de transport sont soumis à un ensemble de contraintes à respecter. Parmi les contraintes au centre des articles de la recherche actuelle et celles prises en compte dans ce manuscrit, il y a : les contraintes disjonctives où une machine peut usiner une seule pièce à la fois, le respect des fenêtres de temps des visites, des contraintes de coordination de visites pour les services à domicile.

Les problèmes de production et de transport sont des problèmes combinatoires dont il est généralement très difficile de trouver une solution et/ou la solution optimale. De nombreux

articles dans la littérature traitent de ces problèmes avec tout type de méthodes exactes et approchées. Dans ce manuscrit, plusieurs approches de résolution sont expérimentées : des approches heuristiques, des modèles de programmation linéaire en nombres entiers, des modèles de programmation par contraintes, des algorithmes de programmation dynamique et une génération de colonnes.

Les articles de la littérature étudient peu fréquemment les problèmes de manière intégrée : soit la partie transport correspond à des contraintes d'un problème de production, soit la partie ordonnancement est une contrainte pour un problème de tournées de véhicules. Les études les plus récentes montrent l'intérêt de résoudre les problèmes de production et de transport de manière intégrée, bien que les approches intégrées soient plus difficiles à concevoir. Ce manuscrit s'inscrit dans cette démarche de résolution de manière intégrée.

La majorité des recherches actuelles concernant les problèmes de tournées de véhicules et/ou d'ordonnancement tend à prendre en compte des critères de qualité de service et des contraintes de coordination. Cette prise de conscience est relativement récente et rend les problèmes encore plus complexes. La qualité de service et les contraintes de coordination sont au cœur des problèmes abordés dans ce manuscrit.

Le point commun des problèmes d'ordonnancement, de tournées de véhicules et d'ordonnancement avec transport est leur modélisation sous forme de graphe où les dates de début des opérations (aussi nommées tâches, ou activités, ou visites suivant la terminologie du problème) doivent être explicitées. Pour un graphe donné acyclique, il est facile d'évaluer les dates de début au plus tôt des opérations afin de minimiser un critère dépendant d'une seule date avec un algorithme (ou fonction d'évaluation) de plus long chemin (de type Bellman-Ford ou Dijkstra). C'est notamment le cas du makespan pour les problèmes d'ordonnancement. Lorsque le critère de minimisation n'est plus le makespan, évaluer les dates de début des visites peut devenir très compliqué. Par exemple, si le critère à minimiser est un critère de qualité de service où les dates au début au plus tôt ne permettent pas d'obtenir la solution optimale. Le critère de qualité de service dépend parfois des dates de début de plusieurs opérations.

Dans ce manuscrit, trois problèmes d'ordonnancement avec transport sont abordés : le premier est le Job-shop Scheduling Problem with Routing (JSPR), le second est le Workforce Scheduling and Routing Problem (WSRP), et le troisième est le Generalised Workforce Scheduling and Routing Problem (GWSRP). Une des contributions majeures est la proposition de trois fonctions d'évaluation :

- Pour le JSPR, la fonction d'évaluation permet de prendre en compte : d'une part, un critère de qualité de service qui dépend des délais entre les dates des opérations ; et d'autre part des contraintes de coordination au sens large (des contraintes de disjonction pour être précis) entre les gammes. Cette fonction d'évaluation est basée sur les algorithmes de Bellman-Ford (Ford and Lester, 1956; Bellman, 1958; Moore, 1959) et de (Cordeau and Laporte, 2003) et est incluse dans une métaheuristique du type GRASP×ELS.
- Pour le WSRP, les gammes sont indépendantes les unes des autres, la fonction d'évaluation du graphe prend en considération un critère de qualité de service qui dépend de la date de début de chaque opération par rapport à l'employé effectuant la gamme. La fonction d'évaluation proposée est basée sur un algorithme dynamique à labels de (Feillet et al., 2004) pour résoudre un problème de type Elementary Shortest Path Problem with Resource Constraints (ESPPRC) avec qualité de service. Cette fonction d'évaluation est généralisée pour être introduite dans une méthode de résolution par génération de

colonnes.

- Pour le GWSRP, l'évaluation doit prendre en compte de nombreuses contraintes de coordination de natures différentes (disjonctive, synchronisation, time-lag minimal, time-lag maximal, etc.) et doit également prendre en considération un critère de qualité de service qui dépend de la date de début de chaque opération par rapport à l'employé effectuant la gamme. Les gammes sont dépendantes les unes des autres. L'évaluation est réalisée par un modèle de Programmation Par Contraintes.

Le Tableau 1 présente les fonctions d'évaluation qui sont utilisées dans ce manuscrit : Bellmand-Ford (Ford and Lester, 1956; Bellman, 1958; Moore, 1959), Dijkstra (Dijkstra, 1959), (Cordeau and Laporte, 2003) et (Feillet et al., 2004). Ces fonctions d'évaluation sont appliquées pour des problèmes d'ordonnancement (colonne Ordo) ou de tournées de véhicules (colonne Transport), certaines fonctions prennent en compte des contraintes de coordination. Les dernières colonnes du tableau concernent la dépendance de la fonction objectif par rapport aux dates des opérations, par exemple, pour un algorithme de type Bellman-Ford, la fonction objectif se focalise généralement sur le makespan et ne dépend donc que d'une seule date. La fonction objectif de (Cordeau and Laporte, 2003) dépend des délais entre les opérations.

Les trois dernières lignes du Tableau 1 présentent les types de fonctions d'évaluation recherchées en fonction du problème, dans ce manuscrit. Les trois problèmes traités (JSPR, WSRP et GWSRP) appartiennent aux problèmes d'ordonnancement avec transport. Pour le JSPR, la fonction d'évaluation doit permettre de prendre en compte des contraintes de coordination, et la fonction objectif dépend des délais entre les dates des opérations. Pour le WSRP, la fonction d'évaluation n'a pas besoin de prendre en considération des contraintes de coordination, mais la fonction objectif prend en compte toutes les dates des opérations. Quant au GWSRP, la fonction d'évaluation doit tenir compte à la fois de contraintes de coordination entre les gammes, et des dates de début de toutes les opérations.

Tableau 1 Fonctions d'évaluation de la littérature et fonctions d'évaluation recherchées

		Type de problème traité			**Contraintes de coordination**	**La fonction objectif dépend**			
		Ordo	Transport	Ordo + transport		D'aucune date	D'une date (makespan)	Des délais entre les dates	De toutes les dates
Fonctions d'évaluation de la littérature	Bellman-Ford	X			X		X		
	Dijkstra	X			X		X		
	(Cordeau and Laporte, 2003)		X					X	
	(Feillet et al., 2004)		X			X			
Fonction d'évaluation recherchée	**JSPR**			**X**	**X**			**X**	
	WSRP			**X**					**X**
	GWSRP			**X**	**X**				**X**

Références bibliographiques :

Bellman, R., 1958. On a routing problem. Quarterly of applied mathematics 16, 87–90.

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579–594. https://doi.org/10.1016/S0191-2615(02)00045-0

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische mathematik 1, 269–271.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44, 216–229.

Ford, J., Lester, R., 1956. Network flow theory. Rand Corp Santa Monica Ca.

Moore, E.F., 1959. The shortest path through a maze, in: Proc. Int. Symp. Switching Theory 1959. pp. 285–292.

Dans ce contexte, ce manuscrit de thèse est structuré en quatre chapitres.

Le premier chapitre est une introduction aux notions nécessaires à la bonne compréhension de ce manuscrit. Il présente les problèmes théoriques de production et les problèmes théoriques de tournées de véhicules les plus courants. Il présente également les méthodes de résolution, exactes et approchées, les plus répandues dans la littérature. La dernière partie du chapitre introduit les notions de fonctions d'évaluation, de solutions semi-actives (ou « au plus tôt »), de solutions « au plus tard », et de l'intérêt de solutions intermédiaires pour prendre en compte des critères de qualité de service.

Le second chapitre présente un problème de production et de transport de biens pour l'ordonnancement d'un atelier : le Job-shop Scheduling Problem with Routing (JSPR). Le JSPR est une extension du Job-shop Scheduling Problem with Transport dans lequel un critère de qualité de service est introduit. Ce chapitre définit donc la qualité de service pour le JSPR en prenant en compte les temps de transfert des pièces, la durée de fabrication d'une gamme et les temps d'attente des pièces sur les buffers d'entrée et de sortie des machines. Une modélisation par programmation linéaire est présentée. Le JSPR se modélise sous la forme d'un graphe disjonctif nécessitant une fonction d'évaluation pour obtenir une solution. La majorité de ces fonctions ne prennent pas en compte le critère de qualité de service. Une nouvelle fonction d'évaluation est proposée afin de minimiser le makespan (date de fin de la dernière pièce) et de maximiser simultanément la qualité de service. Les solutions obtenues ne sont donc pas semi-actives. Les résultats soulignent, d'une part, l'importance de la prise en compte d'un critère de qualité de service ; et d'autre part, la force d'une approche intégrée de résolution par rapport à une approche séquentielle.

Le troisième chapitre se focalise sur le Workforce Scheduling and Routing Problem (WSRP) où la production et le transport sont assimilés à des services à domicile prenant en compte des critères de qualité de services, simultanément, pour les clients et pour les employés. Un modèle linéaire et un modèle de programmation par contraintes sont présentés. L'une des difficultés de ce problème est que le critère de qualité de service pour les employés dépend des dates de début des services à domicile, la solution recherchée ne peut donc pas être semi-active. Par conséquent, pour une tournée donnée, trouver les dates de début maximisant la qualité de service de l'employé ne peut pas être réalisée par un algorithme classique de plus long chemin, et une procédure dynamique est proposée. Une méthode à base de génération de colonnes est présentée, avec le problème maître qui correspond soit à un Set Covering Problem, soit à un Partitioning Problem, et le problème esclave est une nouvelle extension de l'Elementary Shortest Path Problem with Resource Constraints pour prendre en compte les critères de qualité de services. Le problème esclave est résolu par programme linéaire ou par programmation dynamique.

Enfin, le quatrième chapitre introduit un nouveau problème, qui est une extension du WSRP : le Generalised Workforce Scheduling and Routing Problem (GWSRP) où des contraintes de coordination de visites sont prises en considération. Ces contraintes complexifient le problème. Une approche heuristique est proposée, elle est construite en deux grandes étapes successives : la première est basée sur l'obtention d'une solution relâchée, des contraintes de coordination, par une génération de colonnes ; la seconde est une réparation incrémentale de la solution par programmation par contraintes. Les résultats montrent que cette méthode peut résoudre d'une part des problèmes de la littérature de type Vehicle Routing and Scheduling Problem with Time Windows and Synchronized visits comprenant moins de contraintes de coordination que le GWSRP ; et d'autre part de nouvelles instances sont définies spécifiquement pour le GWSRP comprenant de nombreuses contraintes de coordination.

Les travaux de recherches réalisés durant cette thèse ont donné lieu aux publications présentées ci-dessous.

Publications

Journaux

M. Gondran, M.J. Huguet, P. Lacomme, A. Quilliot, N. Tchernev. *A Dial-a-Ride evaluation for solving the job-shop with routing consideration*. Engineering Applications of Artificial Intelligence, 2018, vol. 74, p. 70–89.

M. Gondran, S. Kemmoe-Tchomte, D. Lamy, N. Tchernev. *Bi-objective optimisation approaches to Job-shop problem with power requirements*. Expert Systems With Applications.

É. Bourreau, T. Garaix, M. Gondran, P. Lacomme, N. Tchernev. *A constraint-programming based decomposition method for the Generalised Workforce*. International Journal of Production Research.

Livres

É. Bourreau, M. Gondran, P. Lacomme, M. Vinot. *De la Programmation Linéaire à la Programmation Par Contraintes*. Ellipses, 2019.

É. Bourreau, M. Gondran, P. Lacomme, M. Vinot. *Programmation Par Contraintes : démarches de modélisation pour l'optimisation*. Ellipses, 2020.

Conférences internationales

M. Gondran, M.J. Huguet, P. Lacomme, N. Tchernev. *Comparison between two approaches to solve the Job-shop Scheduling Problem with Routing*. 9th Manufacturing Modelling, Management and Control (MIM), Berlin, Germany, August 28–30 2019.

É. Bourreau, M. Gondran, P. Lacomme. *A local search for the Job-shop Scheduling Problem with Constraint Programming*. 30th European Conference on Operational Research (EURO), Dublin, Ireland, June 23–26 2019.

É. Bourreau, M. Gondran, P. Lacomme. *Efficient Constraint Programming Approaches for routing problem: a case study for the VRP*. 7th Vehicle Routing and Logistics Optimization (VeRoLog), Seville, Spain, June 3–5 2019.

T. Garaix, M. Gondran, P. Lacomme, D. Landa-Silva, N. Tchernev. *Decomposition methods for the workforce scheduling and routing problem*. 29th European Conference on Operational Research (EURO), Valencia, Spain, July 8–11 2018.

T. Garaix, M. Gondran, P. Lacomme, E. Mura, N. Tchernev. *Workforce Scheduling Linear Programming Formulation*. 16th IFAC Synopsium on Information Control Problems in Manufacturing (INCOM), Bergamo, Italy, June 11–13 2018.

M. Gondran, P. Lacomme, N. Tchernev. *Resolution of a Job-shop with routing considering a quality of service for customer: JSSP with routing*. 7th International Conference on Industrial Engineering and Systems Management (IESM), Saarbrücken, Germany, October 11–13 2017.

M. Gondran, P. Lacomme, E. Mura, N. Tchernev. *Resolution of a Job-shop problem with precedence constrains between jobs (JSPC)*. 7th International Conference on Industrial Engineering and Systems Management (IESM), Saarbrücken, Germany, October 11–13 2017.

Keynote presentation

P. Lacomme, M. Gondran, M. Vinot. *At the crossroad of scheduling problems and routing problems*. 18th Free workshop on metaheuristics of a better world (EU/ME), Rome, Italy, April 3–4 2017.

Conférences nationales

É. Bourreau, T. Garaix, M. Gondran, P. Lacomme, N. Tchernev. *Problèmes de coordination de tournées dans le cadre du WSRP : résolution par PPC*. 20ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Le Havre, France, Février 19-21 2019.

M. Gondran, M.J. Huguet, P. Lacomme, N. Tchernev. *A Dial-a-Ride evaluation for solving the Job-shop Problem with Transport*. 19ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Lorient, France, Février 21-23 2018.

T. Garaix, M. Gondran, P. Lacomme, E. Mura, N. Tchernev. *Workforce Scheduling Resolution based on a column generation scheme*. 19ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Lorient, France, Février 21-23 2018.

Chapitre I : Présentation de la problématique

Ce chapitre introduit, d'une part, les problématiques des travaux de recherche présentés dans ce manuscrit et, d'autre part, les problèmes d'ordonnancement et de transport intégrés. Ce chapitre offre une liste non exhaustive de problèmes théoriques d'ordonnancement et de problèmes théoriques de transport, ainsi qu'une présentation des principales méthodes de résolution qui sont utilisées pour traiter ces problèmes d'optimisation.

1. Introduction

Dans le cadre de ces travaux de recherche, les problèmes étudiés se trouvent au carrefour des problèmes d'ordonnancement et de tournées de véhicules avec la prise en compte d'un critère de qualité de service lors de leur résolution. La croissance des activités de production, de distribution et de collecte ainsi que la concurrence dans ces secteurs font de cette problématique un enjeu majeur au sein des usines et des chaînes logistiques.

La résolution de manière intégrée des problèmes d'ordonnancement et de tournées de véhicules est relativement récente et s'appuie sur des méthodes issues des deux communautés (Chen, 2010; Lacomme et al., 2017; Moons et al., 2017; Gondran et al., 2018). Il est donc nécessaire de s'intéresser aux différents problèmes d'ordonnancement et de tournées de véhicules, ainsi qu'à leurs méthodes de résolution. La littérature de ces deux communautés est très riche et propose une multitude de problématiques et de méthodes. Les problèmes d'ordonnancement sont généralement classifiés suivant le type d'opérations à ordonnancer (projet, pièce), l'utilisation de machines ou de ressources, l'existence de contraintes, etc. Les problèmes de tournées de véhicules sont catégorisés suivant l'emplacement de la demande : sur les nœuds du réseau ou sur les arcs du réseau.

Les méthodes de résolution des problèmes de tournées de véhicules et les méthodes de résolution des problèmes d'ordonnancement publiées dans la littérature incluent de nombreuses méthodes exactes et de nombreuses méthodes approchées. Sans chercher à être exhaustif, la programmation linéaire (Artigues, 2017; Mingozzi et al., 1998), la programmation par contraintes (Cambazard and Bourreau, 2004; Baptiste et al., 2012), les algorithmes dynamiques (Feillet et al., 2004) et la génération de colonnes (Desaulniers et al., 2005; Feillet, 2010) font partie des méthodes de résolution exactes les plus utilisées. Les méthodes approchées sont souvent basées sur des métaheuristiques (Lawrence, 1984; Dell'Amico and Trubian, 1993; Nowicki and Smutnicki, 1996; Prins, 2004). Il est important de connaître les méthodes les plus couramment utilisées et les plus efficaces afin d'identifier leurs forces et leurs faiblesses pour connaître leurs domaines d'utilisations privilégiés.

Ce chapitre est structuré en cinq parties. La première partie (section 2) présente les systèmes de production ainsi que les niveaux de planification. Les seconde et troisième parties (section 3 et section 4) présentent les principaux problèmes théoriques d'ordonnancement et les problèmes théoriques de tournées de véhicules. La quatrième partie (section 5) met en évidence la classification des problèmes et les méthodes de résolution les plus courantes, dont certaines sont utilisées dans ce manuscrit. La cinquième – et dernière –

partie (section 6) introduit l'espace de codage et les modélisations des solutions pour des problèmes d'ordonnancement et pour des problèmes de tournées de véhicules.

2. Les systèmes de production avec transport

Les systèmes de production sont des ensembles d'unités – matérielles ou humaines – élaborés dans le but de produire des biens. Les systèmes de production sont contraints par la présence de ressources permettant à la production d'être réalisée. Les ressources dites « classiques » des systèmes de production sont les machines, les opérateurs, les matières premières, les énergies utilisées, les capacités de stockage et les véhicules d'acheminement des biens. La production de biens peut nécessiter le transport de ceux-ci entre les machines, les zones de stockages, etc. La gestion du transport est une partie intégrante du processus de production.

Le pilotage des systèmes de production avec transport est souvent traité de manière séquentielle, c'est-à-dire que, dans la majorité des cas, le problème d'ordonnancement est résolu en premier (sans intégrer le transport et en intégrant, au mieux, des délais entre les opérations) (Dell'Amico, 1996; Rebaine and Strusevich, 1999; Munier-Kordon and Rebaine, 2010). Les dates des opérations de production ainsi obtenues deviennent des contraintes pour le problème de transport qui est donc résolu dans un deuxième temps. Une intégration de ces deux problématiques dans un même schéma global de résolution permettrait d'obtenir un système global plus performant et plus proche des attentes des clients. L'intégration des problèmes d'ordonnancement et d'acheminement a reçu une attention croissante au cours de la dernière décennie (Chen, 2010; Lacomme et al., 2017; Moons et al., 2017; Fu et al., 2018; Gondran et al., 2018; Marandi and Fatemi Ghomi, 2019).

Le pilotage des systèmes de production avec transport est très présent dans les problèmes industriels avec, d'une part, la production de pièces sur des machines ; et d'autre part le transport de ces pièces, entre les machines, par des AGV (Automated Guided Vehicles) ou des chariots filoguidés (Ganesharajah et al., 1998). Mais cette problématique dépasse le cadre industriel et peut se présenter sous d'autres formes, par exemple : la fabrication de pièces peut être assimilée aux étapes d'un séjour hospitalier (rencontre de l'anesthésiste, opérations avec le chirurgien, visites des infirmières).

La gestion d'un système de production avec transport relève de trois niveaux selon l'horizon de temps impliqué (Van Looveren et al., 1986) : le niveau stratégique pour le long terme, le niveau tactique pour le moyen terme et le niveau opérationnel pour le court terme (Figure 1).

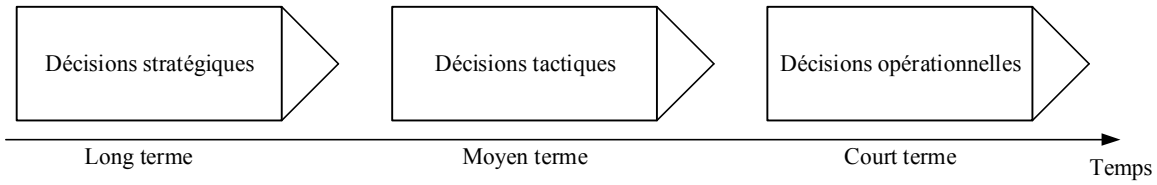


Figure 1 Les niveaux de décision dans les systèmes de production

Les décisions stratégiques de planification représentent des enjeux importants pour l'entreprise et correspondent à des décisions de configuration (ou reconfiguration) des systèmes de production. Les décisions portent sur des problématiques d'ouverture et

fermeture de ligne de production, de localisation de nouveaux sites de productions et/ou de stockage, l'achat d'équipement onéreux, la gestion du système de distributions. Ces décisions sont prises sur un horizon de planification relativement long, et souvent supérieur à deux ans.

Les décisions tactiques de planification portent sur l'utilisation du système et sur la gestion des processus opérationnels tels que la fabrication, le stockage et le transport. Ces décisions sont prises sur un horizon de temps de plusieurs mois.

Les décisions opérationnelles concernent le pilotage et le contrôle des systèmes de production et de transport sur un horizon de temps court, de quelques jours maximum. Ces décisions ont pour objet, entre autres, le réapprovisionnement des entrepôts, l'élaboration de plan de transport, la gestion des lots de production et la gestion du personnel.

Ce manuscrit porte essentiellement sur la planification opérationnelle des systèmes de production avec transport.

3. Problèmes théoriques d'ordonnancement pour la production

La littérature définit un certain nombre de problèmes d'ordonnancement qui permettent de modéliser les différents systèmes de production. Un problème d'ordonnancement est composé d'un ensemble de tâches (également appelées opérations ou activités suivant le contexte) et d'un ensemble de ressources (ou machines). Chacune des tâches est caractérisée par une durée d'exécution, ainsi qu'un ensemble de ressources qui doivent lui être allouées pour son exécution. Un système de production contient plusieurs types de ressources dans des quantités différentes (mais souvent limitées). L'objectif des problèmes d'ordonnancement est de planifier la réalisation des tâches en attribuant les ressources nécessaires et en déterminant leurs dates d'exécution (Carlier and Chrétienne, 1988).

(Toussaint, 2010) définit deux grandes familles de problème d'ordonnancement : les problèmes d'ordonnancement de projet et les problèmes d'ordonnancement d'atelier. Dans les problèmes d'ordonnancement de projet, il y a plusieurs types de ressources disponibles en quantité limitée. Une ressource peut être exploitée par plusieurs activités simultanément. Dans les problèmes d'ordonnancement d'atelier, les ressources sont des machines et ne peuvent généralement exécuter qu'une seule opération à la fois. Différents problèmes d'ordonnancement de projet et d'ordonnancement d'atelier sont présentés et leurs liens sont illustrés dans les sections suivantes.

3.1. Problèmes d'ordonnancement de projet

Les problèmes d'ordonnancement de projet sont désignés sous l'acronyme RCPSP (Resource-Constrained Project Scheduling Problem) et ils sont étudiés depuis les années 1970 (Pritsker et al., 1969). Dans ce type de problèmes, les tâches sont généralement nommées activités et requièrent plusieurs types de ressources dans des quantités différentes.

Une tâche (ou une activité) est caractérisée par sa durée, par la quantité de ressources consommées et par la quantité de ressources restituées à la fin de celle-ci. Il peut exister des ressources renouvelables telles que des kilowatts ou des ressources non renouvelables telles qu'une quantité d'argent nécessaire à l'exécution de la tâche. Chaque type de ressource est en quantité limitée. Pour commencer son exécution, une activité doit posséder la quantité exacte (ni plus ni moins) de chaque ressource nécessaire. Le fait d'affecter à une tâche une quantité

de ressource supérieure à la quantité nécessaire n'est pas pris en compte et ne présente pas d'intérêt dans les problèmes d'ordonnancement de projet qui ne prennent pas en considération le transport des ressources. Si une tâche ne possède pas suffisamment de ressources, elle doit attendre la fin d'une autre activité pour pouvoir récupérer des ressources supplémentaires.

La première publication qui utilise explicitement le terme RCPSP provient de (Pritsker et al., 1969) et, depuis, de nombreuses études ont porté sur ce problème. Le RCPSP a fait l'objet de résolution par des méthodes exactes avec par exemple : des procédures de séparation et d'évaluation (Branch-and-Bound) proposées par (Demeulemeester and Herroelen, 1992) et par (Arno Sprecher, 2000) ; programmation linéaire avec différentes relaxations par (Mingozzi et al., 1998) ; une méthode basée sur la génération de colonnes par (Brucker and Knust, 2000) ; une formulation en programmation par contraintes par (Baptiste and Demassey, 2004). Par ailleurs de nombreuses heuristiques et métaheuristiques ont également été proposées : un algorithme basé sur les règles de priorité par (Kolisch, 1996) ; a list-scheduling based algorithm par (Carlier et al., 2009). (Kolisch and Hartmann, 2006) offrent un état de l'art des différentes heuristiques utilisées pour le RCPSP. Ils proposent de les classer en quatre catégories : les approches basées sur les règles de priorités ; les métaheuristiques classiques (algorithme génétique, recherche taboue, recuit-simulé, etc.) ; les métaheuristiques non standard (recherches locales et algorithmes à population, non courants dans la littérature) ; et les autres approches heuristiques qui sont spécifiques au RCPSP.

De nombreuses extensions et variantes du RCPSP ont été proposées au cours des dernières décennies. (Hartmann and Briskorn, 2010) dressent une classification des extensions du RCPSP en cinq classes, suivant :

- La prise en compte de contraintes liées aux activités : préemptives, consommations de ressources variables au cours du temps, temps de setup, multi-modes.
- La prise en compte de contraintes temporelles : time-lag (minimal et/ou maximal), contraintes sur les dates de début et/ou de fin des activités.
- Le type de ressources : ressources non renouvelables ou partiellement renouvelables, ressource cumulative, disponibilité des ressources variables en fonction du temps.
- La fonction objectif : minimisation de la date de fin de la planification, maximisation de la robustesse, minimisation des coûts dans le cas de ressources payantes, approches multi-objectifs.
- La planification de plusieurs projets en même temps.

3.2. Problème d'ordonnancement d'atelier

Les problèmes d'ordonnancement d'atelier consistent à ordonnancer un ensemble de pièces (ou jobs) sur un ensemble de machines (les ressources). Chaque pièce est associée à une liste ordonnée (ou non) de plusieurs opérations qui définit sa gamme, et chacune de ces opérations est caractérisée par *(i)* une machine (ou un ensemble de machines) sur laquelle (ou lesquelles) l'opération peut avoir lieu et par *(ii)* un temps de traitement (processing time). Dans la plupart des cas, l'ordre des opérations définissant la gamme et la durée de traitement des opérations est connu. La grande différence avec les problèmes d'ordonnancement de projet est que pour les problèmes d'ordonnancement d'atelier, une ressource (ou machine) ne peut exécuter qu'une seule opération à la fois.

De manière générale, une solution d'un problème d'atelier doit définir : l'ordre de passage des opérations sur chaque machine et les dates de début de chaque opération. Dans le cas où une opération est caractérisée par un ensemble de machines sur lesquelles l'opération peut avoir lieu, la solution doit également définir l'affectation d'une machine à l'opération. Le critère le plus souvent utilisé pour les problèmes d'ordonnancement est le critère du « makespan », c'est-à-dire, la date de fin de la dernière opération sur la dernière machine.

La liste suivante regroupe les problèmes d'ordonnancement d'atelier dit « classiques » de la littérature et les problèmes sont classés du plus spécifique au plus général :

- Le problème à une machine, défini par (Graham et al., 1979; Yin et al., 2015) : dans ce cas l'atelier contient une unique machine et chaque gamme comporte une unique opération qui doit être donc réalisée par l'unique machine. Les durées des opérations varient d'une gamme à l'autre. Les systèmes de production complexes sont régulièrement décomposés en problèmes à une machine (Pinedo, 2012).
- Le problème du Flow-shop (Garey et al., 1976; Rossit et al., 2018) : les gammes sont constituées de plusieurs opérations (le problème à une machine est un cas particulier du Flow-shop). Pour chaque gamme l'ordre des opérations est le même et chaque opération ne peut être traitée que par une machine (l'affectation d'une machine à une opération fait partie des données du problème). Le Flow-shop est un cas particulier du Job-shop.
- Le problème du Job-shop (Manne, 1960; Blazewicz et al., 1996; Jain and Meeran, 1999; Çaliş and Bulkan, 2015) : les gammes sont toutes différentes les unes des autres et chaque opération ne peut être réalisée que par une seule machine qui est une donnée du problème. Le Job-shop est un cas particulier du problème de l'Open-shop.
- Le problème de l'Open-shop (Gonzalez and Sahni, 1976; Liaw, 2000; Leung, 2004; Sha and Hsu, 2008) : la machine permettant d'exécuter chaque opération est connue, mais l'ordre des opérations dans chaque gamme n'est pas une donnée du problème.
- Le problème du Group-shop (Sampels et al., 2002; Ahmadizar and Rabanimotlagh, 2014) est un croisement entre les problèmes du Job-shop et de l'Open-shop : certaines opérations d'un job peuvent être regroupées et traitées comme dans le cas de l'Open-shop tandis que d'autres opérations ont un ordre défini par les données du problème similairement au Job-shop.

Pour les problèmes présentés précédemment, l'ensemble des machines pouvant exécuter une opération est un singleton et une donnée d'entrée du problème, il n'y a pas besoin d'affecter une machine à l'opération en faisant un choix parmi un ensemble de machines qui serait spécifié par la gamme. Il existe toutefois de nombreuses extensions à ces problèmes d'ordonnancement qui font intervenir un ensemble de machines pouvant effectuer une opération. Une opération n'est plus caractérisée par une machine sur laquelle elle doit passer, mais par un ensemble de machines parmi lesquelles une machine doit réaliser l'opération, et il y a donc un problème d'affectation à résoudre en plus. Ce problème d'affectation rend, dans la majorité des cas, le problème plus difficile à résoudre. Les extensions les plus courantes des problèmes d'ordonnancement d'atelier sont listées ci-dessous :

- Le problème à machines parallèles homogènes ou hétérogènes (Horn, 1973; Cheng and Sin, 1990; Fanjul-Peyro and Ruiz, 2010) : chaque gamme contient une unique opération, mais cette opération possède un ensemble de machines susceptibles de la réaliser, et l'opération doit donc être affectée à une machine.
- Les problèmes de Flow-shop Flexible (Brah and Hunsucker, 1991; Dai et al., 2013; Lee and Loong, 2019) et de Flow-shop Hybrid (Hunsucker and Shah, 1994; Linn and Zhang, 1999; Ruiz and Vázquez-Rodríguez, 2010) : chaque opération peut s'exécuter sur un

ensemble de plusieurs machines regroupées en étage. Les machines d'un même étage peuvent être identiques, cas du Flow-shop Flexible, ou bien différentes, cas du Flow-shop Hybrid.

- Le problème du Job-shop Flexible (Brucker and Schlie, 1990; Chaudhry and Khan, 2016) : chaque opération doit être réalisée par une machine issue d'un ensemble de machines susceptibles d'exécuter l'opération.

Ces problèmes d'ordonnancement d'atelier étaient initialement étudiés sous une forme déterministe, mais de plus en plus de recherches visent à étendre ces problèmes à des formes stochastiques (Gu et al., 2009) et réactives (just-on-time) (Gabel and Riedmiller, 2008). Pour les problèmes d'ordonnancement sous forme stochastique, les durées de traitement des opérations sont des réalisations de variables aléatoires et des pannes de machines peuvent être prises en compte. Un des objectifs du problème sous forme stochastique est de trouver un ordonnancement dit « robuste » qui est capable de « résister » aux évènements aléatoires en répondant à des questions telles que : « si une machine tombe en panne quel va être son impact sur le reste de la planification ? ». Dans les problèmes réactifs, les opérations ne sont pas connues en avance, et à chaque nouvel ordre de fabrication la nouvelle opération doit être ordonnancée : les décisions sont prises au fur et à mesure que les opérations arrivent.

En plus des formes déterministes et stochastiques ou des planifications réactives, il existe également des extensions des problèmes d'ordonnancement prenant en compte des contraintes supplémentaires entre les opérations ou les gammes, par exemple le Job-shop Scheduling Problem with Time-lag (Caumond et al., 2008; Artigues et al., 2011). Dans ce problème les opérations d'une même gamme ne peuvent pas être séparées de plus d'une certaine durée, et cette durée est modélisée par un time-lag maximal. (Kemmoé et al., 2015) propose un Job-shop avec contraintes financières où le flux de produit et le flux de trésorerie sont simultanément considérés : la réalisation d'une opération nécessite des fonds tandis qu'un produit terminé permet une entrée d'argent.

La Figure 2 présente les deux principaux d'ordonnancement d'atelier et leurs liens selon (Toussaint, 2010).

La différence entre les problèmes d'ordonnancement d'atelier et les problèmes d'ordonnancement de projet est qu'une opération dans un problème d'ordonnancement de projet peut consommer plusieurs ressources simultanément ; tandis qu'une activité dans un problème d'ordonnancement d'atelier n'utilise qu'une seule ressource (une unique machine).

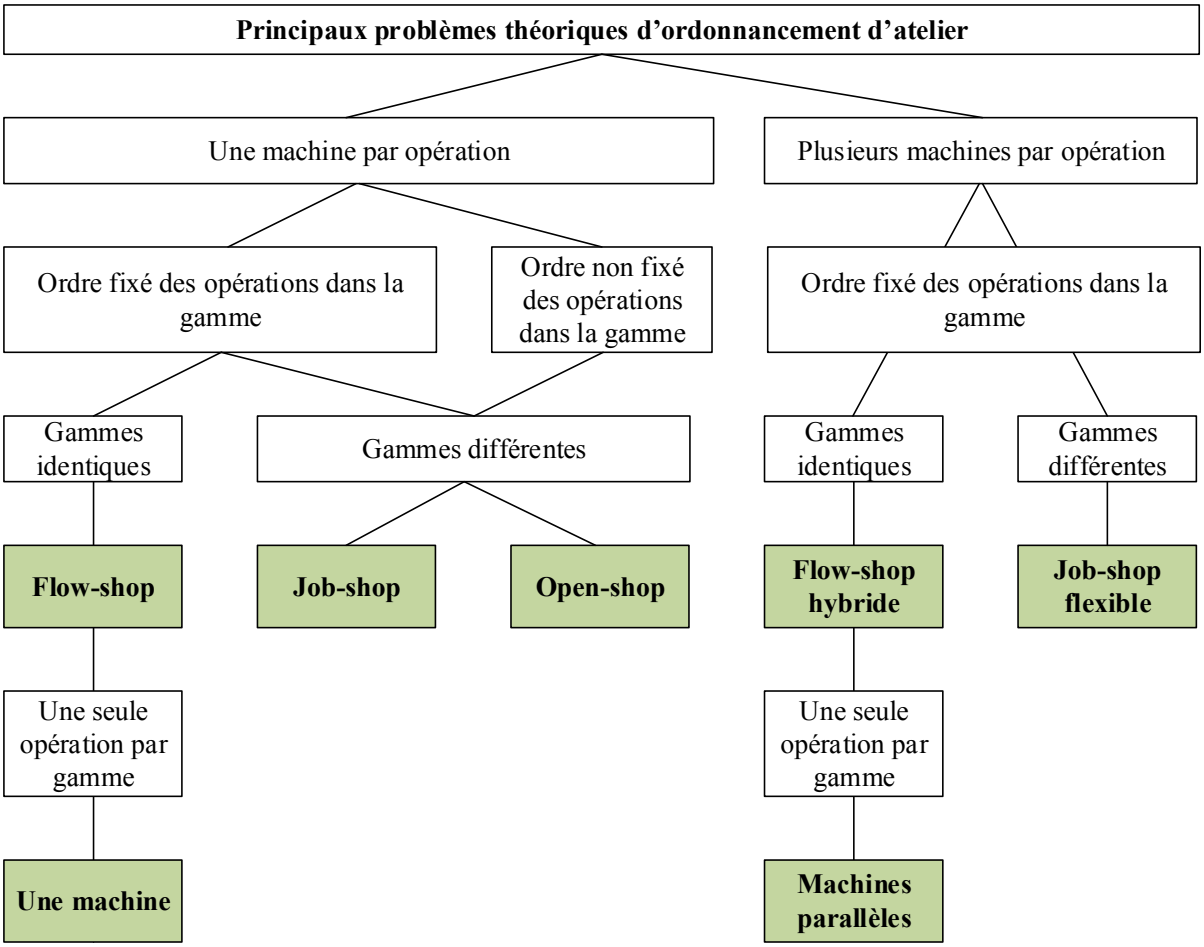


Figure 2 Classification des problèmes d'ordonnancement d'atelier selon (Toussaint, 2010)

4. Problèmes théoriques de tournées de véhicules

Le transport, la collecte et la distribution de biens ou de personnes constituent un des enjeux majeurs de la chaîne logistique et présentent de nombreux intérêts sociaux, économiques et environnementaux. Ces enjeux sont de plus en plus cruciaux dans nos sociétés actuelles où la recherche de gain de temps et/ou l'amélioration des performances sont omniprésentes. L'ensemble des problèmes d'optimisation concernant le transport, la collecte et la distribution est regroupé sous la dénomination de problème de tournées (de véhicules) ou « routing problems ». Cette appellation générique regroupe un grand nombre de problèmes avec des caractéristiques très différentes.

Les problèmes de tournées de véhicules sont composés d'un ensemble de clients, d'un ensemble de véhicules et d'un graphe modélisant un réseau routier. L'objectif de ces problèmes est de déterminer : d'une part, à quel véhicule est affecté chaque client ; et d'autre part, dans quel ordre un véhicule visite les clients qui lui sont affectés. Ces deux considérations sont souvent très fortement liées.

Généralement, les problèmes de tournées de véhicules sont divisés en deux classes en fonction de la position des clients.

1) Si les clients sont situés sur les arcs du graphe, il s'agit alors d'un problème de tournées sur arcs (Arc Routing Problem – ARP), par exemple : le problème du postier chinois

(Chinese Postman Problem – CPP) (Eiselt et al., 1995) et le Capacitated Arc Routing Problem (CARP) (Golden and Wong, 1981).

2) Si les clients sont localisés sur des points précis du réseau, alors il s'agit d'un problème de tournées sur nœuds (Node Routing Problem – NRP), par exemple : les problèmes de visites de clients avec notamment le problème du voyageur de commerce (Traveling Salesman Problem – TSP) (Dantzig et al., 1954; Gutin and Punnen, 2006) et le Vehicle Routing Problem (VRP) (Dantzig and Ramser, 1959; Braekers et al., 2016) ; les problèmes généraux de collecte et livraison (General Pickup and Delivery Problems – GPDP) qui modélisent le transport de biens (Toth and Vigo, 2014) ; et le Dial-A-Ride Problem (DARP) qui modélise le transport de personnes (Stein, 1978; Cordeau and Laporte, 2003).

La Figure 3 représente les différentes classes des problèmes de tournées de véhicules.

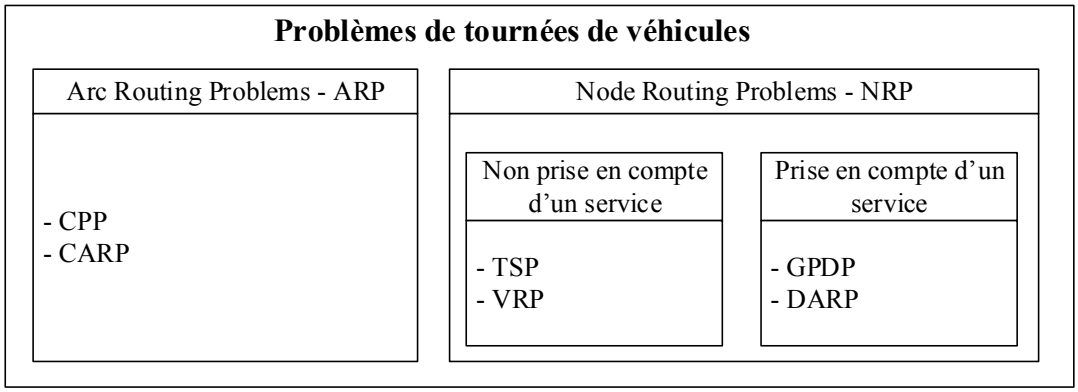


Figure 3 *Les classes des problèmes de tournées de véhicules*

Dans la majorité des problèmes de tournées de véhicules, l'objectif est de minimiser une distance ou un coût lié à la distance parcourue. Ce type de critère est celui utilisé pour le TSP ou encore le VRP (Braekers et al., 2016). Pour ces problèmes, le critère de la distance n'est pas fonction des dates de passage des véhicules sur les nœuds. Les problèmes de type GPDP et DARP prennent en compte un critère de qualité de service et la fonction objectif dépend généralement des dates de passages des véhicules chez les clients (Cordeau and Laporte, 2003; Firat and Woeginger, 2011). De plus, dans ces problèmes les clients ont des contraintes sur le temps passé dans le véhicule et les tournées ont des contraintes sur leur durée totale. Si les tournées sont connues, il est relativement facile d'obtenir les dates de passage des véhicules sur les nœuds pour les problèmes de type TSP et VRP. L'évaluation de ces dates requiert des algorithmes plus complexes si la notion de qualité de service entre en jeu, comme pour le GPDP et le DARP.

Pour les recherches présentées dans ce manuscrit, la modélisation du transport est proche de celles de problèmes de tournées sur nœuds en particulier des General Pickup and Delivery Problems (GPDP), car les biens doivent être transportés d'un sommet à un autre du réseau. Les prochaines sections se concentrent donc sur ces types de problèmes.

4.1. Problème de visites de clients

Un des problèmes de tournées de véhicules parmi les plus connus est le problème du voyageur de commerce (Traveling Salesman Problem – TSP) formalisé par (Dantzig et al.,

1954). L'objectif du TSP consiste à trouver la route (ou tour), permettant à un voyageur de commerce de partir d'une ville (généralement nommée dépôt), de visiter une unique fois toutes les villes et de rentrer dans sa ville de départ, en minimisant la distance parcourue. Le TSP est l'objet de nombreuses publications et d'états de l'art (Larranaga et al., 1999; Gutin and Punnen, 2006; Applegate et al., 2011).

Le problème du TSP est modélisé sous forme de graphe dans lequel les sommets représentent les villes à visiter et les arcs modélisent les routes entre les différentes villes. Chacun des arcs (les routes donc) reliant deux sommets (deux villes) est pondéré par un poids qui correspond à la distance à parcourir entre ces deux villes. Une solution du TSP consiste à définir l'ordre dans lequel le voyageur doit visiter un ensemble des villes et correspond à cycle hamiltonien dont la somme des distances est minimale.

Le Vehicle Routing Problem (VRP) est une extension du TSP, introduite par (Dantzig and Ramser, 1959). Dans ce problème, l'objectif est similaire à celui du TSP, et consiste à visiter l'ensemble des villes en minimisant la distance totale parcourue (Toth and Vigo, 2002). Pour le VRP, la terminologie utilisée n'est plus « villes », mais « clients » (qui sont toujours représentés par les sommets du graphe). Dans le cas du TSP, il n'y a qu'un unique véhicule (ou voyageur) qui doit passer par toutes les villes, tandis que dans le cas du VRP, il s'agit d'une flotte composée de plusieurs véhicules qui doivent visiter l'ensemble des clients. Une tournée est définie par un véhicule et une route, débutant à un dépôt et terminant au même dépôt, qui permet de visiter un sous-ensemble de clients. Une solution doit respecter les contraintes suivantes : chaque client n'est visité qu'une unique fois par une et une seule tournée ; chaque véhicule effectue au plus une tournée ; et tous les clients doivent être visités par un véhicule. Les livres de (Toth and Vigo, 2002; Gendreau et al., 2008; Toth and Vigo, 2014) sont des références internationales concernant les problèmes de VRP et de leurs extensions. (Braekers et al., 2016) proposent un état de l'art récent du VRP prenant en compte 277 articles publiés entre 2009 et juin 2015.

Les problèmes du TSP et du VRP sont la base de nombreux autres problèmes de tournées de véhicules qui sont définis en ajoutant différentes contraintes et/ou objectifs (Toth and Vigo, 2002, 2014). Le VRP comporte souvent des contraintes sur la capacité des véhicules (Capacited Vehicle Routing Problem – CVRP), ou sur la longueur maximale des tournées, ce qui permet de modéliser les contraintes d'autonomie des véhicules ou de législation. Pour le CVRP, une demande (de collecte ou de livraison) à respecter est associée à chaque client et à chaque véhicule est associée une capacité maximale de demandes qu'il peut satisfaire. Une autre extension du VRP fait régulièrement l'objet d'étude : le VRPTW (Vehicle Routing Problem with Time Window) (Pureza et al., 2012; Vidal et al., 2013), dans ce cas, chaque client possède une fenêtre de temps (time window) pendant laquelle le client doit être visité. (Gendreau and Tarantilis, 2010) introduisent un état de l'art se focalisant uniquement sur les problèmes de VRPTW et (Lahyani et al., 2015) présentent un état de l'art des problèmes de type VRP qui sont enrichis de contraintes additionnelles.

4.2. Problèmes généraux de collecte et livraison

La définition de la classe des problèmes de collecte et livraison (GPDP) est proposée par (Savelsbergh and Sol, 1995). Dans ces problèmes, il ne suffit plus de visiter les différents clients comme pour les problèmes de TSP et de VRP ; il faut en plus prendre en compte une demande des clients pour une quantité de produits ou un service. (Berbeglia et al., 2007; Parragh et al., 2008) proposent des états de l'art des problèmes de type GPDP.

Le Vehicle Routing Problem avec collecte et livraison (VRP with Pickup and Delivery – VRPPD) est une extension du VRP où chaque client requiert une quantité de produits, et les véhicules ont une capacité maximale (Dumas et al., 1991; Tasan and Gen, 2012; Pradenas et al., 2013). Une solution est un ensemble de tournées telles que toutes les demandes soient satisfaites (il est interdit de livrer plus ou moins de produits), et que la charge maximale des véhicules soit respectée. L'objectif est de minimiser la distance totale parcourue.

De nombreuses contraintes peuvent être ajoutées au VRPPD, notamment des fenêtres de temps (Dumas et al., 1991) ; une flotte de véhicules hétérogènes ; plusieurs dépôts ; autonomie du véhicule (temps de trajet maximal pour le véhicule) ; temps de trajet maximal pour les produits ; etc. Une extension courante concerne la durée du service effectué chez le client : cette durée peut être nulle ou non. Dans le premier cas, le véhicule arrive chez un client et peut repartir aussitôt, dans le second cas, le véhicule doit attendre un certain temps avant de pouvoir repartir.

Une fenêtre de temps impose que la requête (livraison, collecte ou autre service) du client soit traitée dans un intervalle précis défini par une date au plus tôt et une date au plus tard. Cette contrainte modifie la façon de modéliser et de résoudre les problèmes de tournées de véhicules. En effet, dans un problème de VRPPD sans fenêtre de temps, une solution est simplement composée d'un ordre des visites pour chaque véhicule. Dans un VRPPD avec fenêtre de temps (VRPPDTW) (Sexton and Bodin, 1985), un véhicule peut arriver avant la fenêtre de temps du client, mais dans ce cas il doit attendre l'ouverture de celle-ci pour pouvoir commencer son service. Toutefois, le véhicule ne peut pas arriver après la fenêtre de temps. Une solution du VRPPDPTW doit contenir les tournées de chaque véhicule, mais également la date d'arrivée du véhicule chez le client, la date de début du service, la date de fin de service et la date de départ de chez le client.

La Figure 4, d'après (Chassaing, 2015), illustre par deux exemples l'arrivée du véhicule chez un client possédant une fenêtre de temps. Dans le premier cas, le véhicule arrive dans la fenêtre de temps du client et peut commencer son service tout de suite. Dans le second cas, le véhicule arrive avant la fenêtre de temps et doit attendre avant de débuter son service.

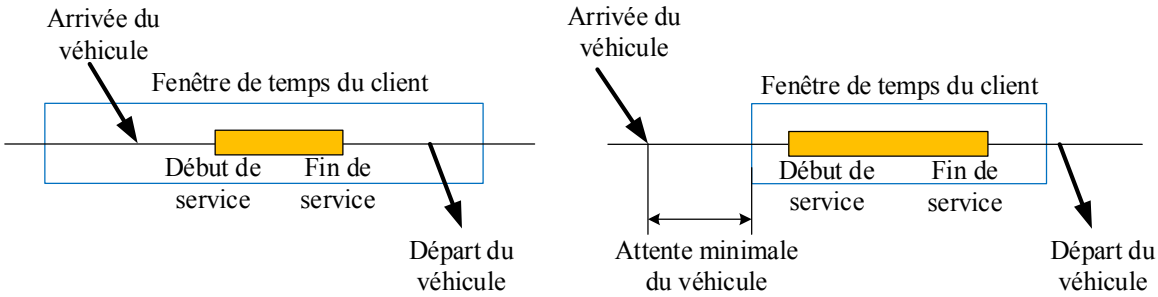


Figure 4 Exemples de fenêtres de temps pour un client (Chassaing, 2015)

Le problème du transport à la demande (Dial-A-Ride Problem – DARP) étend le VRPPDTW en prenant en considération des contraintes de durées de transport maximales et une fonction objectif avec une notion de qualité de service. L'un des premiers articles sur le DARP est proposé par (Stein, 1978) qui propose un modèle mathématique. À l'origine les problèmes de DARP concernent le transport de personnes, mais ils peuvent également être utilisés pour résoudre des problèmes de transport de biens notamment si ces derniers ont des contraintes sur leurs durées maximales de transport (transports de produits dangereux, durée de vie du produit, etc.) (Toth and Vigo, 2014). Le DARP est l'objet de nombreuses études

(Cordeau and Laporte, 2003, 2007; Firat and Woeginger, 2011). (Ho et al., 2018) proposent un état de l'art du DARP prenant en compte les articles publiés depuis 2007.

(Cordeau and Laporte, 2003) font partie des premiers à proposer une qualité de service pour les problèmes de tournées de véhicules. Ils proposent un DARP prenant en compte, dans la fonction objectif, le temps total d'attente des clients, le temps total de trajet, la durée totale de la tournée et le respect des fenêtres de temps. (Cordeau and Laporte, 2003) promeuvent une nouvelle manière de calculer les dates de services pour obtenir une solution de compromis entre les différents critères de la fonction objectif.

Les GDPD et le DARP sont très intéressants dans les enjeux de pilotage des systèmes de production avec transport, car généralement, les matières premières et les biens (en cours de production ou finis) doivent être transportés à l'intérieur même de l'usine. Souvent plusieurs véhicules (AVG, filoguidés) sont disponibles pour effectuer ces transports. De plus, le problème de transport n'a pas souvent de critère de qualité associé, et est simplement considéré comme un sous-problème du problème d'ordonnancement. Or prendre en compte la qualité de service dans le transport peut avoir un grand impact sur le problème d'ordonnancement.

5. Méthodes de résolution

Les problèmes d'optimisation combinatoire consistent à trouver, dans un ensemble discret et fini de solutions, une solution optimale (il peut exister plusieurs solutions optimales, généralement une seule est recherchée). Trouver une solution dans un ensemble fini et discret peut sembler facile avec une stratégie d'énumération de toutes les solutions et de les comparer entre elles pour obtenir la meilleure. Mais le nombre de solutions d'un problème combinatoire est souvent exponentiel, rendant impossible une telle stratégie. La méthode de résolution choisie doit donc tenir compte de la complexité du problème.

Cette section présente la complexité et les différentes classes des problèmes combinatoires, puis les principales méthodes de résolution de ces problèmes : d'abord les méthodes exactes et ensuite les méthodes approchées.

5.1. Complexité et classe de problèmes

Les problèmes d'optimisation combinatoire sont classifiés selon leur complexité. La complexité d'un problème est définie par rapport aux algorithmes utilisés pour le résoudre, c'est l'algorithme de plus faible complexité qui détermine la classe du problème (Turing, 1937; Cook, 1971; Garey and Johnson, 1979; Toussaint, 2010).

La définition de la complexité d'un algorithme est liée aux machines de Turing (Turing, 1937), il en existe deux différentes : les déterministes et les non-déterministes. Les machines déterministes exécutent les étapes de l'algorithme les unes après les autres, chaque nouvelle étape est déterminée par la précédente. Pour les machines de Turing non-déterministes, à chaque nouvelle étape de l'algorithme, elles disposent d'un panel d'étapes possibles à exécuter (et non plus une seule étape), sans déterminer explicitement quelle étape doit être exécutée.

Un algorithme est dit polynomial (ou polynomial en temps), si le nombre d'opérations qu'il effectue suit une fonction polynomiale. Sa complexité est notée, d'après les notations de Landau, $O(n^k)$, avec $k \in \mathbb{N}$, et n la taille des données d'entrée. La complexité est dite constante lorsque $k = 0$ et linéaire lorsque $k = 1$. Il existe d'autres ordres de grandeurs pour la complexité des algorithmes, notamment $O(\log(n))$ qui est une complexité logarithmique, et $O(a^k)$, avec $a \in \mathbb{N}$, $a > 1$, qui est une complexité exponentielle.

Parmi les différentes classes de complexité, uniquement les classes *P*, *NP* et *NP-complet* sont présentées dans ce manuscrit, car elles ont un rôle important pour les problèmes d'ordonnancement et de tournées de véhicules.

Définition : classe P

La classe P (Polynomiale) contient l'ensemble des problèmes qui peuvent être résolus avec un algorithme polynomial en temps par une machine de Turing déterministe (Garey and Johnson, 1979).

Les problèmes de la classe *P* sont généralement considérés comme « faciles ».

Définition : classe NP

Un problème appartient à la classe NP (Non-deterministic Polynomial) si une solution de celui-ci peut être vérifiée en un temps polynomial (Garey and Johnson, 1979).

La classe *NP* contient la majorité des problèmes d'optimisation combinatoire. Tout problème issu de la classe *P* est également contenu dans la classe *NP*, car si un problème peut être résolu avec un algorithme polynomial, alors sa solution peut également être vérifiée en temps polynomial. Savoir si $P = NP$ reste à ce jour un problème ouvert qui interpelle de nombreux chercheurs et qui est un des 7 problèmes du prix du millénaire.

La classe des problèmes *NP-complet* découle de la classe des problèmes *NP* et repose sur la définition de réduction polynomiale. Un problème $\wp$ peut être réduit polynomialement à un problème $\wp'$ si les données du problème $\wp$ peuvent être transformées en données du problème $\wp'$ par un algorithme polynomial. Grâce à la réduction polynomiale, si un algorithme polynomial peut résoudre le problème $\wp$, alors le problème $\wp'$ peut être résolu avec le même algorithme.

Définition : un problème $\wp$ appartient à la classe NP-complet (Garey and Johnson, 1979) si :

(i) $\wp \in NP$: le problème appartient à la classe NP ;

(ii) Tout autre problème de NP peut être réduit polynomialement à $\wp$.

Le premier problème prouvé *NP-complet* est le problème SAT (problème de satisfaisabilité booléenne) par (Cook, 1971). S'il existe un algorithme polynomial pour résoudre un problème *NP-complet*, alors tous les problèmes de la classe *NP* pourraient être résolus en un temps polynomial.

Initialement, les classes *P, NP* et *NP-complet* concernent les problèmes de décision, mais par abus de langage, les problèmes d'optimisations sont inclus dans ces classes, car il est considéré qu'un problème d'optimisation peut toujours être transformé en problème de décision.

Les problèmes issus des classes *NP* et *NP-complet* focalisent l'attention des chercheurs, car ceux-ci ne peuvent généralement pas être résolus de manière exacte pour des problèmes de tailles moyennes et importantes dans des temps de calcul raisonnables. Les prochaines sections présentent des méthodes de résolution exactes pour des problèmes d'optimisation, qui ont pour but de prouver l'optimalité de la solution, mais qui sont souvent très longues ; et des méthodes de résolution approchées qui ont pour objectif de trouver rapidement une solution de qualité satisfaisante, mais pas forcément optimale.

5.2. Résolution par méthodes exactes

5.2.1. Programmation linéaire

La programmation linéaire (Dantzig, 2016) consiste à résoudre un modèle linéaire (aussi appelé « programme linéaire ») de manière exacte, communément, avec la méthode du simplex (Dantzig, 1987) ou la méthode des points intérieurs (Karmarkar, 1984). Les programmes linéaires modélisent les contraintes du problème sous forme d'équations et d'inéquations toutes linéaires (il n'y a pas de terme quadratique, ou d'opérateur logique), il est également impossible de multiplier/diviser des variables entre elles. La fonction objectif du problème doit elle aussi être linéaire. Il existe trois grandes familles de programme linéaire : les programmes linéaires « simples », les programmes linéaires en nombres entiers et les programmes linéaires mixtes.

Un programme linéaire « simple » ne contient que des variables continues, c'est-à-dire, les variables peuvent prendre n'importe quelle valeur réelle dans un intervalle donné. La méthode du simplexe, bien que non-polynomiale dans le pire cas, est très performante pour ce type de modèle, car elle est souvent polynomiale en moyenne.

Un Programme Linéaire en Nombres Entiers (PLNE) ne fait intervenir que des variables entières. Ces problèmes sont généralement plus difficiles à résoudre que les programmes linéaires simples, en raison de la difficulté à respecter la contrainte d'intégrité des variables. Les PLNE sont fréquemment résolus en relaxant la contrainte d'intégrité des variables puis en combinant des méthodes du simplex ou des points intérieurs avec des méthodes de Branch-and-Bound ou Branch-and-Cut pour obtenir une solution en nombres entiers.

Un programme linéaire mixte (Mixed Integer Linear Programming – MILP) comprend à la fois des variables continues et des variables entières. La méthode de résolution d'un MILP est semblable à la résolution d'un PLNE.

La programmation linéaire est utilisée pour de nombreux problèmes de tournées de véhicules (Fügenschuh, 2009; Rasmussen et al., 2012; Braekers et al., 2014) ; des problèmes d'ordonnancement (Stein, 1978; Mingozzi et al., 1998; Bruzzone et al., 2012; Artigues, 2017) ; et des problèmes intégrés d'ordonnancement et de transport (Castillo-Salazar et al., 2016; Garaix et al., 2018; Gondran et al., 2018).

La programmation linéaire est régulièrement mise en difficulté lorsque le nombre de variables est très important, notamment pour des instances de grandes tailles. Dans ce cas le temps de résolution devient très important. Dans certains cas, la quantité de mémoire consommée par les solveurs atteint plusieurs gigaoctets et dépasse la capacité des ordinateurs.

Afin de pallier ces deux problèmes, il existe des méthodes de décomposition, les plus courantes sont la méthode de Dantzig-Wolfe (Dantzig and Wolfe, 1960) et la génération de colonnes.

La programmation linéaire (et les méthodes de décomposition) est un outil très utilisé dans les communautés des problèmes d'ordonnancement et de tournées de véhicules, mais est souvent limitée, face à des problèmes qui comportent un grand nombre de contraintes logiques du type « Si… Alors… Sinon ». La modélisation de ces contraintes fait intervenir une variable binaire, une constante M (ou H), et deux contraintes (un pour le Si, l'autre pour le Sinon). Ce type de contrainte ralentit énormément les solveurs en programmation linéaire. Il existe un autre grand courant de modélisation et de résolution exacte qui est la programmation par contraintes.

5.2.2. Génération de colonnes

Face à des programmes linéaires de grande taille, la génération de colonnes s'appuie sur trois constats : le modèle linéaire comporte un trop grand nombre de variables pour qu'elles puissent toutes être explicitées ; la grande majorité des variables est nulle dans la solution optimale ; et la méthode du simplexe n'a pas besoin d'accéder simultanément à toutes les variables du problème. La génération de colonnes est particulièrement adaptée à la résolution de problème possédant un très grand nombre de variables, car l'ajout itératif de variables peut permettre d'espérer obtenir une solution optimale en n'ayant considéré qu'un sous-ensemble de variables. (Barnhart et al., 1998; Desaulniers et al., 2005) proposent deux états de l'art concernant la génération de colonnes.

L'objectif de la génération de colonnes est de résoudre un problème restreint en nombre de colonnes, c'est-à-dire, de taille réduite avec un ensemble limité de variables, puis de générer les variables (ou colonnes) utiles, de manière itérative, jusqu'à l'obtention d'une solution optimale. La génération d'une nouvelle colonne se fait lors de la résolution d'un sous-problème (ou problème esclave) et consiste à chercher la meilleure variable à ajouter dans le problème restreint, c'est-à-dire la variable de coût réduit minimal. Le coût réduit d'une variable est calculé à partir des variables duales obtenues après la résolution du sous-problème restreint. Le problème initial à résoudre est appelé problème maître (ou problème principal).

Sans perte de généralité, soit le problème initial (ou maître) ($\wp$) suivant :

$$(\wp) \qquad \min\left(\sum_{i\in T} c_i x_i\right)$$

$$\text{s. c.} \qquad \begin{cases} \sum_{i\in T} a_{ij} x_i \geq b_j & \forall j = 1, \dots, n \\ x_i \geq 0 & \forall i \in T \end{cases}$$

Avec :

- T, l'ensemble des indices des variables ;
- $c \in \mathbb{R}^{|T|}$, les coefficients de la fonction objectif ;
- $x \in \mathbb{R}^{|T|}$, le vecteur des variables ;
- $A = (a_{ij}) \in \mathcal{M}_{|T|n}$, la matrice des contraintes ;

- $b \in \mathbb{R}^n$, les coefficients du second membre ;
- λ_j ($\forall j = 1, \dots, n$), la variable duale associée à la contrainte j.

Le problème restreint (RLP), de la génération de colonnes, à l'itération i est le suivant, avec $R_i \subseteq T$ l'ensemble restreint des variables utilisées à l'itération i :

$$(RLP) \quad \min \left(\sum_{i \in R_i} c_i x_i \right)$$

$$\text{s. t.} \quad \begin{cases} \sum_{i \in R_i} a_{ij} x_i \geq b_j & \forall j = 1, \dots, n \\ x_i \geq 0 & \forall i \in R_i \end{cases}$$

La résolution du (RLP) permet d'obtenir les valeurs optimales des variables duales λ_j associées aux contraintes. Toute colonne $k \in T \backslash R_i$ est susceptible de diminuer la valeur de la fonction objectif si celle-ci a un coût réduit négatif (cas d'un problème de minimisation). Le coût réduit $\overline{c_k}$ d'une variable k est donné par :

$$\overline{c_k} = c_k - \sum_{j=1}^{n} a_{kj} \lambda_j$$

S'il n'y a aucun coût réduit de valeur négative, alors il n'existe pas de variable améliorante et la solution du problème restreint est la solution optimale du problème maître. Si une variable x_k a un coût réduit négatif, alors celle-ci est intégrée au problème (RLP) ($R_{i+1} = R_i \cup k$). Il existe plusieurs stratégies pour sélectionner la variable à ajouter dans le (RLP), la stratégie de base est de choisir la variable avec le coût réduit négatif minimal, il existe d'autres stratégies consistant à sélectionner plusieurs variables (de coût réduit négatif) à chaque itération.

La génération de colonnes fonctionne lorsque le problème restreint (RLP) est réalisable, car celui-ci possède donc une solution duale qui permet la recherche de variables améliorantes. Dans certains cas, le (RLP) peut ne pas être réalisable, notamment dans les cas où : trouver une solution réalisable (sans considération pour sa qualité) de ($\wp$) est très difficile, il est alors compliqué de trouver un ensemble de variables R_i permettant d'initialiser le (RLP) ; la génération de colonnes est associée à un Branch-and-Price dans le cadre d'un problème ($\wp$) impliquant des nombres entiers, le branchement peut rendre le (RLP) irréalisable.

La génération de colonnes est régulièrement utilisée pour les problèmes de tournées de véhicules et d'ordonnancement. Il est intéressant de noter qu'il existe de nombreuses méthodes pour accélérer la résolution de la génération de colonnes, (Desaulniers et al., 2001), (Desaulniers et al., 2005), (Feillet, 2010) listent notamment les méthodes suivantes :

- effectuer la résolution du sous-problème avec la programmation dynamique ;
- ne pas résoudre les sous-problèmes à l'optimalité lors des premières itérations ;
- ajouter plusieurs colonnes à partir de la résolution d'un sous-problème (et pas uniquement une seule colonne).

5.2.3. Programmation par contraintes

La Programmation Par Contraintes (PPC) est une méthode de résolution exacte des problèmes combinatoires qui consiste à modéliser les problèmes par un ensemble de contraintes, qui ne sont pas forcément linéaires. Les contraintes non linéaires peuvent être, entre autres, des relations logiques de type « Si… alors » ou de type x_y avec x et y deux variables. La programmation par contraintes combine des techniques issues de la recherche opérationnelle dont la théorie des graphes et la programmation linéaire ; avec des modèles de représentation et de traitement des contraintes issues de l'intelligence artificielle, notamment la satisfaction des contraintes, la propagation des contraintes et des langages déclaratifs de modélisation. Les problèmes de planification et d'ordonnancement des systèmes de productions, ainsi que les problèmes de tournées de véhicules font partie des applications courantes de la PPC (Wallace, 1996; Cambazard and Bourreau, 2004; Rossi et al., 2006; Baptiste et al., 2012; Hojabri et al., 2018; Bourreau et al., 2019, 2020).

Un point clé de la programmation par contraintes est la séparation de la modélisation du problème et de sa résolution. Le problème est modélisé sous forme de « problème de satisfaction de contraintes » (Constraint Satisfaction Problem – CSP). La résolution du CSP s'appuie sur trois principes : le filtrage, la propagation et la recherche de solution par exploration. Le filtrage et la propagation permettent de réduire de l'espace de recherche, et l'exploration complète de cet espace prouve l'existence – ou non – d'une solution réalisable, ou prouve l'optimalité de la solution dans le cas d'un problème d'optimisation.

De manière plus formelle, en PPC, un modèle est défini à partir d'un triplet (X, D, C) (Régin, 2004), où $X = \{x_1, x_2, \dots, x_n\}$ est l'ensemble des variables du problème ; $D = \{D_1, D_2, \dots, D_n\}$ est l'ensemble des domaines associés à chaque variable $x_i \in X$ (D_i est associé à la variable X_i) ; et $C = \{C_1, C_2, \dots, C_m\}$ est l'ensemble des contraintes. Une contrainte traduit une caractéristique du problème qui doit être satisfaite par un sous-ensemble de variables. Un solveur de programmation par contraintes calcule une solution en instanciant chacune des variables du modèle à une valeur satisfaisant simultanément toutes les contraintes.

La propagation de contraintes utilise un algorithme de filtrage pour chaque contrainte : celui-ci, en fonction de variables prises en compte par la contrainte et de leurs domaines respectifs, supprime les valeurs impossibles c'est-à-dire qui ne peuvent appartenir à aucune solution. Par exemple, soient deux variables avec leurs domaines respectifs $A \in [1\,;\,2\,;\,3]$ et $B \in [1\,;\,2\,;\,3]$ avec la contrainte $\mathcal{C} : A + B = 5$, alors les affectations $A = 1$ ou $B = 1$ sont impossibles (aucune solution n'est réalisable avec $A = 1$ ou $B = 1$), les domaines peuvent être réduit à $A \in [2\,;\,3]$ et $B \in [2\,;\,3]$.

La modification du domaine d'une variable x_i, par un algorithme de filtrage, peut avoir des répercussions sur les domaines des autres variables, il est donc utile de réétudier l'ensemble des variables ayant au moins une contrainte en commun avec la variable x_i. Ce mécanisme est appelé propagation. L'ensemble de l'espace de recherche est parcouru en affectant successivement toutes les valeurs possibles à toutes les variables. Cette exploration est efficace, car les algorithmes de filtrage et de propagation sont sollicités après chaque affectation.

L'algorithme de parcours de l'espace des solutions le plus courant est l'algorithme parcours en profondeur d'un arbre et est basé sur des techniques de *backtracking* comme le

rappelle (Malapert, 2010). Il débute avec l'ensemble des variables non affectées, et itérativement, une variable est sélectionnée et une valeur lui est affectée. L'algorithme décrit alors un arbre de recherche où chaque branche correspond à l'affectation d'une valeur à une variable, et chaque nœud à une affectation partielle (par abus de langage, une solution partielle). L'algorithme vérifie à chaque nœud que l'affectation partielle est valide et respecte toutes les contraintes. Si c'est le cas (le nœud représente une solution), alors l'algorithme filtre le domaine de la variable puis il propage les contraintes. Ensuite, l'algorithme sélectionne une nouvelle variable et une nouvelle valeur pour obtenir un nouveau nœud. Sinon, l'algorithme « backtrack » en remontant dans l'arbre, c'est-à-dire que la dernière affectation réalisée est remise en cause, et l'algorithme choisit une nouvelle valeur à la variable.

L'algorithme de parcours de l'arbre de recherche peut arriver sur deux types de feuilles : soit toutes les variables sont instanciées et toutes les contraintes sont respectées, alors la feuille de l'arbre est sur une solution ; soit une (ou plusieurs) variable(s) a (ont) un domaine vide (après filtrage et propagation), et aucune solution n'est possible avec les valeurs déjà affectées aux autres variables.

La Figure 5 propose un parcours en profondeur du problème composé de trois variables avec leurs domaines respectifs : $A = [1\,;\,2]$, $B = [1\,;\,2\,;\,3]$ et $C = [1\,;\,2\,;\,3]$, et d'une unique contrainte $\mathcal{C} : A \neq B \neq C$.

À chaque nœud de l'arbre, la solution partielle est constituée des variables précédées d'un astérisque. Dans le nœud 1, les domaines de chaque variable sont les données du problème. Les arcs sortants du nœud 1 sont des branchements sur la variable A (sans perte de généralité, l'ordre des branchements est arbitraire), dans le premier cas $A = 1$, les domaines des autres variables sont inférés pour donner le nœud 2 avec une nouvelle solution partielle. Par exemple, le domaine de la variable B est réduit : la valeur 1 est supprimée de ce domaine. Ensuite l'algorithme de parcours en profondeur branche à partir du nœud 2 sur le sommet B ($B = 2$) et arrive à une feuille de l'arbre car toutes les variables sont instanciées (grâce à la propagation des contraintes sur le domaine de la variable C) et respectent la contrainte $\mathcal{C}$. L'algorithme de parcours backtrack sur le nœud 2 où il fait un nouveau branchement ($B = 3$) pour obtenir sur le nœud 4 qui est une feuille. Ensuite, il backtrack de nouveau sur le nœud 2. Puisque toutes les possibilités du nœud 2 ont été explorées, l'algorithme backtrack au nœud 1 et effectue un nouveau branchement pour la variable A avec $A = 2$.

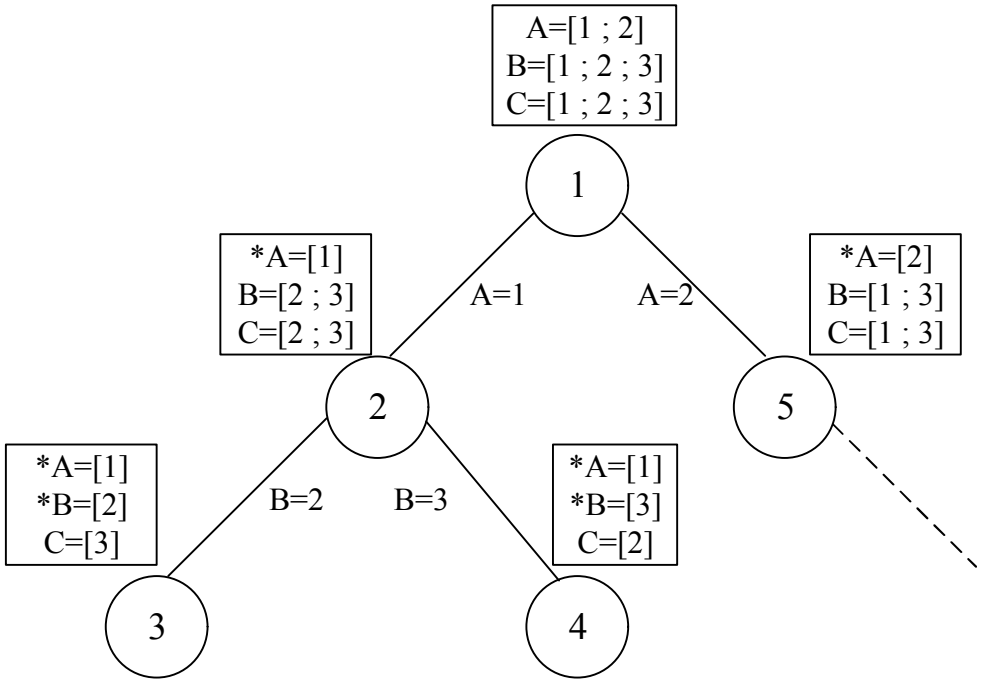


Figure 5 Parcours en profondeur d'un arbre de recherche

Les algorithmes de filtrage étant un des fondements de la PPC, il est important que ceux-ci soient efficaces et puissants. La communauté scientifique de la programmation par contraintes s'est penchée sur ces algorithmes de filtrage et la littérature compte maintenant un grand ensemble de contraintes permettant un filtrage ayant les deux caractéristiques précédemment citées. Ces contraintes sont nommées contraintes globales et utilisent généralement de nombreuses approches issues de la RO. Parmi les nombreuses contraintes globales, les plus courantes sont : la contrainte « AllDifferent » qui impose à un ensemble de variables d'avoir toute une valeur différente (Laurière, 1978; Régin, 1994) ; la contrainte « Cumulative » qui permet de résoudre des problèmes d'ordonnancement avec affectation de ressources (Aggoun and Beldiceanu, 1993) ; la contrainte « Cycle » (ou « Circuit ») qui permet de résoudre des problèmes de type voyageur de commerce (Beldiceanu and Contejean, 1994). Le site web ("Global Constraint Catalog," 2014) (http://sofdem.github.io/gccat/gccat/sec5.html) liste plus de 420 contraintes globales.

5.2.4. Programmation dynamique

La programmation dynamique fait partie des méthodes de résolution exactes. Une des applications des plus connues est l'algorithme des plus courts chemins de (Bellman, 1954). La programmation dynamique repose sur le principe d'optimalité énoncé par Bellman : « toute politique optimale est composée de sous-politiques optimales ». La programmation dynamique est basée sur quatre idées : résoudre des sous-problèmes, stocker leurs solutions, conserver les solutions pertinentes et les utiliser pour construire la solution du problème initial.

Le sous-problème de départ est le plus petit possible et sa solution permet de résoudre un sous-problème de taille (ou niveau) supérieure. Les solutions de sous-problèmes peuvent être combinées pour résoudre le problème de taille supérieure. La solution d'un sous-problème peut être utilisée par plusieurs sous problèmes de niveaux supérieurs.

La Figure 6 illustre la résolution d'un problème, représenté par un rond orange, par programmation dynamique. Les sous-problèmes sont en vert, et les flèches indiquent les liens entre les résolutions des sous-problèmes.

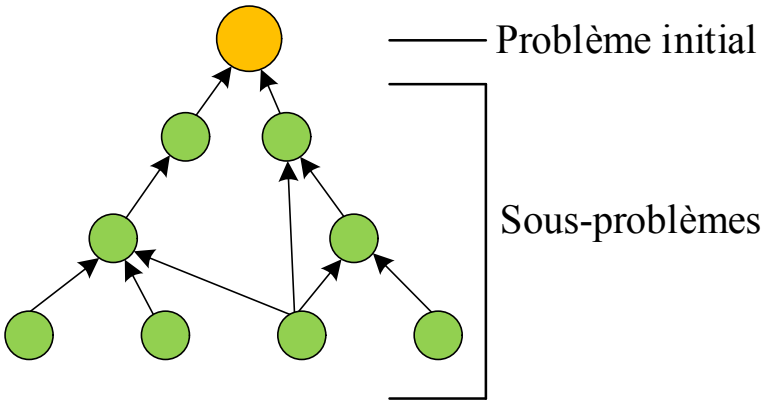


Figure 6 Résolution d'un problème par programmation dynamique

Un problème d'optimisation peut être résolu par programmation dynamique si la solution du problème est composée de solutions partielles des sous-problèmes. Un des algorithmes de programmation dynamique parmi les plus connus est l'algorithme de Bellman-Ford (Ford and Lester, 1956; Bellman, 1958; Moore, 1959) pour le calcul d'un chemin de longueur optimale dans un graphe. Cet algorithme est notamment étendu par (Feillet et al., 2004) pour la résolution du Problème du Plus Court Chemin Élémentaire avec Contraintes de Ressources (Elementary Shortest Path Problem with Resource Constraints – ESPPRC).

5.3. Résolution par méthodes approchées : les métaheuristiques

Les heuristiques et les métaheuristiques sont des méthodes approchées qui ont pour objectif d'obtenir des solutions valides, de préférence de bonne qualité, en un temps raisonnable. Les métaheuristiques sont des schémas génériques de résolution indépendants du problème traité ; à la différence des heuristiques qui sont généralement dédiées à la résolution d'un problème spécifique.

La grande difficulté d'une métaheuristique (et d'une heuristique) est : d'être à la fois capable de diversifier les solutions pour explorer tout l'espace des solutions afin d'éviter de « tomber » dans un optimum local ; et à la fois capable d'intensifier les recherches de solutions dans des zones de l'espace de recherche qui semblent prometteuses de bonnes solutions. Les métaheuristiques et les heuristiques doivent trouver le bon compromis entre ces deux dynamiques de recherche.

(Boussaïd et al., 2013) classent les métaheuristiques en deux catégories : les métaheuristiques basées sur une seule solution et les métaheuristiques basées sur une population. Les métaheuristiques basées sur une seule solution, aussi appelées méthodes à trajectoires, débutent avec une solution et se déplacent dans l'espace des solutions à partir de cette unique solution qui est sans cesse modifiée. Elles englobent, entre autres, le Recuit-Simulé (Simulated Annealing), la recherche taboue (Tabu Search), le GRASP (Greedy Randomized Adaptive Search Procedure), l'ILS (Iterated Local Search) et leurs extensions. Les métaheuristiques basées sur une population travaillent avec un ensemble de solutions appelé population. Itérativement, des individus sont sélectionnés de la population, croisés,

mutés, puis ajoutés dans la population. Ces méthodes sont généralement basées sur le résultat d'observations de phénomènes naturels chez les êtres vivants. Les méthodes les plus populaires sont les algorithmes génétiques et mémétiques, ainsi que les algorithmes de type colonie de fourmis (Ant Colony) et l'essaim de particules (Particle Swarm).

Le nombre de métaheuristiques et d'heuristiques étant immense, il est difficilement imaginable de toutes les présenter. Les sections suivantes présentent quelques métaheuristiques issues des deux communautés, et qui sont très fréquemment utilisées pour résoudre des problèmes d'ordonnancement et/ou de tournées de véhicules. Un soin particulier est donné au GRASP×ELS, car celui-ci est utilisé au cours des recherches présentées dans ce manuscrit.

5.3.1. GRASP

La méthode GRASP (Greedy Randomized Adaptive Search Procedure) a été introduite par (Feo and Resende, 1989, 1995; Resende and Ribeiro, 2010). Cette métaheuristique est basée sur la répétition de deux étapes qui cherchent à combiner les avantages des heuristiques constructives et des méthodes de voisinage. La première étape du GRASP consiste à générer une solution à l'aide d'une heuristique de construction, cette heuristique contient une part d'aléatoire et généralement essaie de fournir une solution de bonne qualité. La seconde étape du GRASP est l'amélioration de la solution par un processus de recherche locale. L'algorithme itère jusqu'à ce que le critère d'arrêt soit rencontré, par exemple un nombre maximal d'itérations.

Cette métaheuristique, facile à implémenter et à mettre en œuvre, est capable d'obtenir de bons résultats, par exemple pour le Job-shop Scheduling Problem (Binato et al., 2002) ou le CARP avec des fenêtres de temps (Capacitated Arc Routing Problem with Time Window) (Reghioui et al., 2007).

5.3.2. ILS

La métaheuristique ILS (Iterated Local Search) a été introduite par (Baum, 1986) et a été notamment reprise pour le problème du TSP par (Lourenço et al., 2003) et (Chen et al., 2010).

L'ILS est une métaheuristique basée sur les voisinages. À partir d'une solution initiale, une nouvelle solution est générée par une mutation, ce qui a pour objectif d'extirper la solution des optimums locaux. Ensuite une recherche locale est appliquée sur la nouvelle solution, la meilleure solution entre la nouvelle solution et la solution initiale est le nouveau point de départ de la métaheuristique. Le paramétrage de la mutation est un point clé de cette métaheuristique, car une perturbation trop faible de la solution ne permet pas à celle-ci de sortir de l'optimum local ; tandis qu'une perturbation trop forte risque de générer des solutions très différentes de la solution initiale, empêchant de chercher des solutions proches de la solution initiale.

5.3.3. ELS

(Wolf and Merz, 2007) sont les premiers à proposer la méthode ELS (Evolutionary Local Search) qui est une extension de l'ILS : à partir d'une solution initiale, plusieurs solutions sont générées par mutation puis subissent une recherche locale. La meilleure des solutions est conservée et devient la nouvelle solution initiale. La différence entre l'ILS et l'ELS est que dans l'ILS, une seule nouvelle solution est générée à chaque itération, alors que dans l'ELS plusieurs solutions sont générées à partir de la même solution initiale à chaque itération.

(Vidal et al., 2014) utilise, entre autres, un ELS pour des problèmes de multi-depot vehicle routing problems et de multi-depot vehicle fleet mix problem.

5.3.4. GRASP×ELS

Le GRASP×ELS est une hybridation de deux métaheuristiques : le GRASP et l'ELS, dont la paternité est régulièrement attribuée à (Prins, 2009) pour le Vehicle Routing Problem (VRP). Cette métaheuristique a été utilisée pour de nombreux problèmes de tournées de véhicules : (Duhamel et al., 2012) l'emploie pour le Heterogeneous VRP ; et d'ordonnancement : (Chassaing et al., 2014) l'utilise pour résoudre un problème de Job-shop avec un paradigme de web service, (Kemmoé-Tchomté et al., 2017) propose une extension d'un ELS avec plusieurs niveaux pour un problème de Job-shop Flexible.

La force du GRASP×ELS se situe dans la concaténation du GRASP comme méthode de diversification et de l'ELS comme méthode d'intensification. La diversification permet d'explorer tout l'espace des solutions tandis que l'intensification permet de converger localement vers une solution de bonne qualité.

L'Algorithme 1 est l'algorithme de principe du GRASP×ELS. Une première solution est obtenue par le GRASP (ligne 7). À partir de cette solution l'ELS est exécuté *maxELS* fois (ligne 9). À chaque itération de l'ELS, *maxVoisins* sont générés par mutation de la solution S (ligne 12), ces voisins sont notés S''. Le meilleur voisin S'' issu des *maxVoisins* (pas forcément meilleur que la solution S) devient le nouveau point de départ de l'ELS suivant (ligne 21). Pour chaque nouvelle solution, une recherche locale est appliquée (ligne 8 et ligne 13).

Algorithme 1 : Principe du GRASP×ELS

Sortie
S^* : meilleure solution trouvée par le GRASP×ELS
Variables
S, S', S'' : solutions intermédiaires
Début
| **Tant Que** critère d'arrêt non satisfait **faire**
| | S = solution obtenue par le GRASP
| | S = solution obtenue par une recherche locale sur S
| | **Pour** $iterEls = 1, \dots, maxELS$ **Faire**
| | | $S''.cout = \infty$
| | | **Pour** $iterVoisinage = 1, \dots, maxVoisins$ **Faire**
| | | | S' = solution obtenue par mutation de S
| | | | S' = solution obtenue par une recherche locale sur S'
| | | | **Si** $S'.cout < S''.cout$ **Alors**
| | | | | $S'' = S'$
| | | | | **Si** $S'.cout < S^*.cout$ **Alors**
| | | | | | $S^* = S'$
| | | | | **FinSi**
| | | | **FinSi**
| | | **FinPour**
| | | $S = S''$
| | **FinPour**
| **FinTantQue**
| **Retourner** $S*$
Fin

5.3.5. Algorithmes génétiques et mémétiques

L'algorithme mémétique est introduit par (Moscato, 1989) et il s'agit d'une extension de l'algorithme génétique (Holland, 1970) qui est lui-même un algorithme évolutionnaire, c'est-à-dire dont le principe s'inspire de la théorie de l'évolution. La puissance de ces algorithmes font d'eux des métaheuristiques très utilisées pour résoudre une multitude de problèmes combinatoires (Moscato et al., 2004).

Les algorithmes génétiques reprennent le principe de la théorie de l'évolution de l'espèce. De manière très générale, chaque organisme vivant est composé de chromosomes qui comportent des gènes. Ce sont ces derniers qui sont transmis de génération en génération lors de la reproduction des individus. Il y a trois étapes dans la transmission : le croisement (aussi appelé crossover), la mutation et la sélection. Les algorithmes génétiques sont initialisés avec une population comportant plusieurs individus et celle-ci va subir les trois étapes de la transmission pendant plusieurs générations.

Le crossover est le croisement de deux individus pour former un ou plusieurs fils. Il existe de nombreux moyens de réaliser un croisement, le cas le plus fréquent étant d'échanger entre les deux individus une partie de leurs gènes. La mutation est la variation d'un ou plusieurs gènes. La mutation est régulièrement aléatoire, mais elle peut aussi être guidée suivant les problèmes traités. Lors de la sélection, une partie de la population est conservée afin de maintenir la taille de la population constante et cette population constitue la

génération suivante qui va subir les trois mêmes étapes. Il existe plusieurs grands courants pour la sélection : la sélection élitiste, où les meilleurs individus sont conservés ; la sélection aléatoire par roulette, où généralement, la probabilité d'un individu à être pris est proportionnelle à sa qualité ; la sélection par tournois, où deux ou trois éléments sont choisis et seulement le meilleur est préservé.

Un algorithme mémétique comporte le même schéma algorithmique qu'un algorithme génétique. Toutefois l'opérateur de mutation est remplacé par une recherche locale qui est appliquée sur chaque individu issu d'un croisement (Moscato et al., 2004).

Pour les algorithmes génétiques, la convergence est assurée par le croisement et la diversification par la mutation, tandis que pour les algorithmes mémétiques, la convergence est assurée par la recherche locale et la diversification par le croisement.

6. Modélisations des solutions

Les solutions d'un problème d'ordonnancement ou de tournées de véhicules sont représentées suivant deux approches (Cheng et al., 1996) : les approches par représentation directe, et les approches par représentation indirecte. Ces deux approches influencent directement les méthodes de résolution et leurs comportements.

6.1. Espace de Codage-Espace des solutions

Les méthodes de résolution utilisant des représentations directes travaillent sur l'espace des solutions, ces approches tirent profit des caractéristiques et des propriétés du problème et des solutions. Les méthodes utilisant des représentations indirectes œuvrent sur l'espace de codage et non plus sur l'espace des solutions. Cette seconde approche nécessite donc une fonction de décodage pour passer de l'espace de codage à l'espace des solutions et éventuellement une fonction d'encodage pour passer de l'espace des solutions à l'espace de codage. Une approche utilisant une représentation indirecte alterne l'utilisation des deux espaces afin d'en tirer profit pour obtenir un schéma algorithmique performant. L'utilisation d'un espace de codage permet généralement de muter les solutions tout en les conservant réalisables, tandis que l'espace des solutions est souvent utilisé par les recherches locales pour améliorer la solution courante. Les fonctions de décodage peuvent être de trois types (Cheng et al., 1996) (elles sont illustrées sur la Figure 7) :

- *1-to-1 mapping* : un élément de l'espace de codage est associé à un unique élément dans l'espace des solutions. Ce type de codage est le plus recherché, car il permet de passer d'un espace à l'autre sans équivoque ;
- *1-to-n mapping* : un élément de l'espace de codage est associé à plusieurs éléments dans l'espace des solutions, ce type de codage est souvent à éviter ;
- *n-to-1 mapping* : plusieurs éléments de l'espace de codage correspondent à un même élément dans l'espace des solutions.

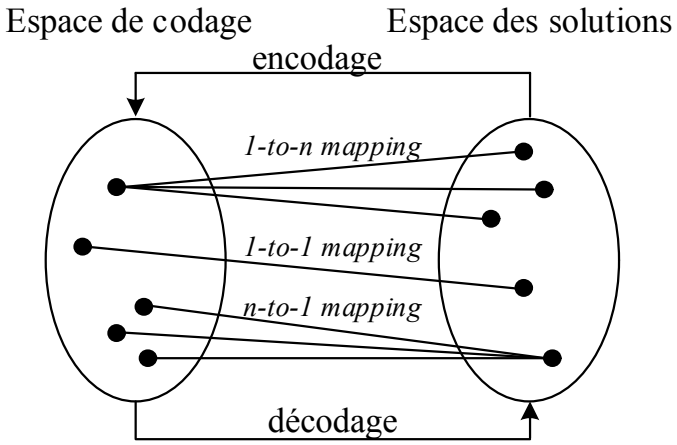


Figure 7 Relations entre l'espace de codage et l'espace des solutions, (Cheng et al., 1996)

Généralement, les méthodes promues dans la littérature comprennent les quatre points clés suivant :

- une représentation des solutions par une représentation indirecte ;
- une fonction de décodage avec une faible complexité permettant de passer d'un espace à l'autre ;
- la définition d'une recherche locale qui explore alternativement l'espace de codage et l'espace des solutions ;
- une métaheuristique favorisant l'exploration de l'espace de codage et évitant de rester bloqué dans des optimums locaux.

6.2. Classes d'ordonnancements

La représentation d'une solution ainsi que la définition d'une fonction de décodage sont intimement liées au problème traité. En effet, lors de la résolution de problèmes d'ordonnancement, avec notamment comme critère la minimisation du makespan, il existe de nombreux ordonnancements identiques possibles qui ont le même makespan : les jobs ont le même ordre de passage sur les machines, mais ils diffèrent uniquement par les dates de début des opérations. Par exemple, en cas de décalage temporel d'opérations (soit avancement, soit retardement de la date de début d'une opération), cela n'entraine pas obligatoirement un impact sur le critère de minimisation. L'intérêt des classes d'ordonnancement est de limiter la recherche de solutions à un sous-ensemble d'ordonnancements qui sont prouvés meilleurs que les ordonnancements non explorés, et ainsi donc de limiter la taille des espaces de codages et de solutions.

(Baker, 1974) présente en détail les différentes classes d'ordonnancement. Cependant, seules deux classes sont présentées dans ce manuscrit : les ordonnancements semi-actifs et les ordonnancements actifs. Ces classes font partie des classes les plus répandues dans la littérature et ont un impact direct sur les recherches présentées ici.

Les classes sont introduites et illustrées par un exemple de Job-shop Scheduling Problem (JSP). Ce JSP est composé de deux pièces (ou jobs), elles-mêmes comportant trois opérations dans leurs gammes opératoires respectives. Trois machines sont disponibles et chacune des opérations doit être réalisée par une machine spécifique. L'objectif du problème est de minimiser le makespan, c'est-à-dire, la date de fin de la dernière opération. Les

gammes opératoires doivent être respectées et sont données par le Tableau 1. La première ligne concerne la pièce J1, l'opération O1 de cette pièce doit être réalisée sur la machine M1 et a une durée de 3 unités de temps.

Tableau 1 Instance P du Job-shop Scheduling Problem

Job 1	O1 = (M1 ; 3)	O2 = (M3 ; 2)	O3 = (M2 ; 3)
Job 2	O1 = (M2 ; 5)	O2 = (M3 ; 3)	O3 = (M1 ; 6)

Définition : ordonnancement semi-actif

Un ordonnancement est dit semi-actif s'il n'est pas possible de commencer une opération plus tôt sans violer une contrainte, sans modifier l'ordre des opérations sur les machines et sans détériorer le critère à minimiser.

La définition d'ordonnancement semi-actif est étendue à la classe d'ordonnancement actif en supprimant la contrainte d'interdiction de modifier l'ordre des opérations sur les machines.

Définition : ordonnancement actif

Un ordonnancement est dit actif s'il n'est pas possible d'avancer l'exécution d'une opération sans retarder la date de début d'une autre opération ou sans violer une contrainte des gammes opératoires.

La Figure 8 est la représentation, par un diagramme de Gantt, d'une solution optimale du problème du *P* (du Tableau 1), cette solution n'est ni semi-active ni active. Les opérations J1-O1, J1-O2 et J1-O3, toutes appartenant au job J1 peuvent être avancées, sans modifier l'ordre des opérations sur chaque machine pour donner un ordonnancement semi-actif (Figure 9).

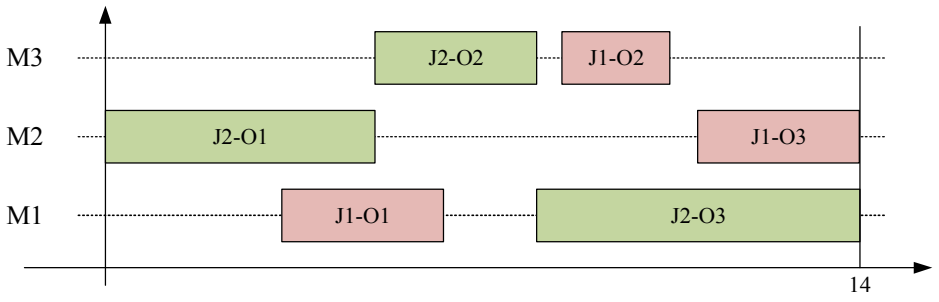


Figure 8 Solution optimale, ni semi-active, ni active, de P

Sur la Figure 9, il est impossible d'avancer la date de début d'une opération sans violer une contrainte de gamme ou changer l'ordre de passage des opérations sur les machines, c'est un ordonnancement semi-actif.

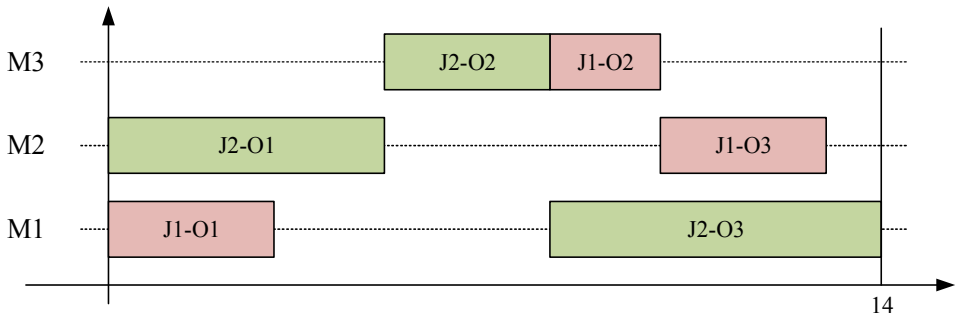


Figure 9 Solution optimale et semi-active de P

La Figure 10 présente une solution active du problème *P*. L'ordre des opérations J1-O2 et J2-O2 est inversé par rapport à l'ordonnancement semi-actif (Figure 9), et toutes les opérations commencent au plus tôt. Cette solution est également optimale.

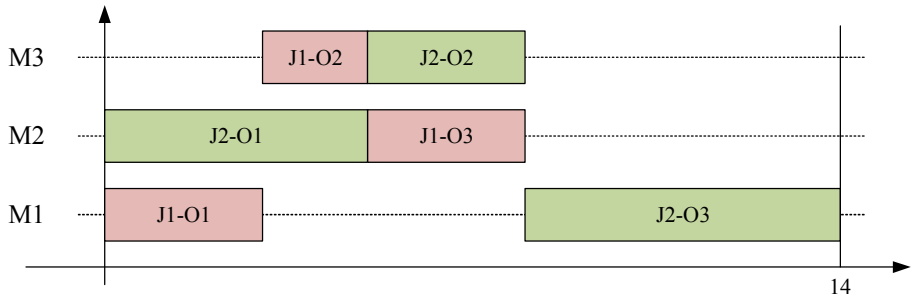


Figure 10 Solution optimale et active de P

La recherche d'ordonnancements semi-actifs est régulièrement privilégiée, car la classe des ordonnancements actifs est contenue dans la classe des ordonnancements semi-actifs, et l'inverse n'est pas vrai. Il est rarement requis un ordonnancement actif, et généralement un ordonnancement semi-actif suffit pour obtenir une solution qui minimise (ou maximise) le critère, c'est le cas notamment pour le makespan. De plus, il existe un plus grand nombre d'ordonnancements semi-actifs que d'ordonnancements actifs, et l'exploration de l'espace des solutions des ordonnancements semi-actifs est beaucoup plus simple. D'autre part, pour la majorité des critères, il existe toujours un ordonnancement optimal qui est semi-actif (Baker, 1974).

Les ordonnancements semi-actifs sont également appelés « au plus tôt » ou « left-shifted », car les dates de début des opérations sont les plus tôt possibles : une opération s'exécute dès que la machine qui lui est associée est libre et que l'opération précédente dans sa gamme a fini.

Les paragraphes suivants présentent les représentations et les fonctions de codage les plus utilisées dans les problèmes d'ordonnancement et de tournées de véhicules.

6.3. Représentation et codage des problèmes d'ordonnancement

L'objectif du Job-shop Scheduling Problem (JSP) est d'ordonnancer la fabrication de pièces (aussi appelées jobs) sur des machines.

6.3.1. Représentation par répétition pour le Job-shop Sheduling Problem

Le Job-shop Scheduling Problem est communément modélisé par le graphe disjonctif introduit par (Roy and Sussmann, 1964) et noté $G = (V, E)$, où V est l'ensemble des nœuds représentant les opérations (cet ensemble contient également deux nœuds factices 0 et $*$, définissant, respectivement, le nœud initial et le nœud final du graphe) ; et où E est l'ensemble (i) des arcs conjonctifs correspondant aux gammes de chaque job, et *(ii)* des arcs disjonctifs définissant les contraintes de partages de ressource (deux opérations nécessitant la même machine doivent être ordonnées). Chaque arc, d'une opération $O_{i,j}$ à une opération $O_{i,j+1}$ a pour poids le processing time $pt_{i,j}$ de $O_{i,j}$. La Figure 11 présente ce graphe avec les données du JSP issu du Tableau 1.

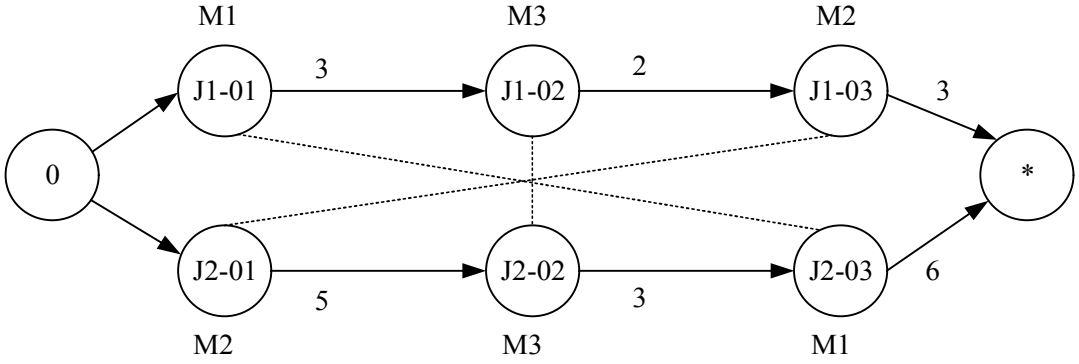


Figure 11 Graphe disjonctif du Job-shop

Un graphe disjonctif peut être représenté dans l'espace de codage par un vecteur topologique, qui permet à la fois d'arbitrer les disjonctions et, grâce à une fonction de décodage basée sur un algorithme de plus long chemin de type Bellman-Ford (présentée dans la section 6.3.2), d'évaluer les dates de début des opérations. Les disjonctions doivent être arbitrées sans créer de cycle car la présence d'un cycle induit une solution irréalisable.

Une première représentation possible d'une solution est un vecteur par ordre total, c'est-à-dire que l'ordre des opérations est explicitement donné. Le Tableau 2 présente un ordre total du problème du Job-shop (Tableau 1). Chaque opération est présente une seule fois et le vecteur est décodé en le lisant de gauche à droite. Chaque opération est ordonnancée au plus tôt : quand l'opération précédente dans la gamme est finie et quand la machine est libre.

Tableau 2 Représentation par ordre total du problème de Job-shop

Ordre total :	J1-O1	J2-O1	J2-O2	J1-O2	J2-O3	J1-O3

La représentation par ordre total peut conduire à la création de vecteurs irréalisables, par exemple si l'opération J2-O2 est placée avant l'opération J2-O1 ou en créant des cycles. Cet inconvénient peut être supprimé en utilisant un vecteur par répétition qui a un niveau d'abstraction supérieur.

(Bierwirth, 1995) propose une représentation par répétition (aussi appelé vecteur de Bierwirth) pour le problème du Job-shop qui fait partie des plus utilisées dans la littérature et qui peut être étendu à d'autres problèmes d'ordonnancement. Ce vecteur est une suite

ordonnée de numéros de jobs, et non plus du numéro des opérations. Chaque job est répété autant de fois qu'il comporte d'opérations dans sa gamme.

Le Tableau 3 présente un vecteur par répétition du problème *P* (Tableau 1). Chaque numéro de job est présent trois fois dans le vecteur, car chacun des jobs contient trois opérations dans sa gamme. La seconde partie du Tableau 3 est la correspondance entre la répétition du job dans le vecteur par répétition et l'opération du job correspondant. Lors de la lecture du vecteur, la $i^{ème}$ répétition du job *j* correspond à l'opération située en position *i* dans la gamme du job *j*. Par exemple, le premier « 1 » correspond donc au job J1, et comme il est le premier, il correspond à l'opération J1-O1 ; le second « 2 » (en troisième position dans le vecteur) correspond à la seconde opération du job J2, soit J2-O2. L'abstraction des numéros des opérations, remplacées par les numéros des jobs, dans le vecteur de Bierwirth permet de ne générer que des vecteurs réalisables.

Tableau 3 Représentation par répétition du problème P

Vecteur par répétition :	1	2	2	1	2	1

Ordre total correspondant	J1-O1	J2-O1	J2-O2	J1-O2	J2-O3	J1-O3

6.3.2. Fonction de décodage avec la représentation par répétition

Un vecteur de Bierwirth est un ordre topologique qui permet d'arbitrer les disjonctions machines. Il est généralement utilisé avec une fonction de codage qui donne lieu à une évaluation au plus tôt des dates de début des opérations, la fonction d'évaluation est basée sur un algorithme de plus long chemin dans un graphe, tel que les algorithmes de Bellman-Ford (Ford and Lester, 1956; Bellman, 1958; Moore, 1959) ou Dijkstra (Dijkstra, 1959). Avec un vecteur topologique, la complexité de la fonction d'évaluation est linéaire.

Le principe de cette fonction de codage est d'ordonnancer au plus tôt chaque opération, dans l'ordre de lecture du vecteur. Pour être ordonnancée au plus tôt, une opération doit : avoir l'opération précédente dans sa gamme de terminée, et la machine de disponible (qui n'exécute pas une autre opération). Par exemple, d'après le vecteur du Tableau 3, la première opération à être ordonnancée est la première opération du job J1, donc J1-O1. Cette opération est ordonnancée à la date 0, car la machine M1 est libre. La seconde opération à être ordonnancée est J2-O1 sur la machine M2 à la date 0 ; la troisième opération est J2-O2 à la date 5 qui est la date de fin de l'opération J2-O1. L'opération J1-O2 peut commencer dès que l'opération J1-O1 est finie (à la date 3) et que l'opération J2-O2 est terminée, car J1-O2 et J2-O2 partagent la même machine. La Figure 12 illustre la solution obtenue par l'évaluation du vecteur par répétition du Tableau 3. Les dates de début des opérations (encadrées sur la Figure 12) sont au plus tôt et correspondent à une solution semi-active. La solution obtenue par cette fonction de décodage appartient à la classe des ordonnancements semi-actifs.

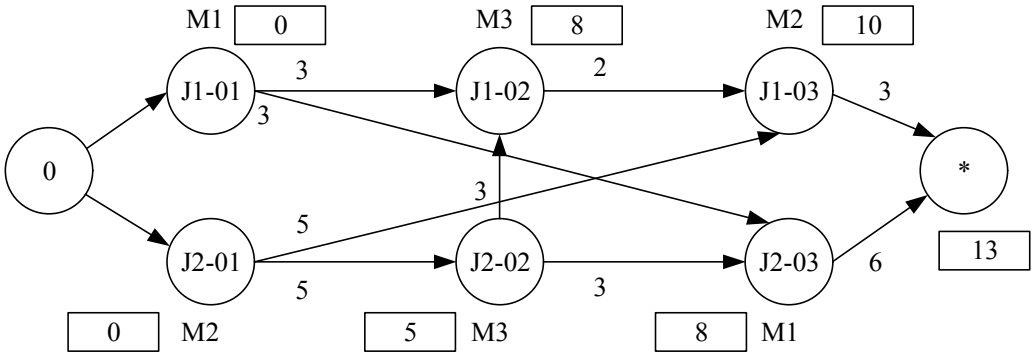


Figure 12 Solution obtenue à partir du vecteur par répétition

Par ailleurs, il est intéressant de noter que tout vecteur par répétition permet d'obtenir une solution grâce à un algorithme de type Bellman-Ford. Dans ce cas, la représentation des solutions sous forme de vecteur par répétition est dite « non surreprésentée » (Caumond, 2006). Une représentation est dite « surreprésentée » lorsque l'algorithme de décodage peut échouer à fournir une solution à partir de cette solution, ce qui est le cas avec les vecteurs par ordre total à cause de la création potentielle de cycles dans le graphe.

Plusieurs vecteurs par répétition évalués par une fonction de décodage de type Bellman-Ford peuvent mener à la même solution. Dans ce cas, la représentation par vecteurs par répétition est dite « multireprésentée » (Caumond, 2006), la fonction de décodage est de type *n-to-one*. La représentation par vecteurs par ordre total est également multireprésentée.

Pour les problèmes d'ordonnancement, il est difficile de trouver des représentations qui ne soient ni surreprésentées ni multireprésentées. Par ailleurs, il est avantageux de travailler avec des solutions semi-actives, car cet ensemble contient une solution optimale (pour le critère du makespan) et permet de réduire les espaces des solutions et de codage. C'est pourquoi le vecteur de Bierwirth et la fonction de codage basée sur l'algorithme de Bellman-Ford sont les fondements de nombreuses méthodes de résolution des problèmes d'ordonnancement, incluant, mais non limité au Job-shop Scheduling Problem.

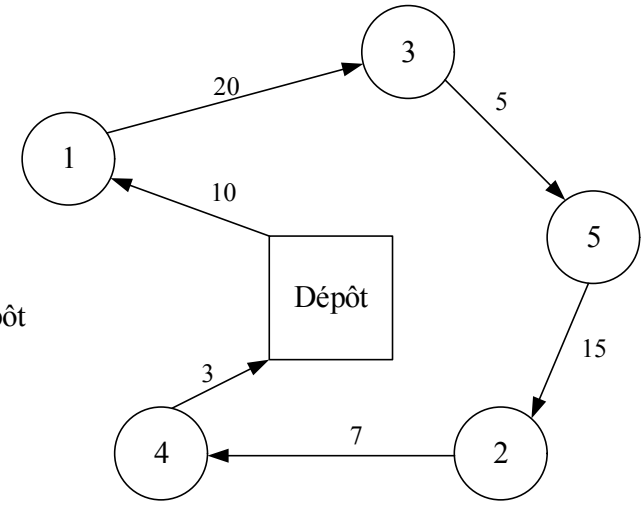
6.4. Représentation et codage des problèmes de tournées de véhicules

6.4.1. Représentation par ordre total du TSP

Les représentations les plus courantes pour les problèmes de tournées de véhicules sont les représentations par ordre total, qui sont des vecteurs de permutation des clients. Les premières représentations ont été introduites par (Beasley, 1983) et (Ulusoy, 1985). Par exemple, pour le TSP, un ordre total est donné par la Figure 13, il est traditionnellement appelé « tour géant ». Le voyageur part donc du dépôt, puis visite le sommet 1, puis le 3, etc., avant de revenir au dépôt. Le graphe correspondant à ce chemin est donné sur la Figure 13. Chaque arc est pondéré par la distance à parcourir entre les deux sommets.

S'il est supposé que le temps de transport entre deux sommets est égal à la distance les séparant, alors pour connaître la date de visite de chaque client, il suffit de parcourir le tour géant et pour chaque sommet la date de début est égale à la date de début de la visite précédente plus la durée de transport. Soit une tournée $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_n]$ avec n le nombre

de sommets du problème, et λ_i un sommet du graphe. La date de début St_v du sommet v en position λ_i dans le vecteur est donnée par $St_v = St_{\lambda_i - 1} + T_{\lambda_i - 1, \lambda_i}$ avec $\lambda_i - 1$ la visite précédente dans le vecteur et $T_{i,j}$ le temps de transport entre les visites i et j. Un tel algorithme définit donc une fonction de décodage du tour géant avec une complexité en $O(n)$ (où n est le nombre de sommets dans le graphe).



Ordre total (ou tour géant) :
Dépôt – 1 – 3 – 5 – 2 – 4 – dépôt

Figure 13 Ordre total et représentation de la solution d'un TSP

Le CVRP (ou par abus de langage, également appelé VRP) est une extension où une flotte de véhicules doit visiter l'ensemble des clients, sachant que chaque client requiert une quantité de produits et que chaque véhicule à une capacité maximale. Un véhicule visitant un client doit livrer toute la quantité requise par celui-ci, et un véhicule ne peut pas transporter plus de produits que sa capacité maximale de charge.

La procédure SPLIT introduite par (Beasley, 1983) est une fonction de décodage qui permet de passer d'un tour géant à une solution du CVRP. Le tour géant permet de relâcher la contrainte de capacité des véhicules, et la fonction de décodage s'assure que cette contrainte est respectée dans la solution finale. La Figure 14 introduit un exemple de CVRP pour deux véhicules de capacité 10. La tournée du véhicule V1 est représentée en vert, tandis que la tournée du véhicule V2 est en violet. À chaque client est associé une quantité de produits requise (en rouge sur la figure), par exemple, le client 1 requiert 5 unités de produits.

Le principe de la procédure SPLIT est de respecter le tour géant et lorsqu'un véhicule atteint sa capacité maximale, il rentre au dépôt. Un second véhicule continue alors la tournée suivant l'ordre du tour géant. La procédure SPLIT permet de « découper » le tour géant en plusieurs tournées respectant les capacités des véhicules. Sur la Figure 14, le tour géant est 1 – 3 – 5 – 2 – 4. Le véhicule V1 commence au dépôt, puis visite le client 1, à ce moment-là, sa charge est de 5. Ensuite, d'après le tour géant, le véhicule visite le client 3 et sa charge devient 8 (5 pour le client 1 plus 3 pour le client 3). Le véhicule V1 visite le client suivant dans le tour géant, le client 5. Or la charge du véhicule V1 est dans ce cas 12 (5+3+4), ce qui est plus que sa capacité maximale (10). Donc le véhicule V1 ne peut pas visiter le client 5, et rentre au dépôt. Le véhicule V2 commence alors sa tournée en partant du dépôt et visite le client 5, puis visite le client 2 et enfin le client 4. Sa capacité maximale est toujours respectée.

La procédure SPLIT incluse dans des heuristiques ou métaheuristiques permet d'obtenir des schémas globaux d'optimisation particulièrement efficaces (Lacomme et al., 2001).

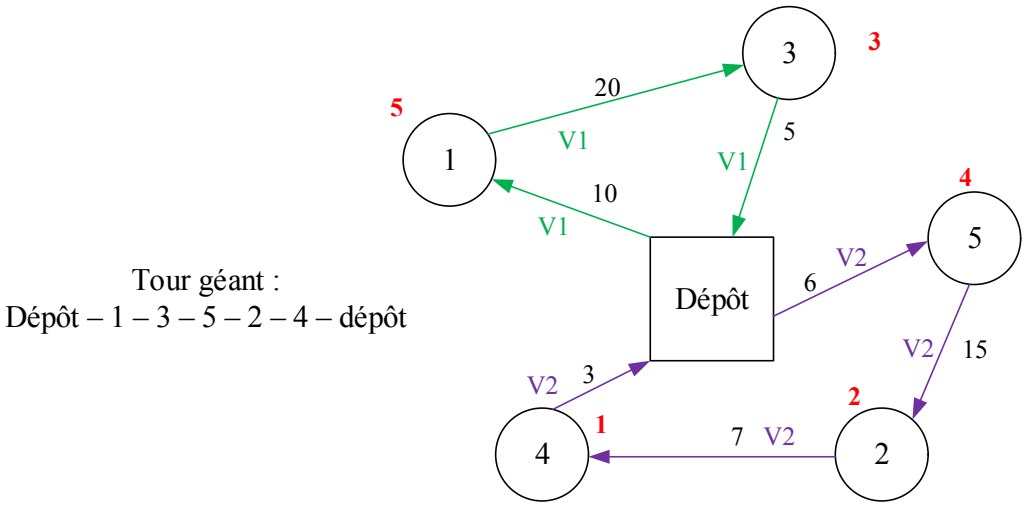


Figure 14 Exemple de solution du CVRP à partir d'un tour géant

De nombreux problèmes de tournées de véhicules sont encodés sous forme de tours géants qui sont décodés par des procédures basées sur le SPLIT de (Beasley, 1983). Ces procédures sont très efficaces et permettent la minimisation de critères tels que la distance parcourue ou le temps de trajet. L'utilisation d'un espace de codage permet de relâcher des contraintes qui sont ensuite insérées par la fonction de décodage, c'est le cas de la contrainte sur la capacité pour le CVRP, ou la contrainte sur les gammes opératoires pour le Job-shop.

Toutefois, il existe des problèmes de tournées de véhicules pour lesquels la solution recherchée ne minimise pas (ou pas uniquement) un critère de distance ou temps de trajet. Pour nombre de ces problèmes, il existe peu de représentations des solutions et des fonctions de décodage disponibles dans la littérature telles qu'elles sont décrites par (Cheng et al., 1996).

C'est le cas pour le DARP pour lequel il n'existe pas de représentation équivalente à un tour géant qui serait « découpé » pour obtenir un ensemble de tournées. De plus, pour le DARP, il y a une difficulté particulière liée à l'évaluation même des tournées. Le DARP ne définit pas un critère à minimiser spécifique pour une tournée, mais nécessite le calcul d'une solution de compromis entre trois critères. Les tournées sont évaluées pour fournir une solution non nécessairement semi-active en fonction de la solution recherchée. Le DARP est un problème de transport de clients (personnes ou biens) entre deux points, et une solution est un compromis entre un critère de distance et des critères liés à la durée totale de la tournée et au temps passé par les clients dans les véhicules. Une solution du DARP est généralement représentée par une succession de tournées respectant toutes les contraintes telle que la capacité, et la fonction de décodage se résume en une fonction d'évaluation des tournées pour uniquement déterminer les dates de début. Par abus de langage, cette fonction d'évaluation est appelée fonction de codage.

La partie haute de la Figure 15 présente les espaces de codage et des solutions du Job-Shop et du CVRP ainsi que les fonctions de décodage. La partie basse de la figure illustre la représentation d'une solution du DARP ainsi que sa fonction d'évaluation pour obtenir une solution. La fonction d'évaluation du DARP la plus courante est celle proposée par (Cordeau and Laporte, 2003) qui est présentée dans la prochaine section.

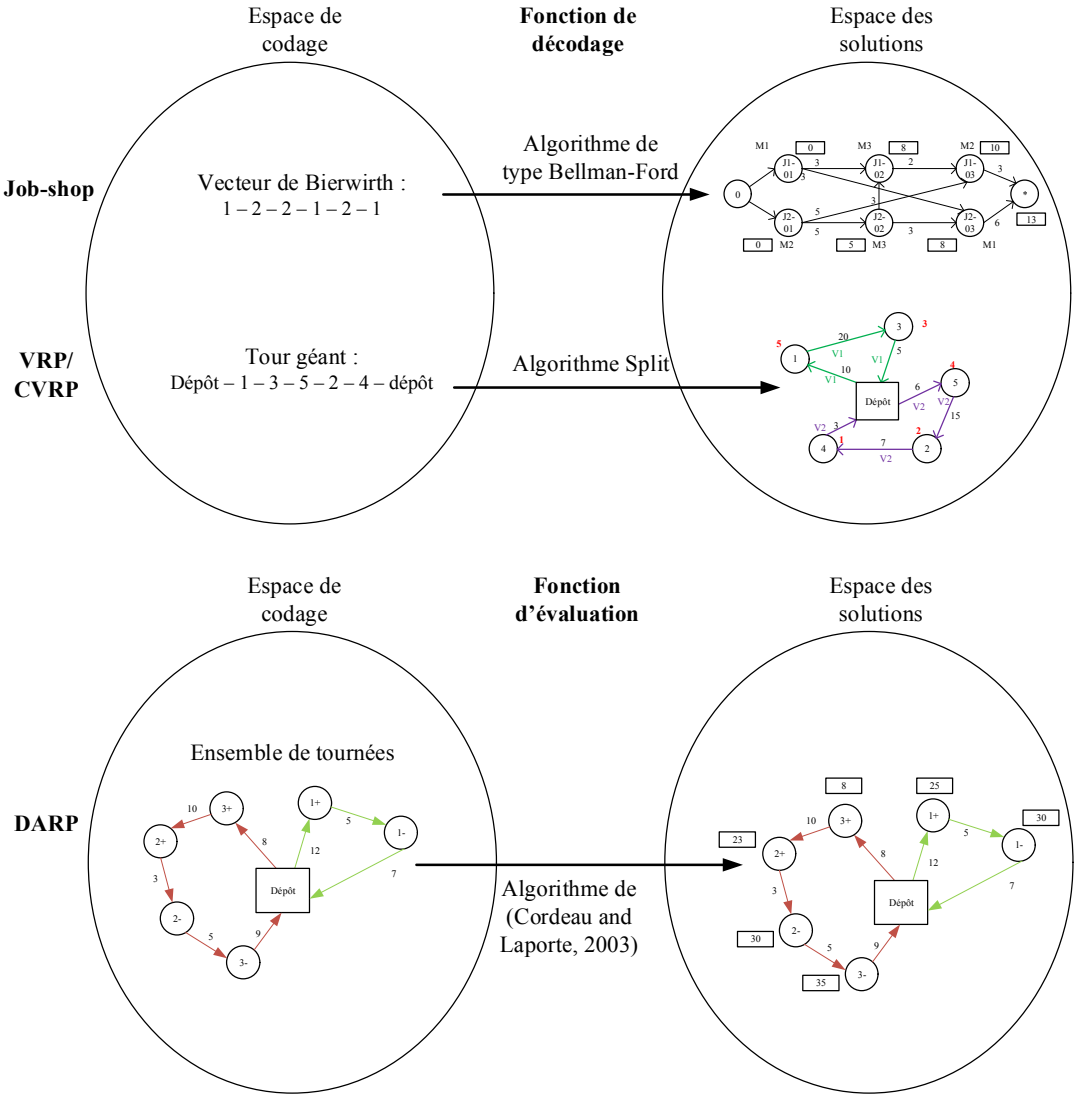


Figure 15 Codage du Job-shop, du CVRP et du DARP

6.4.2. Fonctions d'évaluation pour le DARP

Pour les problèmes de transport à la demande (Dial-A-Ride Problem – DARP), plusieurs fonctions d'évaluation d'une tournée sont proposées dans la littérature, et certaines de ces fonctions ne donnent pas des solutions semi-actives, où les visites ont lieu au plus tôt, mais des solutions qui maximisent des critères de qualité de service. Ces fonctions d'évaluation sont parfois également appelées fonctions de décodage par extension de la définition des représentations indirectes donnée par (Cheng et al., 1996).

La fonction d'évaluation, la plus connue est celle proposée par (Cordeau and Laporte, 2003), et ayant une grande influence sur les travaux de ce manuscrit, elle est présentée dans cette section. La fonction d'évaluation de (Cordeau and Laporte, 2003) est de complexité quadratique. Plus récemment, (Firat and Woeginger, 2011) ont introduit une fonction d'évaluation avec les mêmes critères à minimiser de complexité linéaire, mais celle-ci n'est pas utilisée dans ce manuscrit, car elle est moins courante dans la littérature. Il faut, toutefois, noter que la fonction de (Cordeau and Laporte, 2003) et celle de (Firat and Woeginger, 2011)

ne donnent pas la même solution finale car elles fournissent des solutions de compromis différentes.

(Stein, 1978) introduit le DARP (Dial-A-Ride Problem), mais la définition la plus courante est celle proposée par (Cordeau and Laporte, 2003). Un DARP est constitué de n clients à transporter et de m véhicules. À chaque client est associée une requête de transport entre deux sommets : le sommet d'origine et le sommet de destination du client. Les véhicules ont une capacité (nombre de clients transportés simultanément) limitée qui doit être respectée en tout temps au cours de leurs tournées. Un client est donc caractérisé par : deux sommets, celui de départ (pickup) et celui d'arrivée (delivery) ; un temps de service ; une fenêtre de temps dans laquelle le véhicule doit commencer son service ; et un temps de trajet maximal. Il est évident que les tournées doivent respecter l'ordre : « sommet pickup – sommet delivery » pour chaque client. De plus, les véhicules ont une durée maximale à respecter pour réaliser leur tournée.

Une solution du DARP est un ensemble de tournées, où chaque tournée est associée à un véhicule, et dont les dates d'arrivée chez un client, dates de début de service, dates de fin de service et dates de départ de chez le client sont connues.

Soit une tournée T_k, les notations suivantes sont introduites par (Cordeau and Laporte, 2003) puis reprises par (Chassaing et al., 2016), et sont illustrées sur la Figure 16 :

- Les données :
  - $[E_i ; L_i]$: la fenêtre de temps de la visite i ;
  - d_i : durée du service chez le client i ;
  - $i+$: sommet de départ du client i ;
  - $i-$: sommet d'arrivée du client i.
- Informations concernant la tournée k :
  - s_i^k : la position de i dans la tournée T_k ;
  - TD_{T_k} : la durée totale de la tournée ;
  - v^k : véhicule associé à la tournée.
- Les variables de décisions :
  - A_i : date d'arrivée du véhicule sur le sommet i ;
  - B_i : date de début de service sur le sommet i ;
  - C_i : date de fin de service sur le sommet i ;
  - D_i : date de départ du véhicule du sommet i ;
  - W_i^+ : attente du véhicule sur le sommet i avant le début du service, $W_i^+ = B_i - A_i$;
  - W_i^-: attente du véhicule sur le sommet i après la fin du service, $W_i^- = D_i - C_i$;
  - RT_i : temps de trajet du client i, $RT_i = B_{i-} - B_{i+}$.

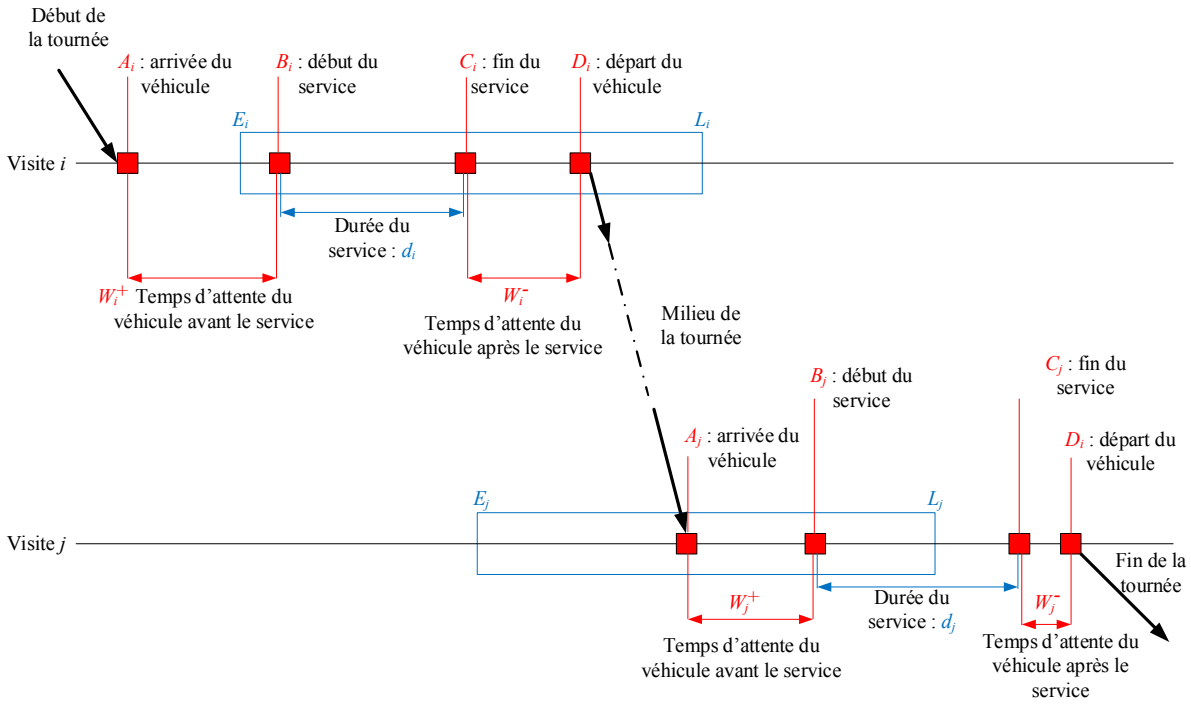


Figure 16 Notations pour le DARP

Les solutions du DARP sont représentées indirectement par un ensemble de tours géants où chaque tour est associé à un véhicule. La Figure 17 présente les tournées d'un DARP avec trois clients et deux véhicules. Le sommet de type « z+ » correspond au point de départ du client z et le sommet « z- » correspond au sommet destination de z. La tournée $T1$ est associée au véhicule $V1$, et la tournée $T2$ au véhicule $V2$. La tournée $T1$ débute au dépôt, puis collecte le client 3 (sommet 3+), ensuite, le véhicule $V1$ collecte le client 2, puis le dépose
(sommet 2-), ensuite il dépose le client 3, et enfin $V1$ retourne au dépôt.

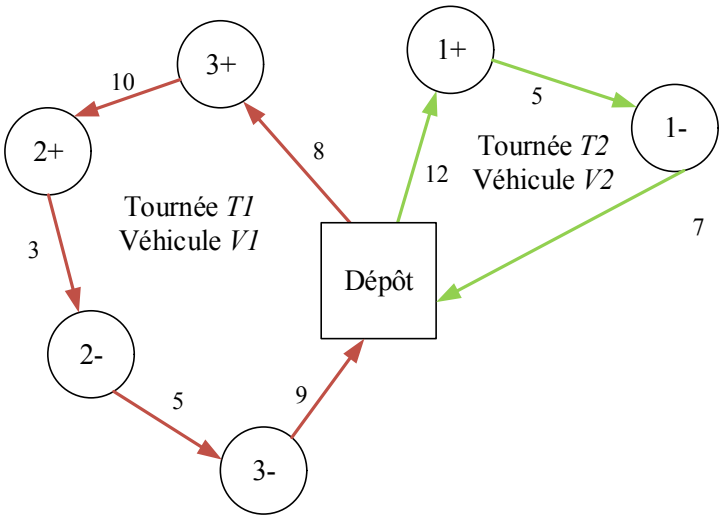


Figure 17 Tournées d'un DARP

Le Tableau 4 est la représentation de la solution du DARP de la Figure 17. Chaque tournée est directement donnée.

Tableau 4 Représentation d'une solution du DARP

Tournée $T1$ (véhicule $V1$)	3+	2+	2-	3-
Tournée $T2$ (véhicule $V2$)	1+	1-		

Les différentes fonctions d'évaluation d'une tournée, et notamment celle de (Cordeau and Laporte, 2003), sont présentées à partir de la tournée $T2$ de la Figure 17. La tournée $T2$ est illustrée par la Figure 18 où la fenêtre de temps de chaque sommet est représentée par un rectangle proportionnel à sa taille qui est situé en dessous des nœuds sur un axe temporel. Les arcs sont pondérés par la distance entre les deux sommets reliés. Les temps de service sur les clients sont considérés nuls dans cet exemple. Pour les différentes évaluations, les trois critères définis par (Cordeau and Laporte, 2003) sont calculés : la somme des temps d'attente, la durée du trajet des clients et la durée totale de la tournée. La date de début de chaque visite doit donc se trouver dans les rectangles représentant les fenêtres de temps sur la Figure 18.

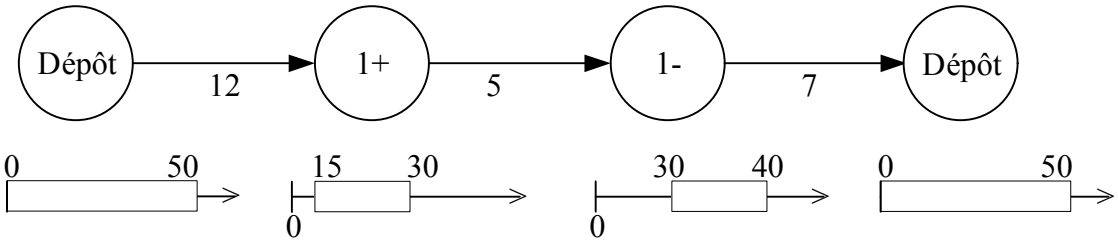
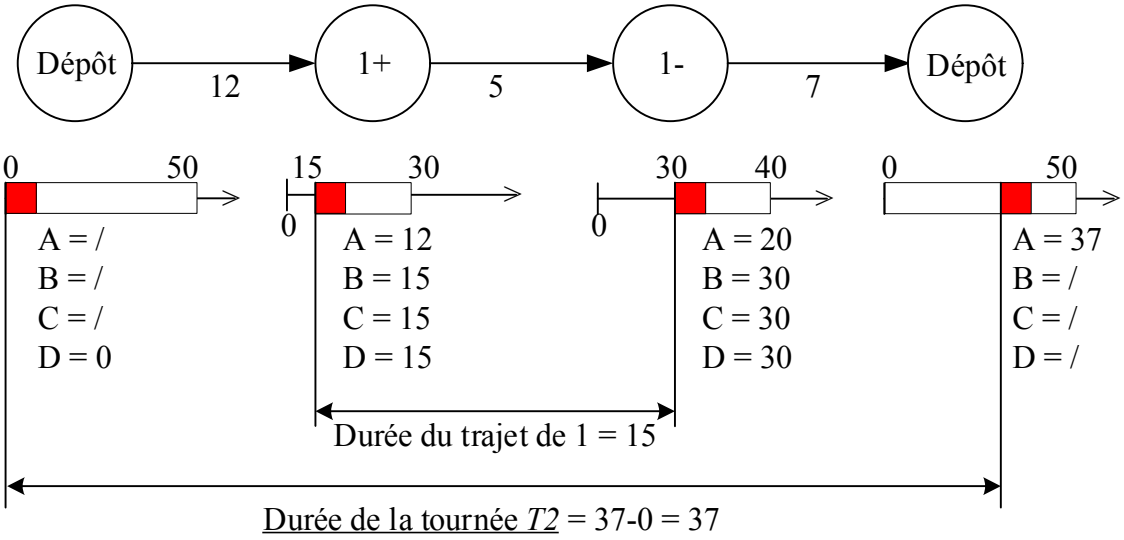


Figure 18 Tournée $T2$

Une fonction d'évaluation « au plus tôt » permet d'obtenir les dates d'arrivée, de début, de fin et de départ au plus tôt sur chaque sommet : dès que le service est fini, le véhicule quitte la visite et dès que le véhicule arrive, le service débute (sous condition que la fenêtre de temps soit respectée). Sur la Figure 19, les fenêtres de temps des sommets sont représentées par un rectangle sur un axe temporel. Le carré rouge dans chaque rectangle représente la date de début de service (la variable B). Le véhicule part le plus tôt possible du dépôt, soit la date $D = 0$, il arrive sur le sommet 1+ à la date $A = 12$, mais doit attendre la date 15 d'ouverture de la fenêtre de temps pour commencer son service ($B = 15$), le service étant de durée nulle, la date de fin est $C = 15$. Le véhicule part à la date 15 ($D = 15$), arrive au temps 20 sur le sommet suivant, etc. Les dates de départ, d'arrivée, de début de service, de fin de service sont données par la Figure 19. La durée du trajet du client 1 est de 15, cette durée est définie par la date de début de service du client au point de livraison moins la date de départ du client au point de chargement de celui-ci, ici la durée du trajet du client 1 est donc : $30 - 15 = 15$. La durée de la tournée est la différence entre la date d'arrivée au dépôt final et la date de départ du dépôt initial, dans cet exemple la durée de la tournée est $37 - 0 = 37$, et le temps d'attente du véhicule sur les différents sommets est de 13.

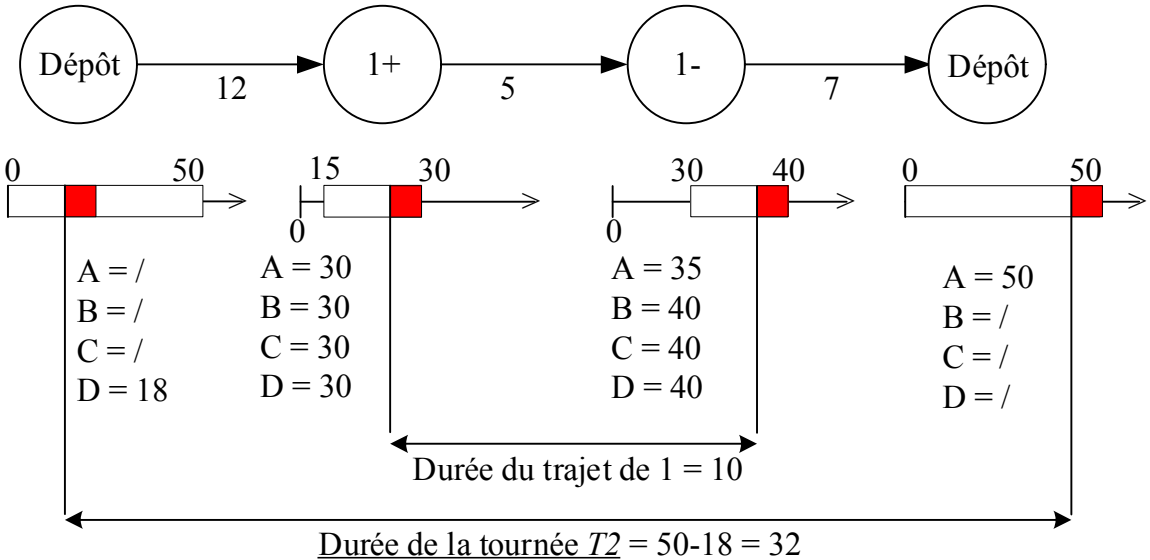


Durée de la tournée *T2* = 37-0 = 37

Somme des attentes = (15-12)+(15-15)+(30-20)+(30-30) = 13

Figure 19 Évaluation au plus tôt de la tournée $T2$

La Figure 20 propose les dates d'arrivée, de début de service, de fin de service et de départ au plus tard. Le véhicule ne part pas de la date 0 du dépôt, mais à une date ultérieure qui est la date maximale permettant d'effectuer la tournée en respectant les fenêtres de temps. La date de départ du dépôt est 18. De manière générale, chaque date est la date maximale possible afin que la suite de la tournée soit toujours réalisable. La durée de trajet de 1 est maintenant de 10, la durée de la tournée est de 32, et le temps d'attente est de 5. Sur le sommet 1-, le véhicule arrive à la date $A = 35$ et pourrait commencer son service, mais il attend la date la plus tard possible et débute son service à la date $B = 40$.



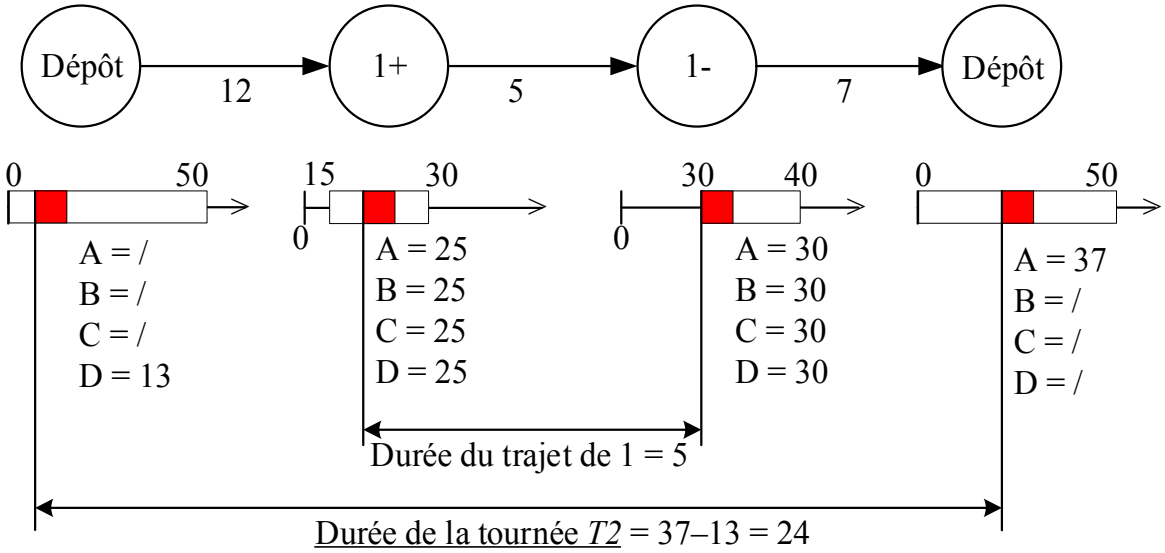
Durée de la tournée *T2* = 50-18 = 32

Somme des attentes = (30-30)+(30-30)+(40-35)+(40-40) = 5

Figure 20 Évaluation au plus tard de la tournée $T2$

La Figure 21 présente les dates d'arrivée, de début de service, de fin de service et de départ d'après l'évaluation de (Cordeau and Laporte, 2003). Cette fonction d'évaluation est présentée en détail dans le chapitre 2. Au cours de cette évaluation, les durées du trajet de 1,

de la tournée *T*1 et du temps d'attente sont minimisées, ces dates sont respectivement 5, 24 et 0.



Somme des attentes = (25-25)+(25-25)+(30-30)+(30-30) = 0

Figure 21 Évaluation de (Cordeau and Laporte, 2003) de la tournée T2

Le Tableau 5 présente les valeurs des trois critères (durée du trajet du client 1, durée de la tournée et somme des attentes) pour la tournée *T*2. L'évaluation au plus tôt a les trois critères les plus élevés, respectivement, de valeur 15, 37 et 13. L'évaluation au plus tard a des critères de valeurs inférieures à l'évaluation au plus tôt, mais ceci résulte du hasard des données de l'instance, et des cas inverses peuvent être facilement trouvés. L'évaluation de (Cordeau and Laporte, 2003) propose une solution avec les critères minimisés.

*Tableau 5 Valeurs des critères en fonction de la fonction d'évaluation de T*2

Fonction d'évaluation	**Durée du trajet du client 1**	**Durée de la tournée**	**Somme des attentes**
Évaluation au plus tôt	15	37	13
Évaluation au plus tard	10	32	5
Évaluation de (Cordeau and Laporte, 2003)	5	24	0

7. Conclusion

Ce chapitre introduit les systèmes de production avec transport puis les principales familles de problèmes d'ordonnancement et de tournées de véhicules sont présentées, avec des exemples pratiques et théoriques.

Les méthodes de résolution les plus courantes pour les problèmes d'ordonnancement et de tournées de véhicules sont exposées, avec d'un côté des méthodes exactes : programmation linéaire, génération de colonnes, programmation par contraintes, programmation dynamique ; et de l'autre côté des méthodes approchées : métaheuristiques. Ces différentes approches sont complémentaires et permettent de faire face à des problèmes de différentes tailles tout en conservant des temps de résolution acceptables d'un point de vue

utilisateur. Les méthodes de résolution sont indissociables de la complexité et des classes de problèmes, ces notions sont donc rappelées dans ce chapitre.

Les liens entre les métaheuristiques et les méthodes de codage des problèmes sont particulièrement mis en évidence sur deux problèmes : le Job-shop Scheduling Problem (un problème d'ordonnancement) et le Dial-A-Ride Problem (un problème de tournées de véhicules). Cette mise en lumière a pour but d'expliciter l'importance et les enjeux des représentations des problèmes. Les méthodes utilisées pour ces deux problèmes sont les bases de codage de nombreux problèmes d'ordonnancement et de tournées de véhicules.

La suite de ce manuscrit s'articule autour de trois problèmes théoriques qui permettent de modéliser les problématiques des systèmes de production avec transport : le Job-shop Scheduling Problem with Routing, puis le Workforce Scheduling and Routing Problem et enfin le Generalised Workforce Scheduling and Routing Problem.

8. Bibliographie

Aggoun, A., Beldiceanu, N., 1993. Extending chip in order to solve complex scheduling and placement problems. Mathematical and Computer Modelling 17, 57–73. https://doi.org/10.1016/0895-7177(93)90068-A

Ahmadizar, F., Rabanimotlagh, A., 2014. Group shop scheduling with uncertain data and a general cost objective. The International Journal of Advanced Manufacturing Technology 70, 1313–1322. https://doi.org/10.1007/s00170-013-5353-7

Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J., 2011. The traveling salesman problem: a computational study, Princeton university press. ed.

Arno Sprecher, 2000. Scheduling Resource-Constrained Projects Competitively at Modest Memory Requirements. Management Science 46, 710–723.

Artigues, C., 2017. On the strength of time-indexed formulations for the resource-constrained project scheduling problem. Operations Research Letters 45, 154–159. https://doi.org/10.1016/j.orl.2017.02.001

Artigues, C., Huguet, M.-J., Lopez, P., 2011. Generalized disjunctive constraint propagation for solving the job shop problem with time lags. Engineering Applications of Artificial Intelligence 24, 220–231. https://doi.org/10.1016/j.engappai.2010.07.008

Baker, K.R., 1974. Introduction to sequencing and scheduling. John Wiley & Sons.

Baptiste, P., Demassey, S., 2004. Tight LP bounds for resource constrained project scheduling. OR Spectrum 26, 251–262. https://doi.org/10.1007/s00291-003-0155-1

Baptiste, P., Le Pape, C., Nuijten, W., 2012. Constraint-based scheduling: applying constraint programming to scheduling problems, Springer Science & Business Media.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H., 1998. Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research 46, 316–329. https://doi.org/10.1287/opre.46.3.316

Baum, E.B., 1986. Iterated descent: A better algorithm for local search in combinatorial optimization problems. Manuscript.

Beasley, J., 1983. Route first—Cluster second methods for vehicle routing. Omega 11, 403–408. https://doi.org/10.1016/0305-0483(83)90033-6

Beldiceanu, N., Contejean, E., 1994. Introducing global constraints in CHIP. Mathematical and Computer Modelling 20, 97–123. https://doi.org/10.1016/0895-7177(94)90127-9

Bellman, R., 1958. On a routing problem. Quarterly of applied mathematics 16, 87–90.

Bellman, R., 1954. Some Applications of the Theory of Dynamic Programming-A Review. Journal of the Operations Research Society of America 2, 275–288.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. TOP 15, 1–31. https://doi.org/10.1007/s11750-007-0009-0

Bierwirth, C., 1995. A generalized permutation approach to job shop scheduling with genetic algorithms. Operations-Research-Spektrum 17, 87–92.

Binato, S., Binato, S., Loewenstern, D.M., Resende, M.G.C., 2002. A Grasp for Job Shop Scheduling. Essays and Surveys in Metaheuristics, Springer, Boston, MA 15, 59–79. https://doi.org/10.1007/978-1-4615-1507-4_3

Blazewicz, J., Domschke, W., Pesch, E., 1996. The job shop scheduling problem: Conventional and new solution techniques. European journal of operational research 93, 1–33.

Bourreau, É., Gondran, M., Lacomme, P., Vinot, M., 2020. Programmation Par Contraintes : démarches de modélisation pour des problèmes d'optimisation. Ellipses.

Bourreau, É., Gondran, M., Lacomme, P., Vinot, M., 2019. De la programmation linéaire à la programmation par contraintes. Ellipses.

Boussaïd, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Information Sciences 237, 82–117. https://doi.org/10.1016/j.ins.2013.02.041

Braekers, K., Caris, A., Janssens, G.K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transportation Research Part B: Methodological 67, 166–186. https://doi.org/10.1016/j.trb.2014.05.007

Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I., 2016. The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering 99, 300–313. https://doi.org/10.1016/j.cie.2015.12.007

Brah, S.A., Hunsucker, J.L., 1991. Branch and bound algorithm for the flow shop with multiple processors. European Journal of Operational Research 51, 88–99. https://doi.org/10.1016/0377-2217(91)90148-O

Brucker, P., Knust, S., 2000. A linear programming and constraint propagation-based lower bound for the RCPSP. European Journal of Operational Research 127, 355–362. https://doi.org/10.1016/S0377-2217(99)00489-0

Brucker, P., Schlie, R., 1990. Job-shop scheduling with multi-purpose machines. Computing 45, 369–375. https://doi.org/10.1007/BF02238804

Bruzzone, A.A.G., Anghinolfi, D., Paolucci, M., Tonelli, F., 2012. Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops. CIRP Annals 61, 459–462. https://doi.org/10.1016/j.cirp.2012.03.084

Çaliş, B., Bulkan, S., 2015. A research survey: review of AI solution strategies of job shop scheduling problem. Journal of Intelligent Manufacturing 26, 961–973.

Cambazard, H., Bourreau, É., 2004. Conception d'une contrainte globale de chemin, in: 10e Journées nationales sur la résolution pratique de problèmes NP-complets. Presented at the JNPC'04, pp. 107–121.

Carlier, J., Chrétienne, P., 1988. Problèmes d'ordonnancement : Modélisation, complexité, algorithmes, Masson. ed.

Carlier, J., Moukrim, A., Xu, H., 2009. The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm. Discrete Applied Mathematics 157, 3631–3642. https://doi.org/10.1016/j.dam.2009.02.012

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R., 2016. Workforce scheduling and routing problems: literature survey and computational study. Annals of Operations Research 239, 39–67.

Caumond, A., 2006. Le problème de jobshop avec contraintes : modélisation et optimisation (Thèse de doctorat). Université Blaise Pascal-Clermont-Ferrand II, Clermont-Ferrand.

Caumond, A., Lacomme, P., Tchernev, N., 2008. A memetic algorithm for the job-shop with time-lags. Computers & Operations Research 35, 2331–2356. https://doi.org/10.1016/j.cor.2006.11.007

Chassaing, M., 2015. Problèmes de transport à la demande avec prise en compte de la qualité de service (Thèse de doctorat). Université Blaise Pascal-Clermont-Ferrand II, Clermont-Ferrand.

Chassaing, M., Duhamel, C., Lacomme, P., 2016. An ELS-based approach with dynamic probabilities management in local search for the Dial-A-Ride Problem. Engineering Applications of Artificial Intelligence 48, 119–133.

Chassaing, M., Fontanel, J., Lacomme, P., Ren, L., Tchernev, N., Villechenon, P., 2014. A GRASP×ELS approach for the job-shop with a web service paradigm packaging. Expert Systems with Applications 41, 544–562. https://doi.org/10.1016/j.eswa.2013.07.080

Chaudhry, I.A., Khan, A.A., 2016. A research survey: review of flexible job shop scheduling techniques. International Transactions in Operational Research 23, 551–591. https://doi.org/10.1111/itor.12199

Chen, P., Huang, H., Dong, X.-Y., 2010. Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. Expert Systems with Applications 37, 1620–1627. https://doi.org/10.1016/j.eswa.2009.06.047

Chen, Z.-L., 2010. Integrated Production and Outbound Distribution Scheduling: Review and Extensions. Operations Research 58, 130–148. https://doi.org/10.1287/opre.1080.0688

Cheng, R., Gen, M., Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation. Computers & industrial engineering 30, 983–997.

Cheng, T.C.E., Sin, C.C.S., 1990. A state-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research 47, 271–292. https://doi.org/10.1016/0377-2217(90)90215-W

Cook, S.A., 1971. The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing - STOC '71. pp. 151–158. https://doi.org/10.1145/800157.805047

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Annals of Operations Research 153, 29–46. https://doi.org/10.1007/s10479-007-0170-8

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579–594. https://doi.org/10.1016/S0191-2615(02)00045-0

Dai, M., Tang, D., Giret, A., Salido, M.A., Li, W.D., 2013. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Robotics and Computer-Integrated Manufacturing 29, 418–429. https://doi.org/10.1016/j.rcim.2013.04.001

Dantzig, G.B., 2016. Linear programming and extensions, Princeton university press.

Dantzig, G.B., 1987. Origins of the Simplex Method. Stanford Univ CA Systems Optimization Lab.

Dantzig, G.B., Fulkerson, R., Johnson, S., 1954. Solution of a Large-Scale Traveling-Salesman Problem. Journal of the Operations Research Society of America 2, 393–410. https://doi.org/10.1287/opre.2.4.393

Dantzig, G.B., Ramser, J.H., 1959. The Truck Dispatching Problem. Management Science 6, 80–91.

Dantzig, G.B., Wolfe, P.M., 1960. Decomposition principle for linear programs. Operations research 8, 101–111.

Dell'Amico, M., 1996. Shop Problems with Two Machines and Time Lags. Operations Research 44, 777–787.

Dell'Amico, M., Trubian, M., 1993. Applying tabu search to the job-shop scheduling problem. Annals of Operations research 41, 231–252.

Demeulemeester, E., Herroelen, W., 1992. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management science 38, 1803–1818.

Desaulniers, G., Desrosiers, J., Solomon, M.M., 2005. Column generation, Springer Science & Business Media.

Desaulniers, G., Desrosiers, J., Solomon, M.M., 2001. Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems, in: Essays and Surveys in Metaheuristics. Springer US, Boston, MA, pp. 309–324. https://doi.org/10.1007/978-1-4615-1507-4_14

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische mathematik 1, 269–271.

Duhamel, C., Lacomme, P., Prodhon, C., 2012. A hybrid evolutionary local search with depth first search split procedure for the heterogeneous vehicle routing problems. Engineering Applications of Artificial Intelligence 25, 345–358. https://doi.org/10.1016/j.engappai.2011.10.002

Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal of Operational Research 54, 7–22. https://doi.org/10.1016/0377-2217(91)90319-Q

Eiselt, H.A., Gendreau, M., Laporte, G., 1995. Arc Routing Problems, Part I: The Chinese Postman Problem. Operations Research 43, 231–242.

Fanjul-Peyro, L., Ruiz, R., 2010. Iterated greedy local search methods for unrelated parallel machine scheduling. European Journal of Operational Research 207, 55–69. https://doi.org/10.1016/j.ejor.2010.03.030

Feillet, D., 2010. A tutorial on column generation and branch-and-price for vehicle routing problems. 4or 8, 407–424.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44, 216–229.

Feo, T.A., Resende, M.G., 1995. Greedy randomized adaptive search procedures. Journal of global optimization 6, 109–133.

Feo, T.A., Resende, M.G.C., 1989. A probabilistic heuristic for a computationally difficult set covering problem. Operations research letters 8, 67–71.

Firat, M., Woeginger, G.J., 2011. Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. Operations Research Letters 39, 32–35. https://doi.org/10.1016/j.orl.2010.11.004

Ford, J., Lester, R., 1956. Network flow theory. Rand Corp Santa Monica Ca.

Fu, L.-L., Aloulou, M.A., Artigues, C., 2018. Integrated production and outbound distribution scheduling problems with job release dates and deadlines. Journal of Scheduling 21, 443–460. https://doi.org/10.1007/s10951-017-0542-0

Fügenschuh, A., 2009. Solving a school bus scheduling problem with integer programming. European Journal of Operational Research 193, 867–884. https://doi.org/10.1016/j.ejor.2007.10.055

Gabel, T., Riedmiller, M., 2008. Adaptive reactive job-shop scheduling with reinforcement learning agents. International Journal of Information Technology and Intelligent Computing 24(4).

Ganesharajah, T., Hall, N.G., Sriskandarajah, C., 1998. Design and operational issues in AGV-served manufacturing systems. Annals of Operations Research 76, 109–154.

Garaix, T., Gondran, M., Lacomme, P., Mura, E., Tchernev, N., 2018. Workforce Scheduling Linear Programming Formulation. IFAC-PapersOnLine 51, 264–269. https://doi.org/10.1016/j.ifacol.2018.08.289

Garey, M.R., Johnson, D.S., 1979. Computers and intractability: A Guide to the Theory of NP-Completeness, San Francisco: WH Freeman. ed.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The Complexity of Flowshop and Jobshop Scheduling. Mathematics of Operations Research 1, 117–129.

Gendreau, M., Potvin, J.-Y., Bräumlaysy, O., Hasle, G., 2008. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography, in: The Vehicle Routing Problem: Latest Advances and New Challenges, Springer Science & Business Media. Boston, MA, pp. 143–169. https://doi.org/10.1007/978-0-387-77778-8

Gendreau, M., Tarantilis, C.D., 2010. Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Montreal, QC, Canada : Cirrelt.

Global Constraint Catalog [WWW Document], 2014. URL http://sofdem.github.io/gccat/gccat/sec5.html

Golden, B.L., Wong, R.T., 1981. Capacitated arc routing problems. Networks 11, 305–315.

Gondran, M., Huguet, M.-J., Lacomme, P., Quilliot, A., Tchernev, N., 2018. A Dial-a-Ride evaluation for solving the job-shop with routing considerations. Engineering Applications of Artificial Intelligence 74, 70–89. https://doi.org/10.1016/j.engappai.2018.05.010

Gonzalez, T., Sahni, S., 1976. Open Shop Scheduling to Minimize Finish Time. Journal of the ACM 23, 665–679. https://doi.org/10.1145/321978.321985

Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey, in: Annals of Discrete Mathematics. Elsevier, pp. 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X

Gu, J., Gu, X., Gu, M., 2009. A novel parallel quantum genetic algorithm for stochastic job shop scheduling. Journal of Mathematical Analysis and Applications 355, 63–81. https://doi.org/10.1016/j.jmaa.2008.12.065

Gutin, G., Punnen, A. (Eds.), 2006. The traveling salesman problem and its variations. Springer Science & Business Media 12.

Hartmann, S., Briskorn, D., 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207, 1–14. https://doi.org/10.1016/j.ejor.2009.11.005

Ho, S.C., Szeto, W.Y., Kuo, Y.-H., Leung, J.M.Y., Petering, M., Tou, T.W.H., 2018. A survey of dial-a-ride problems: Literature review and recent developments. Transportation Research Part B: Methodological 111, 395–421. https://doi.org/10.1016/j.trb.2018.02.001

Hojabri, H., Gendreau, M., Potvin, J.-Y., Rousseau, L.-M., 2018. Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. Computers & Operations Research 92, 87–97. https://doi.org/10.1016/j.cor.2017.11.011

Holland, J.H., 1970. Adaptation in Natural and Artificial Systems. M.I.T.P.

Horn, W.A., 1973. Minimizing Average Flow Time with Parallel Machines. Operations Research 21, 846–847.

Hunsucker, J.L., Shah, J.R., 1994. Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment. European Journal of Operational Research 72, 102–114.

Jain, A.S., Meeran, S., 1999. Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research 113, 390–434.

Karmarkar, N., 1984. A new polynomial-time algorithm for linear programming, in: The Sixteenth Annual ACM Symposium on Theory of Computing. Presented at the ACM, pp. 302–311.

Kemmoé, S., Lamy, D., Tchernev, N., 2015. An Optimization Approach for Job-shop with Financial Constraints - In the Context of Supply Chain Scheduling Considering Payment Delay between Members, in: Proceedings of the International Conference on Operations Research and Enterprise Systems. SCITEPRESS - Science and and Technology Publications, Lisbon, Portugal, pp. 190–198. https://doi.org/10.5220/0005271301900198

Kemmoé-Tchomté, S., Lamy, D., Tchernev, N., 2017. An effective multi-start multi-level evolutionary local search for the flexible job-shop problem. Engineering Applications of Artificial Intelligence 62, 80–95. https://doi.org/10.1016/j.engappai.2017.04.002

Kolisch, R., 1996. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research 90, 320–333. https://doi.org/10.1016/0377-2217(95)00357-6

Kolisch, R., Hartmann, S., 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174, 23–37. https://doi.org/10.1016/j.ejor.2005.01.065

Lacomme, P., Gondran, M., Vinot, M., 2017. At the crossroad of scheduling problems and routing problems, in: 18th Free Workshop on Metaheuristics of a Better World (EU/ME). Rome, Italy.

Lacomme, P., Prins, C., Ramdane-Chérif, W., 2001. A Genetic Algorithm for the Capacitated Arc Routing Problem and Its Extensions. Applications of Evolutionary Computing 2037, 473–483. https://doi.org/10.1007/3-540-45365-2_49

Lahyani, R., Khemakhem, M., Semet, F., 2015. Rich vehicle routing problems: From a taxonomy to a definition. European Journal of Operational Research 241, 1–14. https://doi.org/10.1016/j.ejor.2014.07.048

Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S., 1999. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review 13, 129–170.

Laurière, J.L., 1978. A Language and a Program for Stating and Solving Combinatorial Problems Artificial Intelligence. Artificial intelligence 10, 29–127.

Lawrence, S., 1984. Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. Carnegie-Mellon University, School of Industrial Administration, Pittsburgh.

Lee, T.-S., Loong, Y.-T., 2019. A review of scheduling problem and resolution methods in flexible flow shop. International Journal of Industrial Engineering Computations 10, 67–88. https://doi.org/10.5267/j.ijiec.2018.4.001

Leung, J.Y. (Ed.), 2004. Handbook of scheduling: algorithms, models, and performance analysis, CRC press. ed.

Liaw, C.-F., 2000. A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research 124, 28–42. https://doi.org/10.1016/S0377-2217(99)00168-X

Linn, R., Zhang, W., 1999. Hybrid flow shop scheduling: A survey. Computers & Industrial Engineering 37, 57–61. https://doi.org/10.1016/S0360-8352(99)00023-6

Lourenço, Helena.R., Martin, Olivier.C., Stutzle, Thomas., 2003. Iterated Local Search, in: Handbook of Metaheuristics. Springer, Boston, MA, pp. 320–353.

Malapert, A., 2010. Techniques d'ordonnancement d'atelier et de fournées basées sur la programmation par contraintes (Thèse de doctorat). Université de Nantes, faculté des sciences et des techniques, l'École Nationale Supérieure des Techniques Industrielles et des Mines de Nantes.

Manne, A.S., 1960. On the Job-Shop Scheduling Problem. Operations research 8, 219–223.

Marandi, F., Fatemi Ghomi, S.M.T., 2019. Integrated multi-factory production and distribution scheduling applying vehicle routing approach. International Journal of Production Research 57, 722–748. https://doi.org/10.1080/00207543.2018.1481301

Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L., 1998. An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation. Management Science 44, 714–729. https://doi.org/10.1287/mnsc.44.5.714

Moons, S., Ramaekers, K., Caris, A., Arda, Y., 2017. Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. Computers & Industrial Engineering 104, 224–245. https://doi.org/10.1016/j.cie.2016.12.010

Moore, E.F., 1959. The shortest path through a maze, in: Proc. Int. Symp. Switching Theory 1959. pp. 285–292.

Moscato, P., 1989. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts Towards Memetic Algorithms. Caltech concurrent computation program, C3P Report 826.

Moscato, P., Cotta, C., Mendes, A., 2004. Memetic algorithms. New optimization techniques in engineering, Springer, Berlin, Heidelberg 53–85.

Munier-Kordon, A., Rebaine, D., 2010. The two-machine open-shop problem with unit-time operations and time delays to minimize the makespan. European Journal of Operational Research 203, 42–49.

Nowicki, E., Smutnicki, C., 1996. A fast taboo seach algorithm for the job shop problem. Management Science 42, 797–814.

Parragh, S.N., Doerner, K.F., Hartl, R.F., 2008. A survey on pickup and delivery models Part II: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft 58, 81–117.

Pinedo, M.L., 2012. Scheduling: Theory, Algorithms, and Systems. Springer Science & Business Media, Berlin.

Pradenas, L., Oportus, B., Parada, V., 2013. Mitigation of greenhouse gas emissions in vehicle routing problems with backhauling. Expert Systems with Applications 40, 2985–2991. https://doi.org/10.1016/j.eswa.2012.12.014

Prins, C., 2009. A GRASP x evolutionary local search hybrid for the vehicle routing problem. Bio-inspired algorithms for the vehicle routing problem, Springer, Berlin, Heidelberg 35–53.

Prins, C., 2004. A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. Computers & Operations Research 31, 1985–2002.

Pritsker, A.A.B., Watters, L.J., Wolfe, P.M., 1969. Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. Management Science 16, 93–108.

Pureza, V., Morabito, R., Reimann, M., 2012. Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the VRPTW. European Journal of Operational Research 218, 636–647. https://doi.org/10.1016/j.ejor.2011.12.005

Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J., 2012. The Home Care Crew Scheduling Problem: Preference-based visit clustering and temporal dependencies. European Journal of Operational Research 219, 598–610. https://doi.org/10.1016/j.ejor.2011.10.048

Rebaine, D., Strusevich, V., 1999. Two-machine open shop scheduling with special transportation times. Journal of the Operational Research Society 50, 756–794.

Reghioui, M., Prins, C., Labadi, N., 2007. GRASP with path relinking for the capacitated arc routing problem with time windows. Workshops on Applications of Evolutionary Computation, Springer, Berlin, Heidelberg 722–731.

Régin, J.-C., 2004. Modélisation et contraintes globales en programmation par contraintes (Habilitation universitaire). Université de Nice.

Régin, J.-C., 1994. A filtering algorithm for constraints of difference in CSPs. Proceedings AAAI-94 362–367.

Resende, M.G.C., Ribeiro, C.C., 2010. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications, in: Handbook of Metaheuristics. Boston, MA, pp. 283–319.

Rossi, F., van Beek, P., Walsh, T., 2006. Handbook of Constraint Programming, Elsevier.

Rossit, D.A., Tohmé, F., Frutos, M., 2018. The Non-Permutation Flow-Shop scheduling problem: A literature review. Omega 77, 143–153. https://doi.org/10.1016/j.omega.2017.05.010

Roy, B., Sussmann, B., 1964. Les problèmes d'ordonnancement avec contraintes disjonctives. Note Ds 9.

Ruiz, R., Vázquez-Rodríguez, J.A., 2010. The hybrid flow shop scheduling problem. European Journal of Operational Research 205, 1–18. https://doi.org/10.1016/j.ejor.2009.09.024

Sampels, M., Blum, C., Mastrolilli, M., Rossi-Doria, O., 2002. Metaheuristics for Group Shop Scheduling, in: International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, pp. 631–640. https://doi.org/10.1007/3-540-45712-7_61

Savelsbergh, M.W.P., Sol, M., 1995. The General Pickup and Delivery Problem. Transportation Science 29, 17–29. https://doi.org/10.1287/trsc.29.1.17

Sexton, T.R., Bodin, L.D., 1985. Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling. Transportation Science 19, 378–410.

Sha, D.Y., Hsu, C.-Y., 2008. A new particle swarm optimization for the open shop scheduling problem. Computers & Operations Research 35, 3243–3261. https://doi.org/10.1016/j.cor.2007.02.019

Stein, D.M., 1978. Scheduling Dial-a-Ride Transportation Systems. Transportation Science 12, 232–249. https://doi.org/10.1287/trsc.12.3.232

Tasan, A.S., Gen, M., 2012. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. Computers & Industrial Engineering 62, 755–761. https://doi.org/10.1016/j.cie.2011.11.025

Toth, P., Vigo, D. (Eds.), 2014. Vehicle routing: problems, methods, and applications, Society for Industrial and Applied Mathematics.

Toth, P., Vigo, D. (Eds.), 2002. The vehicle routing problem, SIAM monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics, Philadelphia, Pa.

Toussaint, H., 2010. Algorithmique rapide pour les problèmes de tournées et d'ordonnancement. (Thèse de doctorat). Université Blaise Pascal-Clermont-Ferrand II, Clermont-Ferrand.

Turing, A.M., 1937. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society s2-42, 230–265. https://doi.org/10.1112/plms/s2-42.1.230

Ulusoy, G., 1985. The fleet size and mix problem for capacitated arc routing. European Journal of Operational Research 22, 329–337. https://doi.org/10.1016/0377-2217(85)90252-8

Van Looveren, A., Gelders, L., Van Wassenhove, L., 1986. A review of FMS planning models. Modelling and design of flexible manufacturing systems 3–31.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2014. Implicit depot assignments and rotations in vehicle routing heuristics. European Journal of Operational Research 237, 15–28. https://doi.org/10.1016/j.ejor.2013.12.044

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research 40, 475–489. https://doi.org/10.1016/j.cor.2012.07.018

Wallace, M., 1996. Survey: Practical Applications of Constraint Programming. Constraints 1, 139–168.

Wolf, S., Merz, P., 2007. Evolutionary local search for the super-peer selection problem and the p-hub median problem., in: International Workshop on Hybrid Metaheuristics. Springer, Berlin, Heidelberg, pp. 1–15.

Yin, Y., Wu, W.-H., Cheng, T.C.E., Wu, C.C., 2015. Single-machine scheduling with time-dependent and position-dependent deteriorating jobs. International Journal of Computer Integrated Manufacturing 28, 781–790.

Chapitre II : Job-shop avec Transport et qualité de service – Job-shop avec Routing

Ce chapitre s'intéresse à la résolution conjointe de problèmes d'ordonnancement et de transport et plus particulièrement du problème de Job-shop avec Routing (JSPR) par opposition au problème de type Job-shop avec Transport (JSPT). Ces deux problèmes se modélisent sous forme de graphe disjonctif. Pour le JSPT, la solution du problème de transport n'est liée à aucun critère de qualité de service et la solution est donc souvent semi-active. Le Job-shop avec Routing considère explicitement les opérations de transport et utilise pour la résolution du problème de transport des algorithmes issus de la communauté transport. Il est montré que la partie routing du JSPR, est un problème de la famille des tournées de véhicules et en particulier de la famille des Pickup and Delivery Problem. L'algorithme classique, dans la communauté routing, proposé par (Cordeau and Laporte, 2003) est adapté au problème du JSPR (Gondran et al., 2017, 2018a, 2019).

La Qualité de Service (QoS) dans le JSPR est définie par la durée des tournées, la durée de transport des pièces et le temps d'attente de celles-ci. Une nouvelle fonction d'évaluation – nommée TLH – est proposée pour évaluer un graphe disjonctif en minimisant simultanément le makespan et en maximisant la qualité de service. Ainsi, la solution obtenue n'est pas semi-active, mais un compromis entre les différents critères. Cette fonction d'évaluation est incluse dans une métaheuristique. Un modèle PLNE est également proposé.

1. Introduction

Ce chapitre porte sur la modélisation et la résolution d'un nouveau problème intégré : le Job-shop Scheduling Problem with Routing (JSPR) qui est une généralisation du Job-shop Scheduling Problem with Transport (JSPT). Le JSPT fait partie des problèmes « classiques » de la littérature pour modéliser les systèmes de production avec transport. Le Job-shop Scheduling Problem with Transport est une extension du Job-shop Scheduling Problem (JSP). Le JSPT prend en compte la gestion d'une flotte de véhicules qui doit transporter les pièces entre les machines. Le JSPR se définit comme un problème de type JSPT dans lequel est ajouté un critère de qualité de service concernant le transport. Contrairement au JSPT qui a pour unique objectif de minimiser le makespan (date de fin de la dernière opération), le JSPR prend en compte la maximisation de la qualité de service lors de la minimisation du makespan.

Ce chapitre propose, entre autres, une nouvelle fonction d'évaluation du graphe disjonctif du JSPT, pour obtenir une solution du JSPR. Le graphe disjonctif du JSPT est introduit par (Hurink and Knust, 2005) et repris dans (Lacomme et al., 2013; Zeng et al., 2015, 2014; Zhang et al., 2014). L'approche présentée pour le JSPR diffère du JSPT par les points suivants :

- La définition de la qualité de service pour le JSPR, qui est comparable à la qualité de service définie pour le Dial-A-Ride Problem (DARP).

- Une nouvelle fonction d'évaluation du graphe disjonctif qui minimise le makespan et maximise la qualité de service.
- Une extension des benchmarks du JSPT pour englober le JSPR et les cas d'une flotte de véhicules de capacité non unitaire.

La nouvelle fonction d'évaluation est incluse dans un schéma de résolution de type métaheuristique. Un modèle PLNE pour le JSPR est également introduit dans ce chapitre.

Ce chapitre contient cinq parties : la première est une introduction comprenant notamment l'état de l'art du JSPT et la définition de la qualité de service pour le DARP. La seconde section concerne la définition du nouveau problème qui est le JSPR. La troisième partie est une formalisation linéaire du JSPR. La quatrième partie a pour objet la proposition d'une nouvelle approche de résolution du JSPR en prenant en compte simultanément la minimisation du makespan et la maximisation de la qualité de service. La cinquième section est une évaluation numérique de l'approche et permet de montrer l'intérêt d'introduire la notion de qualité de service.

1.1. Le Job-shop Scheduling Problem

Le Job-shop Scheduling Problem (JSP) est un problème d'ordonnancement très étudié dans la littérature. Les deux états de l'art proposés, respectivement, par (Blazewicz et al., 1996) et par (Jain and Meeran, 1999) font toujours référence car, à notre connaissance et malgré les nombreuses études, il n'y a pas d'état de l'art plus récent sur le JSP. Toutefois (Çaliş and Bulkan, 2015) proposent un état de l'art partiel se focalisant sur les approches issues de l'intelligence artificielle pour le JSP. Le JSP est prouvé NP-complet par (Lenstra et al., 1977). (Manne, 1960) fait partie des premiers à présenter une formulation linéaire du Job-shop Scheduling Problem.

L'objectif du Job-shop Scheduling Problem (JSP) est d'ordonnancer la fabrication de pièces (aussi appelées jobs) sur des machines. L'ensemble des pièces est noté $J = \{J_1, J_2, \dots, J_n\}$, et l'ensemble des machines est noté $M = \{M_1, M_2, \dots M_m\}$. Chacune des pièces J_i est définie par une gamme opératoire G_i qui est une succession d'opérations $G_i = \{O_{i,1}, O_{i,2}, \dots, O_{i,n}\}$ à exécuter. Chaque opération $O_{i,j}$ est définie par un temps de traitement (ou processing time) $Pt_{i,j}$ et par une machine $\mu_{i,j}$ qui doit réaliser l'opération. L'ensemble des opérations (tous jobs confondus) est noté O. Avant d'être traitée, la pièce attend dans le buffer (ou stock) d'entrée de la machine, et à la fin de son traitement, elle est située dans le buffer (ou stock) de sortie de la machine. Les hypothèses communément admises pour le JSP sont les suivantes :

- Les opérations d'un job doivent être exécutées dans l'ordre de sa gamme.
- Une opération ne peut commencer que si l'opération précédente appartenant à la même gamme est finie.
- Aucune préemption n'est autorisée.
- Une machine ne peut traiter qu'une seule opération à la fois.
- Une opération $O_{i,j}$ ne peut s'exécuter que sur une seule machine ($\mu_{i,j}$).
- Tous les jobs ont une et une seule opération à exécuter sur chacune des machines, le problème est donc de taille rectangulaire.
- Les temps de traitement $Pt_{i,j}$ des opérations sont connus et déterministes.

- Les temps de transport, de montage et de démontage sont négligés ou compris dans les temps $Pt_{i,j}$ des opérations.
- Tous les jobs et les machines sont disponibles dès le début de l'ordonnancement.
- Une éventuelle panne de machine ou tout incident n'est pas pris en compte.
- Les files d'attente (ou buffers) devant les machines sont infinies (c'est-à-dire que le nombre de pièces attendant avant de passer sur une machine n'est pas pris en compte).

L'intérêt du Job-shop Scheduling Problem réside dans sa fonction objectif qui est souvent la minimisation du makespan (date de fin de la dernière opération finissant le plus tard). Sa difficulté se trouve dans l'arbitrage des disjonctions : une machine ne peut traiter qu'une seule opération à la fois et il est donc nécessaire de définir dans quel ordre les différentes opérations sont éxécutées sur chaque machine. Une solution du Job-shop est définie : *(i)* par l'arbitrage des disjonctions, c'est-à-dire, en donnant l'ordre de passage des pièces sur chaque machine ; et *(ii)* par la date de début de chaque opération.

Le Job-shop Scheduling Problem est communément modélisé par le graphe disjonctif introduit par (Roy and Sussmann, 1964) (voir chapitre 1) et noté $G = (V, E)$. V est l'ensemble des nœuds représentant les opérations (cet ensemble contient également deux nœuds fictifs 0 et $*$, modélisant, respectivement, le début et la fin du planning) ; et E est l'ensemble (i) des arcs conjonctifs correspondant aux contraintes de précédence entre chaque opération successive d'un même job (les gammes), et *(ii)* des arcs disjonctifs définissant les contraintes de précédence entre opérations nécessitant la même machine. Chaque arc reliant l'opération $O_{i,j}$ à l'opération $O_{i,j+1}$ a pour poids le processing time $Pt_{i,j}$ de $O_{i,j}$.

1.2. Job-shop Scheduling Problem with Transport

Les contraintes liées au transport sont présentes dans de nombreux problèmes d'ordonnancement : le Job-shop Scheduling Problem with Transport (JSPT) (Bilge and Ulusoy, 1995; Knust, 1999; Zheng et al., 2014); le Flexible Manufacturing Systems (FMS) (Caumond et al., 2009) ; le Flexible Job-shop Scheduling Problem with Transport (Zhang et al., 2012); le Resource-Constrained Project Scheduling Problem with Transport (Quilliot and Toussaint, 2012) et le Hoist Scheduling Problem (HSP), qui peut être modélisé comme un cas particulier du JSPT avec des time-lags minimaux et maximaux (Adnen El and Elhafsi, 2016; Honglin et al., 2017). Ces problèmes incluent, entre autres, la gestion des AGVs (Automated Guided Vehicles) et les problèmes d'ateliers de type Hoist Scheduling Problem où les opérations de transport sont réalisées par des véhicules et/ou par des robots. Sur l'ensemble de ces problèmes, le transport fait l'objet d'une modélisation implicite dans le sens où la fonction objectif ne tient pas compte de critères liés au transport.

La coordination entre le transport et l'ordonnancement peut être modélisée de deux façons : la première est une modélisation explicite du transport tandis que la seconde est une modélisation implicite du transport. Ce chapitre porte sur la qualité de service du transport dans le JSPT et nécessite donc une modélisation explicite du transport.

Le Job-shop Scheduling Problem avec Transport (JSPT) est une extension du JSP dans laquelle une flotte $R = \{R_1, \dots, R_r\}$ constituée de r véhicules doit transporter les pièces entre les machines. Chaque véhicule R_i a une capacité de transport de C^{Ri} pièces simultanément. Le transport est explicitement modélisé par deux opérations : une opération de chargement (où la pièce attend sur le buffer de sortie de la machine et est chargée dans le véhicule, aussi

appelée opération de pickup) et une opération de livraison (où la pièce est déposée sur le buffer d'entrée de la machine, aussi appelée opération de delivery). Une opération $O_{i,j}$ est dite opération-machine et les opérations de chargement et de livraison sont regroupées sous le nom d'opération-transport. La modélisation par deux opérations du transport permet de prendre en considération des véhicules de capacité unitaire ($\forall r \in R, C^r = 1$) et des véhicules de capacité non unitaire ($\forall r \in R, C^r \geq 1$).

Le JSPT est un problème NP-difficile puisque le Job-shop Scheduling Problem est NP-difficile (Lenstra and Kan, 1979), et le problème de transport est assimilable à un Pickup and Delivery Problem qui est lui aussi connu comme étant NP-Difficile (Lenstra and Kan, 1981). Pour le JSPR, le critère d'optimisation le plus courant est la minimisation du makespan. Le JSPR est classifié $JR|t_{kl}, t'_{kl}|C_{max}$ d'après la notation $\alpha|\beta|\gamma$ introduite par (Graham et al., 1979) et étendue par (Knust, 1999). J signifie job-shop, R indique que la flotte de véhicules contient un nombre limité de véhicules identiques et que les pièces doivent être transportées d'une machine à une autre. t_{kl} signifie que les temps de transport entre les machines sont dépendants des (positions des) machines et indépendants des pièces et t'_{kl} signifie que les déplacements à vide des véhicules est dépendant des (positions des) machines. La fonction objectif à minimiser est le makespan C_{max}.

Une solution du JSPT est caractérisée par la définition d'un ordre de passage des pièces sur chaque machine ; par l'affectation d'un véhicule à chaque opération de transport (chargement et livraison) ; par l'ordonnancement des opérations de transport de chaque véhicule ; et par le calcul des dates de début des opérations-machines et des opérations de transport.

(Knust, 1999) est le premier à se focaliser sur la formulation du JSPT avec un seul véhicule de capacité unitaire. (Hurink and Knust, 2002) et (Hurink and Knust, 2005) proposent une formalisation linéaire du JSPT avec un seul véhicule de capacité unitaire, une recherche taboue et des jeux d'instances pour le JSPT avec un seul véhicule de capacité unitaire.

(Bilge and Ulusoy, 1995) sont les premiers à introduire le JSPT avec une flotte de véhicules de capacité unitaire. Ils le résolvent grâce à une heuristique. Ils proposent également une formulation linéaire. Ils introduisent un ensemble de 82 instances qui est utilisé par de nombreux articles. (Ulusoy et al., 1997) introduisent le premier algorithme génétique pour résoudre le JSPT avec plusieurs véhicules de capacité unitaire, ainsi que des bornes inférieures pour les 82 instances. Historiquement, il s'agit de deux articles fondateurs qui introduisent à la fois, un nouveau problème et le premier jeu de données pour le JSPT.

(Abdelmaguid et al., 2004) proposent un algorithme génétique hybride pour résoudre le JSPT avec des véhicules de capacité unitaire. Leur heuristique est évaluée sur les instances de (Bilge and Ulusoy, 1995).

Dans sa thèse, (Caumond, 2006) s'intéresse au JSP avec des contraintes additionnelles telles que des time-lags minimaux et un Job-shop Scheduling Problem avec Transport. Les time-lags minimaux constituent une modélisation des temps de transport permettant d'obtenir des opérations-machines suffisamment espacées dans le temps pour pouvoir par la suite insérer les opérations de transport entre les opérations-machines. Les time-lags minimaux sont utilisés dans la modélisation de problèmes de niveaux tactiques ou opérationnels, c'est-à-dire lors de la conception de planning prévisionnel pour des horizons de planification suffisamment grands. (Caumond, 2006) propose un état de l'art sur les time-lags comprenant

leur définition générale et leur formulation mathématique. Par ailleurs, il introduit une modélisation efficace et une métaheuristique pour résoudre le JSP avec time-lags (Caumond et al., 2008). Quant au JSPT, (Caumond, 2006) propose une formulation mathématique (Caumond et al., 2009) et un algorithme génétique qui prennent en compte également des contraintes sur la gestion et la capacité des buffers d'entrée et de sortie des machines.

(Khayat et al., 2006) proposent un PLNE et une formulation sous forme de programmation par contraintes pour le JSPT avec une flotte de véhicules de capacité unitaire. Ils comparent les deux approches sur des instances basées sur celles de (Bilge and Ulusoy, 1995).

(Morihiro et al., 2006) esquissent les prémisses de la résolution du JSPT avec des contraintes sur les tournées de véhicules. Ils essayent, pour un ordonnancement donné, de planifier les tournées de la flotte de véhicules de capacité non unitaire. Ils prennent en considération les deadlocks (blocages des véhicules), des capacités finies pour les buffers d'entrée et de sortie et des fenêtres de temps pour les opérations de livraison. Leur fonction objectif est une somme pondérée du nombre de kilomètres parcourus par les véhicules et des temps de retards sur les points de livraison.

(Reddy and Rao, 2006) étudient la résolution conjointe d'un problème d'ordonnancement et d'AGVs (Automated Guided Vehicles) dans des FMS (Flexible Manufacturing System). Ils cherchent à minimiser plusieurs objectifs : le makespan, le mean flow et le mean tardiness avec la métaheuristique NSGA-II. Leur métaheuristique s'appuie sur un vecteur de jobs, donnant l'ordre de passage des pièces sur les machines. Lorsqu'une pièce nécessite d'être transportée, la méthode de résolution sélectionne le véhicule qui atteindra le plus tôt la machine sur laquelle le job est présent. Leur étude porte sur un JSPT avec une flotte de véhicules de capacité unitaire. Ils évaluent leur méthode sur les instances de (Bilge and Ulusoy, 1995).

(Abdelmaguid, 2007) propose un PLNE pour le Job-shop Scheduling Problem with Material Handling (JSPMH) avec pour unique objectif de minimiser le makespan dans le cas d'une flotte de véhicules de capacité unitaire. Il propose également une modélisation sous forme de graphe où chaque opération est représentée par quatre sommets, symbolisant respectivement : la date de début de la livraison, la date de fin de la livraison, la date de début du chargement et la date de fin du chargement. Une opération-machine est représentée par le poids de l'arc entre la date de fin de livraison et la date de début de chargement. Par conséquent, (Abdelmaguid, 2007) considère que l'opération-machine débute directement après la livraison. Cette modélisation est la base de l'article de (Abdelmaguid and Nassef, 2010) dans lequel les auteurs proposent une heuristique.

(Lacomme et al., 2007) et (Lacomme et al., 2013) proposent une modélisation du JSPT sous forme de graphe conjonctif-disjonctif pour une flotte de véhicules de capacité unitaire. Ce graphe disjonctif est une extension du graphe présenté par (Hurink and Knust, 2005). Ils proposent également une représentation d'une solution basée sur trois vecteurs : le premier concerne les disjonctions des opérations-machines, le second implique les disjonctions des opérations de transport et le troisième vecteur est relatif aux affectations des véhicules aux opérations de transport. Ils proposent un nouveau jeu d'instances.

(Kesen and Baykoç, 2007) démontrent l'utilisation possible d'un modèle de simulation pour un JSPT avec une politique de gestion des transports de type « Maximum priority » où les jobs sont envoyés en priorité vers la machine n'ayant réalisé aucune pièce depuis le plus

longtemps. Ils prennent en compte une flotte de véhicules de capacité unitaire et testent leur modèle sur leurs propres instances

(Deroussi et al., 2008) proposent une nouvelle représentation de l'espace de codage pour un JSPT avec une flotte de véhicules de capacité unitaire. Cet espace de codage est basé sur les véhicules plutôt que sur les machines et a pour avantage d'être plus réduit que celui proposé par (Ulusoy et al., 1997). Ils ne considèrent pas les opérations-machines, mais uniquement des opérations de transport qui doivent être ordonnancées, la durée d'une opération de transport dépend de la durée de l'opération-machine précédente. Leur fonction de décodage est une simulation à événement discret. Cet espace de codage et cette fonction de décodage sont insérés dans trois métaheuristiques : un ILS, un recuit-simulé et une hybridation des deux.

(Caumond et al., 2009) proposent un PLNE pour un problème d'ordonnancement avec transport pour un seul véhicule de capacité unitaire et appliqué au cas des FMS (Flexible Manufacturing System). Ils prennent en compte des contraintes de capacités des buffers d'entrée et de sortie, un nombre maximal de jobs présents simultanément dans le système, une règle de type FIFO (First In, First Out) pour les buffers d'entrée et de sortie, une règle de non-anticipation des déplacements à vide des véhicules (le véhicule ne peut pas partir à destination d'un buffer de sortie d'une machine tant que la pièce à charger n'est pas disponible). Ils définissent de nouveaux benchmarks fondés sur les instances de (Bilge and Ulusoy, 1995).

(Subbaiah et al., 2009) étudient le JSPT pour une flotte de véhicules de capacité unitaire avec une métaheuristique basée sur le comportement d'un troupeau de moutons. Celle-ci repose sur une population d'individus et est relativement proche d'un algorithme génétique. Ils minimisent les critères du makespan et du mean tardiness. Ils utilisent les benchmarks de (Bilge and Ulusoy, 1995) pour tester leur approche.

(Babu et al., 2010) et (Kumar et al., 2011) présentent un algorithme à Évolution Différentielle (DE) pour le JSPT avec une flotte de véhicules de capacité unitaire, et avec pour critère la minimisation du makespan. Ils testent leur méthode sur les instances de (Bilge and Ulusoy, 1995).

(Deroussi and Norre, 2010) soulèvent une problématique de JSPT, pour une flotte de véhicules de capacité unitaire, avec la partie ordonnancement « flexible », c'est-à-dire que chaque pièce peut être réalisée par plusieurs machines, et non pas par une unique machine comme c'est le cas pour le JSPT classique. Une solution du problème est similaire à une solution du JSPT et doit en plus affecter une machine à chaque opération. Ils proposent un ILS et de nouvelles instances spécifiques à ce problème.

(Larabi, 2010) étudie les différentes combinaisons possibles du JSPT : un seul véhicule de capacité unitaire, un véhicule de capacité non unitaire, une flotte de véhicules de capacité unitaire, et enfin plusieurs véhicules de capacité non unitaire. Pour chacun des problèmes, il propose une formulation mathématique et un algorithme génétique. (Larabi, 2010) introduit une modélisation du JSPT avec une flotte de véhicules de capacité non unitaire sous forme d'un graphe conjonctif-disjonctif comprenant les opérations-transport et les opérations-machines. Les opérations de transport sont divisées en deux sous-ensembles : les opérations de chargement après une opération-machine où la pièce est chargée sur le véhicule, et les opérations de livraison où la pièce est déposée avant une opération-machine. (Larabi, 2010) utilise la représentation de l'espace de codage de (Lacomme et al., 2007). Cet espace de

codage permet une évaluation en complexité polynomiale du graphe disjonctif et évite tout cycle. Il définit un jeu d'instances pour le JSPT avec une flotte de véhicules de capacité non unitaire.

(El Khoukhi et al., 2011) proposent une colonie de fourmis pour résoudre le JSPT Just-In-Time, avec un ou plusieurs véhicules de capacité unitaire. Le système étudié est Just-In-Time (JIT) signifiant que la demande n'est pas connue en avance, mais en temps réel. L'objectif de leur étude est la minimisation des pénalités de retard, des livraisons ayant été effectuées « trop tôt » c'est-à-dire avant leurs dates de livraison souhaitées, et du nombre de trajets à vide des véhicules. Des contraintes sont ajoutées sur la capacité des buffers d'entrée et de sortie. Leur méthode est évaluée sur les instances de (Larabi, 2010).

(Erol et al., 2012) utilisent un système multiagents pour résoudre le JSPT Just-In-Time. Ils expliquent qu'une heuristique basée sur les multiagents est intéressante dans le cas de système décentralisé, c'est-à-dire que le problème est découpé en sous-systèmes, chacun contrôlé par un agent indépendant qui agit de manière locale avec les informations locales dont il dispose. Un système centralisé comporte un contrôleur qui a accès à toutes les informations et gère tout le système, le contrôleur a une vision globale du système en temps réel et prend toutes les décisions. Ils proposent une architecture multiagents pour le JSPT et expérimentent celle-ci sur les instances de (Bilge and Ulusoy, 1995) dans le cadre d'une flotte de véhicules de capacité unitaire.

(Zhang et al., 2012) s'intéressent au JSPT flexible avec des temps de traitement bornés (les processing times ne sont pas fixes, mais appartiennent à un intervalle). Les temps de traitement étant bornés, ils ne calculent pas les dates de début des opérations, mais ils définissent une fenêtre durant laquelle l'opération peut avoir lieu. Leur fonction objectif prend en considération à la fois le makespan et le temps total d'attente (Total Waiting Time) des pièces dans les buffers d'entrée et de sortie. Ils considèrent le temps d'attente dans les buffers d'entrée comme étant la différence entre le début de l'opération au plus tôt et la fin de l'opération précédente au plus tard. De manière similaire, pour le temps d'attente dans les buffers de sortie, ils le définissent comme la différence entre la date de début au plus tard de l'opération avec le temps d'exécution le plus long et la date de fin au plus tôt de l'opération précédente. Ils proposent une formulation mathématique non linéaire pour leur étude et un algorithme génétique. Ils testent leur méthode sur des instances issues du Flexible Job-shop Scheduling Problem (FJSP) et du Hoist Scheduling Problem (HSP), en prenant un ou plusieurs véhicules de capacité unitaire. La proposition de ce manuscrit est de prendre en compte plus de critères de qualité de service, notamment le temps de trajet des jobs et le temps de réalisation d'un job complet. La définition du temps d'attente des jobs proposée dans ce manuscrit diffère de leur définition.

(Lacomme et al., 2013) concentrent leur étude sur le JSPT avec un ou plusieurs véhicules de capacité unitaire et la minimisation du makespan. Ils étendent les travaux de la thèse de (Larabi, 2010) en proposant une modélisation avec un graphe disjonctif permettant de prendre un compte une flotte de véhicules de capacité non unitaire, et un algorithme mémétique. Leurs études numériques utilisent les instances de (Bilge and Ulusoy, 1995) et celles de (Hurink and Knust, 2005).

(Nageswararao et al., 2014) promeuvent un « Binary particle swarm vehicle heuristic Algorithm » pour résoudre le JSPT avec une flotte de véhicules de capacité unitaire. Ils minimisent le retard moyen (mean tardiness) et prennent un compte un facteur de robustesse.

Ils définissent la robustesse comme étant l'inverse du makespan (robustesse = 1 / makespan). Ils testent leur méthode sur les instances de (Bilge and Ulusoy, 1995).

(Zeng et al., 2014) proposent une extension du JSP avec le Blocking Job-shop with Transport, pour une flotte de véhicules de capacité unitaire. Une machine est bloquée et inutilisable tant que le job n'est pas récupéré par un véhicule. Ils proposent deux modèles non linéaires en nombres entiers et une heuristique dédiée à leur problème. Ils testent leur méthode sur les instances de (Bilge and Ulusoy, 1995).

(Zhang et al., 2014) étudient un JPS avec des contraintes de transport et des temps de traitement bornés : ces derniers ne sont pas fixes, mais compris dans un intervalle. Les capacités des buffers des machines sont limitées. Ils proposent une modélisation explicitant les opérations de stockage et une heuristique de type Shifting Bottleneck. Ils testent cette dernière sur les instances de (Bilge and Ulusoy, 1995), de (Hurink and Knust, 2005) et du FJSP avec transport de (Deroussi and Norre, 2010).

(Zheng et al., 2014) proposent un PLNE pour JSPT avec une flotte de véhicules de capacité unitaire et une recherche taboue. Leur PLNE n'est pas basé sur un flot, mais sur des « moving tasks ». Ils expérimentent leur PLNE sur de nouvelles instances et leur méthode taboue sur les instances de (Bilge and Ulusoy, 1995).

(Sahin et al., 2015) soumettent une approche basée sur un système multiagents pour un JSPT, dans le cas d'une flotte de véhicules de capacité unitaire, avec des machines flexibles dans un environnement dynamique (Just-In-Time). Leur méthode est testée sur les instances (Bilge and Ulusoy, 1995).

(Umar et al., 2015) traitent du JSPT pour une flotte de véhicules de capacité unitaire avec pour objectifs à optimiser : le makespan, les temps de trajets des AGVs et le retard des pièces. Ils prennent en compte les conflits dans le réseau routier (suivant les zones, les AVGs ne peuvent pas se doubler, ils ne peuvent pas se croiser, etc.). Ils proposent un algorithme génétique hybride qui se décompose en plusieurs étapes : la résolution du problème d'ordonnancement des jobs, puis la génération du besoin de ressources pour le transport, ensuite l'affectation des AGVs et enfin le parcours effectué par chaque AVG pour éviter les conflits. Ils testent leur algorithme sur les instances de (Bilge and Ulusoy, 1995) enrichies.

(Zeng et al., 2015) s'intéressent à un problème de Job-shop multi-cellules avec des opérations de transport entre les cellules. Dans chaque cellule, un ordonnancement correspondant à un Job-shop doit être réalisé, puis un transport des produits entre les cellules est organisé. Ces deux étapes sont réalisées au moyen d'une heuristique à deux niveaux basée sur un algorithme mémétique : un niveau pour l'intracellule (concernant le JSP) et un autre niveau pour l'intercellule (résolution du transport). Ils testent leur algorithme sur de nouvelles instances basées sur celles de (Hurink and Knust, 2005).

(Xie and Allen, 2015) proposent un état de l'art du Job-Shop with material handling, englobant notamment le Job-Shop avec Transport. Ils insistent sur l'importance de prendre en compte le transport dans les problèmes d'ordonnancement, car ils estiment que la supposition qu'il n'y ait pas de temps de transport entre les machines ne reflète pas la réalité. Ils notent également le grand nombre de méthodes approchées et le faible nombre d'articles avec une approche exacte. Ils classifient les articles en deux catégories, ceux avec une approche statique et ceux avec une approche dynamique (Just-In-Time).

(Baruwa and Piera, 2016) affirment que pour obtenir des ordonnancements (du JSPT avec une flotte de véhicules de capacité unitaire) de bonne qualité en termes de makespan, il est impératif de traiter de façon intégrée l'ordonnancement des opérations et le transport des pièces. Ils proposent une heuristique basée sur les réseaux de Petri. Ils testent leur méthode sur les instances de (Bilge and Ulusoy, 1995).

(Nouri et al., 2016c) proposent un état de l'art du JSPT : 25 papiers sont recensés entre 1995 et 2014. Ils classifient la littérature suivant 7 critères : nombre de ressources pour le transport, leur type (AGV, Robot, Material Handling Vehicles, Transport Resources), la complexité des gammes (une opération ou plus), la flexibilité du système de production (Job-shop flexible ou non), contrainte de recirculation (une machine visitée plusieurs fois par le même job), les critères d'optimisation (monocritère, multicritères), et les méthodes implémentées.

(Nouri et al., 2016a), (Nouri et al., 2016b) et (Nouri et al., 2016d) proposent une série d'articles pour la résolution du JSPT avec une heuristique basée sur un modèle multiagents holonics. Dans (Nouri et al., 2016a), un seul véhicule de capacité unitaire est opérationnel, le JSP est non flexible. Dans (Nouri et al., 2016d), le cas traité correspond à une flotte de véhicules de capacité unitaire et un JSP non flexible. Dans (Nouri et al., 2016b), une flotte de véhicules de capacité unitaire est considérée et le JSP est flexible. Leur méthode est testée sur les instances de (Bilge and Ulusoy, 1995).

(Zheng et al., 2016) s'intéressent au JSPT Just-In-Time pour une flotte de véhicules de capacité unitaire. Ils décrivent les différences entre une approche centralisée (tous les AVGs sont gérés par un unique contrôleur) et une approche distribuée (plusieurs contrôleurs distribués supervisent les AVGs). Ils promeuvent une métaheuristique basée sur la régulation d'hormone par l'humain pour une approche distribuée, elle est testée sur les instances de (Bilge and Ulusoy, 1995).

(Ahmadi-Javid and Hooshangi-Tabrizi, 2017) s'estiment (d'après leur article) être les premiers à s'intéresser à un problème de JSPT avec un problème d'affectation des employés aux machines dépendant des compétences des travailleurs. Ils utilisent une flotte de véhicules de capacité unitaire. Ils proposent un PLNE et un « Anarchic Society Optimization algorithm ». Ils testent leur PLNE et leurs deux approches sur des instances générées à partir de celles de (Bilge and Ulusoy, 1995).

1.3. Positionnement des travaux du manuscrit

Le Tableau 1 résume les articles traitant du JSPT publiés durant les vingt-cinq dernières années.

La première colonne du Tableau 1 indique le nom de l'article, la seconde colonne est le nombre de véhicules contenus dans la flotte soit 1 ($= 1$), soit plusieurs (> 1). La troisième et la quatrième colonnes sont les capacités du/des véhicules (soit unitaire, soit non unitaire), la cinquième colonne offre les contraintes supplémentaires étudiées. Les colonnes six et sept concernent les approches heuristiques et exactes, les colonnes huit, neuf et dix portent sur les critères de la fonction objectif, respectivement, makespan ($Cmax$), qualité de service (QoS) ou autre. Les colonnes onze et douze se rapportent aux benchmarks utilisés.

Tableau 1 Liste des publications des vingt-cinq dernières années (Gondran et al., 2018a)

	Transport			**Contraintes**	**Approches**		**Critère optimisé**			**Benchmark**	
	Nb véhicules	Capacité unitaire	Capacité non unitaire		Heuristique	Exacte	$Cmax$	QoS	Autres	Bilge and Ulusoy	Autres
(Bilge and Ulusoy, 1995)	>1	X			Time window approach	PLNE	X			***Bilge and Ulusoy**	
(Ulusoy et al., 1997)	>1	X			Genetic Algorithm		X			X	
(Hurink and Knust, 2002)	=1	X			Tabu Search		X				***Knust**
(Hurink and Knust, 2005)	=1	X			Tabu Search		X				Knust
(Abdelmaguid et al., 2004)	>1	X			Hybrid Genetic Algorithm		X			X	
(Caumond, 2006)	>1	X		Capacités des buffers limitées	Genetic Algorithm	PLNE	X				X
(Khayat et al., 2006)	>1	X				PLNE, PPC	X				X
(Morihiro et al., 2006)	>1		X	Capacités des buffers limitées, fenêtre de temps, deadlock-free	Distributed agents	Formulation mathématique			Distance, retard		X
(Reddy and Rao, 2006)	>1	X			Evolutionary algorithm		X		Durées des jobs Retard moyen	X	
(Abdelmaguid, 2007)	>1	X				PLNE	X				
(Lacomme et al., 2007)	>1	X			Disjunctive Graph		X				X
(Kesen and Baykoç, 2007)	>1	X			Simulation		X				Knust
(Deroussi et al., 2008)	>1	X			ILS, SA et SALS					X	
(Caumond et al., 2009)	=1	X		Capacités des buffers limitées, nombre maximal de jobs, règle de gestion	Time window approach	PLNE	X				X
(Subbaiah et al., 2009)	>1	X			Sheep flock heredity		X		Retard moyen	X	
(Abdelmaguid and Nassef, 2010)	>1	X			GTA		X				X
(Babu et al., 2010)	>1	X			DE		X			X	
(Deroussi and Norre, 2010)	>1	X		Flexible	ILS		X				***FJSP**
(Larabi, 2010)	>1 and =1	X	X		Memetic Algorithm		X			X	***Larabi** / Knust
(El Khoukhi et al., 2011)	>1 and =1	X	X	Just-In-Time, capacités des buffers limitées	Ant Colony	Formulation mathématique	X		Retard, Déplacement à vide		Larabi

Tableau 1 (suite) Liste des publications des vingt-cinq dernières années (Gondran et al., 2018a)

	Transport			**Contraintes**	**Approches**		**Critère optimisé**			**Benchmark**	
	Nb véhicules	Capacité unitaire	Capacité non unitaire		Heuristique	Exacte	$Cmax$	QoS	Autres	Bilge and Ulusoy	Autres
(Kumar et al., 2011)	>1	X			Differential evolution		X			X	
(Erol et al., 2012)	>1	X		Just-In-Time	Multiagent		X			X	
(Zhang et al., 2012)	>1 and =1	X		Flexible	Genetic Algorithm	Formulation mathématique	X		Temps d'attente	X	FJSP / HSP / Knust
(Lacomme et al., 2013)	>1	X			Memetic Algorithm		X		Retard moyen, Robustesse	X	Knust
(Nageswararao et al., 2014)	>1	X			Binary particle swarm		X			X	
(Zeng et al., 2014)	>1	X		Blocking Job-shop	Heuristique		X			X	X
(Zeng et al., 2015)	=1	X		Multi-cellules	Genetic Algorithm		X				X
(Zhang et al., 2014)	>1	X		Flexible, temps de stockage	Shifting bottleneck heuristic		X			X	FJSP / Knust
(Zheng et al., 2014)	>1	X			Tabu search	Formulation mathématique	X			X	X
(Sahin et al., 2015)	>1	X		Just-In-Time	Multiagent		X			X	
(Umar et al., 2015)	>1	X			Genetic Algorithm		X		Temps de transport des AGVs, Retard, conflit	X	X
(Xie and Allen, 2015)				Statique et Just-In-Time	État de l'art						
(Baruwa and Piera, 2016)	>1	X			Réseau de Petri		X			X	
(Nouri et al., 2016c)					État de l'art		X				
(Nouri et al., 2016d)	>1 and =1	X			Holonic multiagents		X			X	Knust
(Nouri et al., 2016a)	=1	X			Holonic multiagents		X			X	
(Nouri et al., 2016b)	>1 and =1	X		Flexible	Holonic multiagents		X			X	Knust /FJSP
(Zheng et al., 2016)	>1	X		Just-In-Time	Hormone regulation		X			X	
(Ahmadi-Javid and Hooshangi-Tabrizi, 2017)	>1	X		Gestion du planning des employés	Anarchic Society Optimization algorithm		X			X	
(GONDRAN et al, 2018)	**>1**	**X**	**X**		**GRASP×ELS**	**PLNE**	**X**	**X**		**X**	

Il est important de remarquer que seulement trois articles prennent en considération une flotte de véhicules de capacité non unitaire, (Morihiro et al., 2006), (Larabi, 2010), (El Khoukhi et al., 2011), mais aucun ne fournit de résultats numériques sur des instances. Ils proposent des études comparatives – en pourcentage – sur le gain potentiel lié à l'utilisation d'une flotte de véhicules de capacité non unitaire par rapport à une flotte de véhicules de capacité unitaire. Ils ont démontré qu'un gain peut être réalisé.

Plusieurs formalisations mathématiques et PLNE sont proposés pour le JSPT, mais il n'y a aucun article qui offre un PLNE capable d'être résolu par un solveur linéaire, et quasiment tous les articles proposent une méthode de résolution approchée. La grande majorité des articles ont pour fonction objectif la minimisation du makespan, certains prennent en compte des contraintes sur le transport, mais aucun article ne prend en considération un critère de qualité de service pour le routage, excepté (Zhang et al., 2012) qui essaye de minimiser les temps d'attente dans un JSPT flexible.

Et enfin, il est important de garder à l'esprit qu'une des contributions majeures de ce chapitre porte sur la proposition d'une métaheuristique originale et plus particulièrement sur la conception d'une fonction d'évaluation – Time-Lag Heuristic (TLH) – d'un graphe disjonctif pour minimiser le makespan et maximiser simultanément la qualité de service.

La fonction d'évaluation – Time-Lag Heuristic (TLH) – d'un graphe disjonctif prend en compte un critère de qualité de service et se base sur l'algorithme (Cordeau and Laporte, 2003) issu des problèmes de tournées de véhicules pour le Dial-A-Ride-Problem. Une gamme du JSPT peut être vue comme une succession de chargement et de livraison. Le JSPR met en évidence cette analogie et explicite la partie routing du problème. La prochaine section vise donc à rappeler la définition de la qualité de service dans le Dial-A-Ride Problem d'après (Cordeau and Laporte, 2003).

1.4. Qualité de service pour le Dial-A-Ride Problem

Le Dial-A-Ride Problem (DARP) est introduit par (Stein, 1978) (cf. chapitre 1) et il est formalisé par (Cordeau and Laporte, 2003, 2007), qui propose la définition suivante : chaque client p a une requête de transport entre un point de ramassage (nœud de chargement ou nœud de pickup) et un point de livraison (nœud de livraison ou nœud de delivery). Chaque nœud i (de chargement ou de livraison) est associé : à une durée de service d_i qui correspond au temps nécessaire pour déposer ou charger le client p_i ; et à une fenêtre de temps $[E_i\ ;\ L_i]$ durant laquelle l'opération de chargement ou de livraison doit être effectuée. Un véhicule arrivant avant la fenêtre de temps d'une opération doit attendre l'ouverture de celle-ci avant de commencer le service. Le transport du client est assuré par une flotte homogène et limitée de véhicules de capacité Q. Toutes les tournées commencent et finissent au dépôt.

L'originalité du DARP proposé par (Cordeau and Laporte, 2003) réside dans le type de solution recherché : les dates des visites ne sont pas nécessairement calées à gauche (au plus tôt) afin de maximiser la qualité de service. Démarrer une visite plus tard peut permettre de limiter, par exemple, le temps passé par un client dans le véhicule. Généralement, pour les problèmes « classiques », un calcul des dates au plus tôt est suffisant pour ordonnancer les opérations de transport. Un algorithme résolvant un DARP maximise la qualité de service du point de vue du client, ce qui signifie qu'il faut définir un algorithme pour déterminer la date d'arrivée du véhicule sur un nœud, la date de début de service (date qui peut être différente de

la date d'arrivée), la date de fin de service et la date de départ du véhicule. Quatre variables sont nécessaires pour fournir une description d'une tournée (Figure 1) :

- A_i est la date d'arrivée du véhicule au nœud i.
- st_i est la date de début du service au nœud i.
- C_i est la date de fin du service au nœud i, $C_i = st_i + d_i$.
- D_i est la date de départ du véhicule du nœud i.

(Cordeau and Laporte, 2003) introduisent trois critères (Figure 1) pour répondre aux considérations de qualité de service du DARP :

- R_p est le temps de transport (Riding Time) du client p, il est défini entre la date de départ du nœud i où le client est chargé et sa date de début sur son nœud de livraison i'. Le nœud de livraison n'est pas nécessairement le nœud suivant directement le nœud de chargement, plusieurs clients peuvent être chargés et/ou déchargés entre les deux. $R_p = st_{i'} - D_i$.
- DT_r est la durée du trajet (Duration Time) du véhicule r, définie comme la différence entre la date d'arrivé E_n^r du véhicule au dépôt et sa date de départ B_0^r du dépôt, $DT_r = E_n^r - B_0^r$.
- WT_p est le temps d'attente (Waiting Time) du client p entre : (1) sa date D_i de départ du nœud i et la date C_i de fin de son service au nœud i, $WT_p^1 = B_i - C_i$; (2) sa date de début de service st_i au nœud i et sa date A_i d'arrivée sur le nœud i, $WT_p^2 = st_i - A_i$. Le temps d'attente est donné par : $WT_p = WT_p^1 + WT_p^2$.

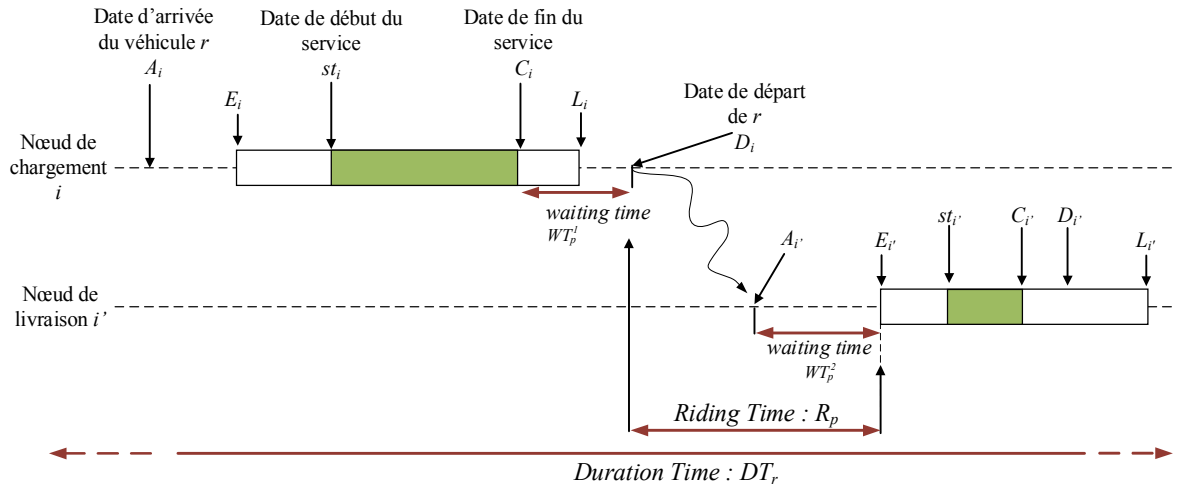


Figure 1 Description d'une tournée

Une solution du DARP est définie par un ensemble de tournées qui répondent à toutes les demandes en respectant d'une part les fenêtres de temps et d'autre part la capacité maximale des véhicules. De plus, les critères suivants sont inclus dans la mesure de la qualité de service d'une solution :

- Le Total Riding Time : $TRT(s) = \sum_{p=1}^{n} R_p$.
- Le Total Duration : $TD(s) = \sum_{j=1}^{k} D_j$, où k est le nombre de tournées.
- Le Total Waiting Time : $TWT(s) = \sum_{i=1}^{2n} WT_i$.

Compte tenu des trois critères précédemment cités, une solution du DARP n'est pas semi-active, c'est-à-dire que les dates de début des différentes opérations (départ, début de service,…) ne sont pas nécessairement planifiées au plus tôt (ou collées à gauche). Pour des tournées données, les dates de début minimisant les différents critères peuvent être calculées par l'algorithme en $O(n^2)$ de (Cordeau and Laporte, 2003) ou en $O(n)$ par l'algorithme de

(Firat and Woeginger, 2011). Ces deux algorithmes ne fournissent pas les mêmes dates, mais un compromis différent entre ces critères. L'algorithme de (Cordeau et Laporte, 2003) est couramment utilisé dans les approches heuristiques, et tire parti d'un graphe acyclique auxiliaire où les critères sont minimisés de façon itérative sans détériorer les critères minimisés précédents.

2. Le Job-shop Scheduling Problem with Routing

Le Job-shop Scheduling Problem with Routing (JSPR) consiste à résoudre le Job-shop Scheduling Problem avec une modélisation explicite du transport et un critère de qualité de service associé au transport. Pour la résolution de ce problème, une démarche originale basée sur des approches issues de problèmes de tournées de véhicules et des approches issues de problèmes d'ordonnancement est proposée. Cette démarche comprend : la modélisation explicite du transport et la prise en compte d'un critère de Qualité de Service (QoS) dans la fonction objectif. La modélisation du transport dans le JSPR par des opérations de chargement et des opérations de livraison offre un paradigme tout à fait adapté aux problèmes de tournées de véhicules, et permet de tirer profit des algorithmes les plus efficaces issus des communautés des problèmes de tournées de véhicules et des problèmes d'ordonnancement.

2.1. Modélisation sous forme de graphe

Le graphe conjonctif-disjonctif du Job-shop Scheduling Problem (Roy and Sussmann, 1964) est généralisé par (Hurink and Knust, 2005) puis par (Lacomme et al., 2007) pour le JSPT en ajoutant des nœuds pour modéliser les opérations de chargement et de livraison par les véhicules. Ainsi, dans le graphe disjonctif il existe trois types de sommets qui modélisent : (*i*) les opérations $O_{i,j}$ de traitement ; (*ii*) les opérations $P_{i,j}$ de chargement (pickup) ; (*iii*) les opérations $D_{i,j}$ de livraison (delivery ou chargement) (Gondran et al., 2017).

Le triplet ($D_{i,j}$; $O_{i,j}$; $P_{i,j}$), représenté par la Figure 2, définit un ensemble de trois opérations représentant successivement une opération de livraison (ou déchargement), une opération-machine et une opération de chargement.

De manière générale, la date de début d'une opération-machine $O_{i,j}$ dépend de la date de fin de l'opération de livraison $D_{i,j}$ (le temps de livraison/déchargement est supposé nul ou inclut dans le temps de transport). La date de début d'une opération de chargement $P_{i,j}$ doit être supérieure à la date de fin de l'opération $O_{i,j}$ (plus le temps de chargement qui est lui aussi supposé nul ou inclut dans le temps de traitement, sans perte de généralité).

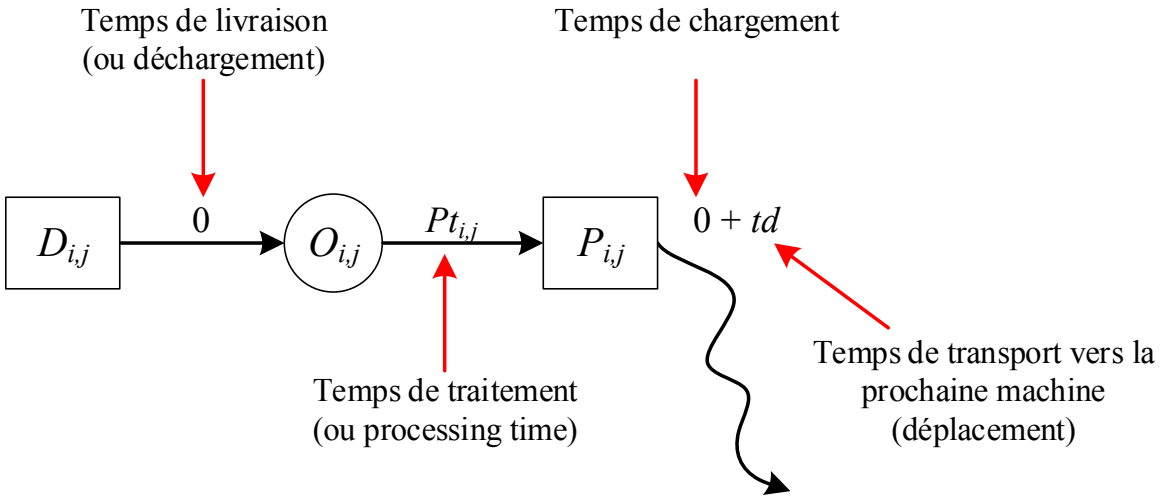


Figure 2 Arcs conjonctifs définissant le passage d'un job sur une machine

Par conséquent, chaque job j possédant n_j opérations-machines dans sa séquence contient $3 \times n_j - 2$ nœuds : n_j opérations-machines, $n_j - 1$ opérations de chargement et $n_j - 1$ opérations de livraison. Si les jobs doivent être transportés du dépôt initial à la première machine de leur séquence, alors la séquence du job doit inclure le dépôt.

Chaque paire d'opérations-machines $(O_{i,j},O_{i,j+1})$ représente une requête de transfert $R_{(O_{i,j},O_{i,j+1})}$ du job i qui doit être transporté de la machine $\mu_{i,j}$ à la machine $\mu_{i,j+1}$. Chaque requête de transfert $R_{(O_{i,j},O_{i,j+1})} = \left(P_{i,j}, D_{i,j+1}\right)$ de machine $\mu_{i,j}$ à la machine $\mu_{i,j+1}$ est définie par une opération de chargement $P_{i,j}$ et par une opération de livraison $D_{i,j+1}$. Cette opération de chargement $P_{i,j}$ et cette opération de livraison $D_{i,j+1}$ sont caractérisées par le véhicule r qui leur est affecté et par leurs dates de début respectives. Une requête de transfert est pleinement déterminée par une séquence ordonnée d'opérations réalisant un chemin de $\mu_{i,j}$ à $\mu_{i,j+1}$, où toutes les opérations (de chargement et de livraison) appartenant au chemin sont affectées au même véhicule r. La durée du transfert est définie par le délai entre la date de début de l'opération de chargement $P_{i,j}$ et la date de début de l'opération de livraison $D_{i,j+1}$.

Une disjonction entre deux opérations de type chargement et/ou livraison correspond à une opération de transport, et est modélisée par un arc du sommet qui modélise l'opération de chargement à la machine $\mu_{i,j}$, au sommet qui modélise la livraison à la machine $\mu_{k,p}$, ayant pour valeur $td^{r,c}_{\mu_{i,j},\mu_{k,p}}$ (les deux opérations sont affectées au véhicule r qui est chargé de c jobs). Cette remarque est valable pour toutes les configurations (P,D), (P,P), (D,D), (D,P) où P (pickup) est une opération de chargement et D (delivery) une opération de livraison.

La tournée d'un véhicule est composée d'une séquence ordonnée d'opérations de chargement et d'opérations de livraison satisfaisant les contraintes suivantes : 1) une opération de livraison $D_{i,j+1}$ doit être séquencée après l'opération de chargement $P_{i,j}$; 2) à tout moment, le nombre de jobs dans le véhicule doit être inférieur à la capacité du véhicule. Par exemple, sur la Figure 3, la tournée est : $tr = (P_{i,j}, P_{k,p}, D_{i,j+1}, D_{k,p+1})$, pour un véhicule de capacité deux. Les deux contraintes sont vérifiées.

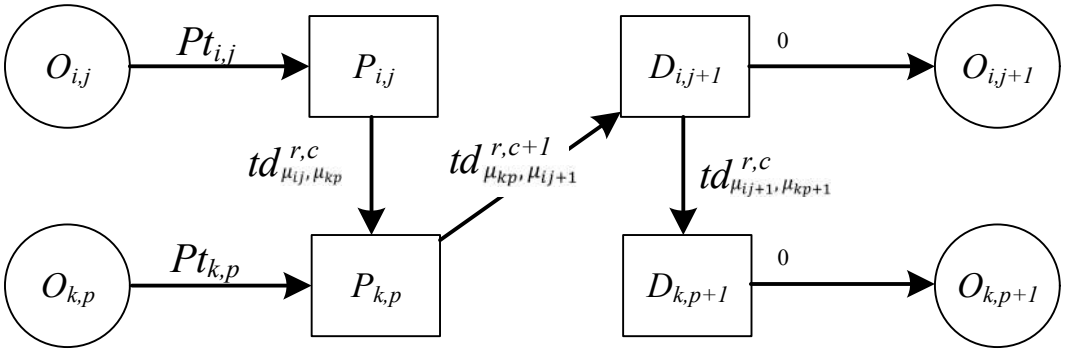


Figure 3 Opérations de chargement et livraison avec un véhicule de capacité non unitaire

2.2. Définition d'une solution

Un graphe disjonctif orienté est obtenu : 1) en affectant un véhicule à chaque opération de chargement et de livraison ; 2) en définissant les disjonctions entre les opérations-machines partageant la même machine ; et 3) en arbitrant les disjonctions entre les opérations affectées au même véhicule. Un graphe acyclique peut être évalué pour calculer les dates de début au plus tôt des opérations, cette évaluation est communément accomplie par un algorithme de plus long chemin, car généralement, seulement des solutions semi-actives sont recherchées (voir chapitre 1). La Figure 4 présente un graphe disjonctif orienté et évalué.

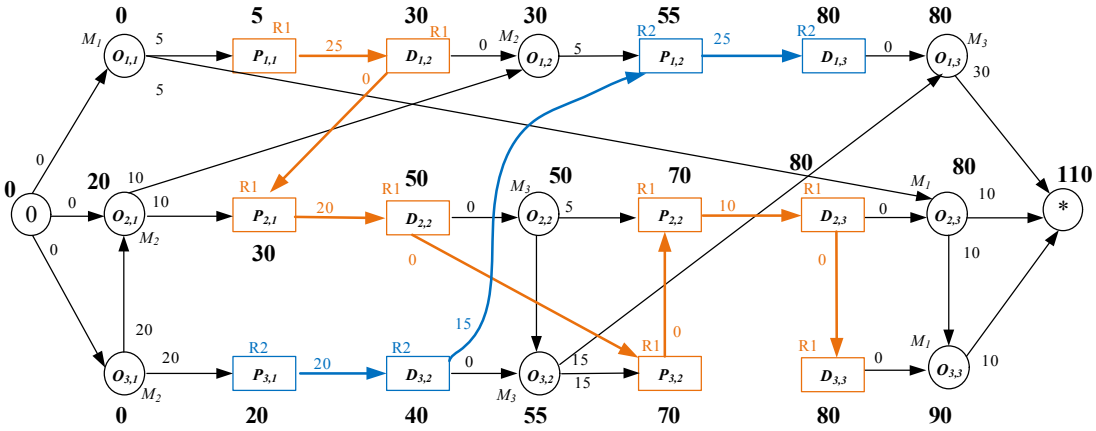


Figure 4 Graphe disjonctif orienté et évalué (une solution du problème)

Le graphe de la Figure 4 illustre un ordonnancement en définissant les dates de début des opérations-machines et en définissant les tournées des deux véhicules (de capacité deux chacun), celles-ci sont composées d'une séquence ordonnée d'opérations de chargement et d'opérations de livraison.

Par exemple, l'opération-machine $O_{1,2}$ est réalisée de la date 30 à la date 35 (Figure 4), cette dernière date n'est pas explicitement modélisée et n'apparaît pas dans le graphe, elle résulte de la somme de la date de début qui est 30 et de la durée de l'opération qui est 5. La date (au plus tôt) de l'opération de chargement $P_{1,2}$ est 55, cette date définit le moment où le véhicule $R2$ charge le job J_1 pour être transporté de sa position courante (sur la machine M_2) à la prochaine machine de la gamme de J_1, c'est-à-dire, la machine M_3. Entre les dates 35 et 55, la pièce occupe le buffer de sortie de la machine M_2, qui est supposé être de capacité illimitée.

Le véhicule $R1$ est affecté au transport du job J_1 de la machine M_1 à la machine M_2 (la durée de transport est de 25 unités de temps) entre les opérations-machines $O_{1,1}$ et $O_{1,2}$. Le véhicule commence l'opération de chargement à la date 5 et l'opération de livraison à la date 30. L'opération suivante affectée au véhicule $R1$ consiste à transporter le job J_2 de la machine M_2 à la machine M_3. Entre l'opération de livraison $D_{1,2}$ (déchargement du job J_1 sur la machine M_2 à la date 30) et l'opération de chargement $P_{2,1}$ du job J_2 sur M_2 (à la date 30), le véhicule $R1$ accomplit un transport à vide entre les machines M_2 et M_2 avec une durée de trajet de 0, ce transport est modélisé par un arc de valeur nulle et correspond en réalité à l'attente du véhicule sur la même machine.

Entre l'opération de livraison $D_{3,2}$ et l'opération de chargement $P_{1,2}$ du job J_1, le véhicule $R1$ réalise un transport à vide de M_3 à M_2 qui est modélisé par un arc de valeur 15 (15 est la durée de transport de M_3 à M_2). Le véhicule $R2$ est affecté au transport de J_3 de M_2 à M_3 puis du transport de J_1 de M_2 à M_3 (cf. Figure 4).

La Figure 5 introduit le diagramme de Gantt du graphe disjonctif orienté et évalué illustré par la Figure 4.

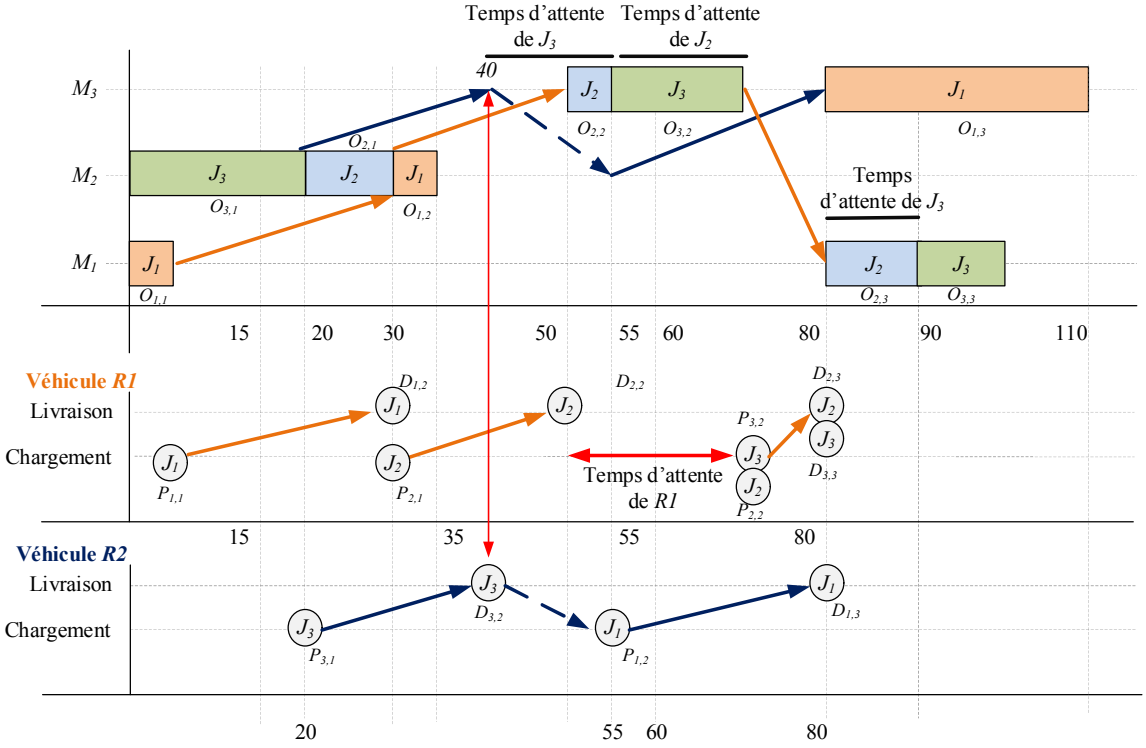


Figure 5 Diagramme de Gantt d'une solution semi-active

La Figure 5 met en évidence les temps d'attente des véhicules et des jobs. Par exemple, le véhicule $R1$ réalise l'opération de livraison $D_{2,2}$ du job J_2 sur la machine M_3 à la date 50 puis, il attend jusqu'à la date 70 pour réaliser l'opération de chargement du job J_3 sur la même machine (M_3), par conséquent, le véhicule $R1$ attend 20 unités de temps.

De manière similaire, le job J_3 est déposé (ou délivré) à la date 40 sur la machine M_3 (l'opération de livraison $D_{3,2}$) par le véhicule $R2$, et il attend jusqu'à la date 55 dans le buffer d'entrée de la machine M_3 avant de commencer son opération-machine $O_{3,2}$. Une telle

situation est induite par l'ordre de passage des jobs sur la machine M_3 (cet ordre est J_2, J_3, J_1), qui est lui-même issu du choix d'arbitrage des disjonctions machines.

Certains jobs peuvent n'avoir aucun temps d'attente sur certaines machines, dans ce cas : 1) la date de début de l'opération-machine est ordonnancée à la fin de la date de livraison ; 2) la date de fin de l'opération-machine correspond à la date de début de l'opération de chargement. Par exemple, le job J_2 finit son premier traitement à la date 30 sur la machine M_2, et il est chargé par l'opération de chargement ($P_{2,1}$) à la date 30, puis transporté de M_2 à M_3 de la date 30 à la date 50 (grâce à l'opération de transport $T_{(O_{2,1},O_{2,2})} = (P_{2,1}, D_{2,2}, R1)$). Ce même job J_2 est déposé à la date 50 (opération de livraison $D_{2,2}$) sur la machine M_3.

2.3. Approche indirecte de résolution

Une solution du Job-shop avec transport est un graphe disjonctif orienté acyclique (cf. Figure 4) dans lequel un algorithme de plus long chemin a permis d'obtenir les dates de début au plus tôt des opérations. L'approche classique de résolution des problèmes de types Job-shop avec transport se décompose en trois étapes (Figure 6) : l'obtention d'un graphe non-orienté qui découle directement des données des problèmes ; l'obtention d'un graphe disjonctif orienté acyclique où toutes les opérations de transport sont affectées à un véhicule et les disjonctions machines arbitrées ; l'obtention d'un graphe disjonctif orienté acyclique et évalué.

Une des difficultés de ces trois étapes est d'obtenir un graphe orienté acyclique, car comme cela a été souligné plusieurs fois, beaucoup de graphes disjonctifs orientés sont cycliques et ne modélisent pas une solution du Job-shop avec transport. Cette difficulté peut être contournée en utilisant une approche indirecte avec l'utilisation d'un vecteur de Bierwirth et d'un vecteur d'affectation qui permettent simultanément d'orienter le graphe de manière acyclique. Ces deux vecteurs définissent des ordres topologiques. Cette représentation du graphe disjonctif orienté et acyclique permet d'évaluer les dates de début au plus tôt des opérations avec un algorithme polynomial (de l'ordre $O(n)$) de plus long chemin de type Dijkstra (Dijkstra, 1959) ou Bellman-Ford (Ford and Lester, 1956; Bellman, 1958; Moore, 1959).

(Lacomme et al., 2013) introduisent une représentation du JSPT basée sur deux vecteurs : le premier est directement adapté du vecteur de (Bierwirth, 1995) pour ordonnancer les opérations, quant au second il permet d'affecter les véhicules aux opérations de transport. Ce type de représentation appartient à la famille des représentations indirectes de type *n-to-1 mapping* selon la classification établie par (Cheng et al., 1996) (voir chapitre 1).

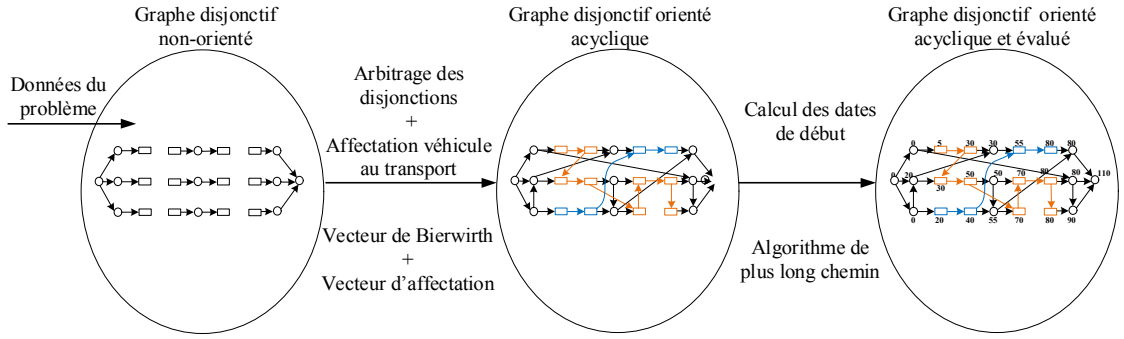


Figure 6 Les trois étapes de résolution

Un vecteur P de Bierwirth est un vecteur par répétition dans lequel chaque job i composé de n_i opérations-machines est répété exactement $3 \times n_j - 2$ fois. Dans le vecteur P, la première occurrence d'un job i réfère à la première opération-machine de ce job, la seconde occurrence réfère à la première opération de chargement du job i, la troisième occurrence fait référence à la première opération de livraison de i, la quatrième occurrence de i est la seconde opération-machine, l'occurrence suivante est une opération de chargement et ainsi de suite. Ce vecteur définit donc un ordre de passage des jobs sur chaque machine. Par exemple, une représentation de la solution de la Figure 4 est introduite par la Figure 7 où $P[1] = 1$ fait référence à l'opération-machine $O_{1,1}$ sur la machine M_1 ; $P[2] = 1$ concerne l'opération de chargement du job J_1 sur la machine M_1 ; et la troisième occurrence de 1 ($P[3] = 1$) modélise l'opération de livraison du job J_1 sur la machine M_2.

Le second vecteur, appelé OA, représente uniquement l'affectation des véhicules aux opérations de chargement, car une opération de livraison $D_{i,j+1}$ est automatiquement affectée au même véhicule que la précédente opération de chargement $P_{i,j}$ puisque les véhicules ne peuvent pas échanger les pièces transportées. Un job chargé par le véhicule r est déchargé par le même véhicule. La première opération de chargement dans le vecteur P est située en $P[2] = 1$ (Figure 7), elle concerne le job J_1 sur la machine M_1 et est assignée au premier véhicule du vecteur OA qui est le véhicule $R1$ ($OA[1] = 1$). La seconde opération de chargement dans le vecteur de Bierwirth est localisée en position 6 ($P[6] = 3$), et est affectée au véhicule $R2$ puisque $OA[2] = 2$.

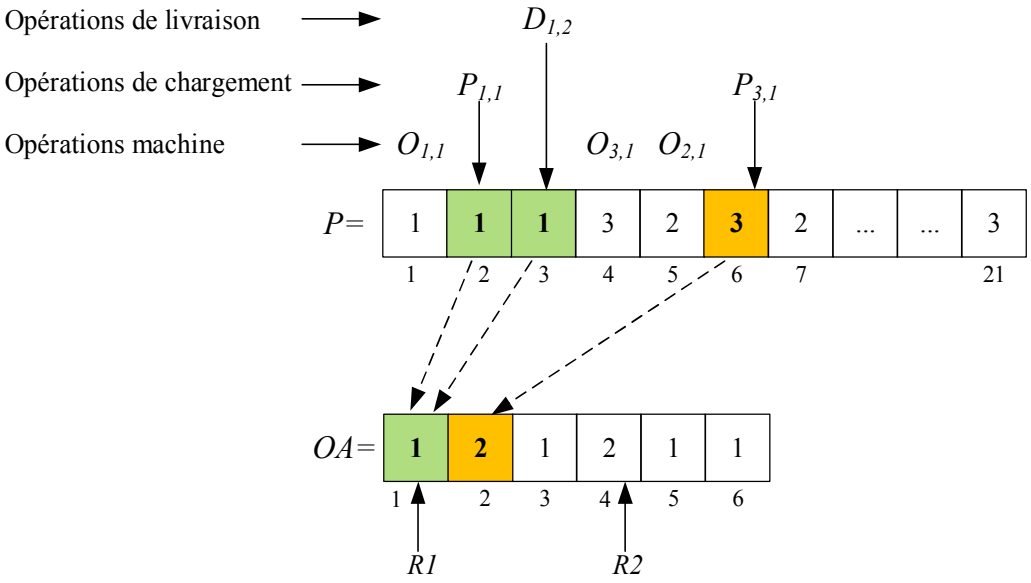


Figure 7 Une représentation de la solution

Les deux vecteurs peuvent être directement utilisés par une approche de résolution indirecte du Job-shop avec routing (Gondran et al., 2018b). La difficulté du JSPR réside dans l'algorithme (ou fonction de décodage) qui permet de passer des vecteurs (ou du graphe disjonctif orienté acyclique) à une solution minimisant le makespan et maximisant la qualité de service simultanément. La fonction d'évaluation TLH (Time-Lag Heuristic) introduite dans ce chapitre est une fonction d'évaluation du graphe disjonctif.

2.4. Qualité de service dans le JSPR

Chaque opération de transport $T_{(O_{i,j},O_{y,x})}=(X_{i,j},Y_{y,x},r)$ d'une machine $\mu_{i,j}$ (qui réalise l'opération-machine $O_{i,j}$) jusqu'à une autre machine $\mu_{y,x}$ (qui réalise l'opération-machine $O_{y,x}$) est composée d'une opération initiale $X_{i,j}$ (qui vaut soit $P_{i,j}$ soit $D_{i,j}$), d'une opération finale $Y_{y,x}$ (qui vaut $P_{y,x}$ ou $D_{y,x}$) et d'un véhicule r affecté aux deux opérations. Par conséquent : $T_{(O_{i,j},O_{y,x})} = (P_{i,j},P_{y,x},r)|(P_{i,j},D_{y,x},r)|(D_{i,j},D_{y,x},r)|(D_{i,j},P_{y,x},r)$. De plus, une opération de transport $T_{(O_{i,j},O_{y,x})}$ est définie par :

- la date de départ $B_{i,j}$ du véhicule r de la machine $\mu_{i,j}$;
- la date d'arrivée $A_{y,x}$ du véhicule r sur la machine $\mu_{y,x}$;
- la véhicule r assigné au transport.

Soient $st_{i,j}$ la date de début de l'opération-machine $O_{i,j}$ et $C_{i,j} = st_{i,j} + Pt_{i,j}$ sa date de fin. Puisqu'une opération de transport $T_{(O_{i,j},O_{y,x})}$ est un couple d'opérations de chargement et de livraison, $B_{i,j}$ est la date de départ du véhicule (date de début de l'opération de transport) (cf. Figure 8) ; et $A_{y,x}$ est la date d'arrivée du véhicule. La date de début au plus tôt d'une opération-machine dépend à la fois de la date d'arrivée du véhicule transportant le job, et de la date de fin de l'opération-machine précédemment ordonnancée sur cette machine.

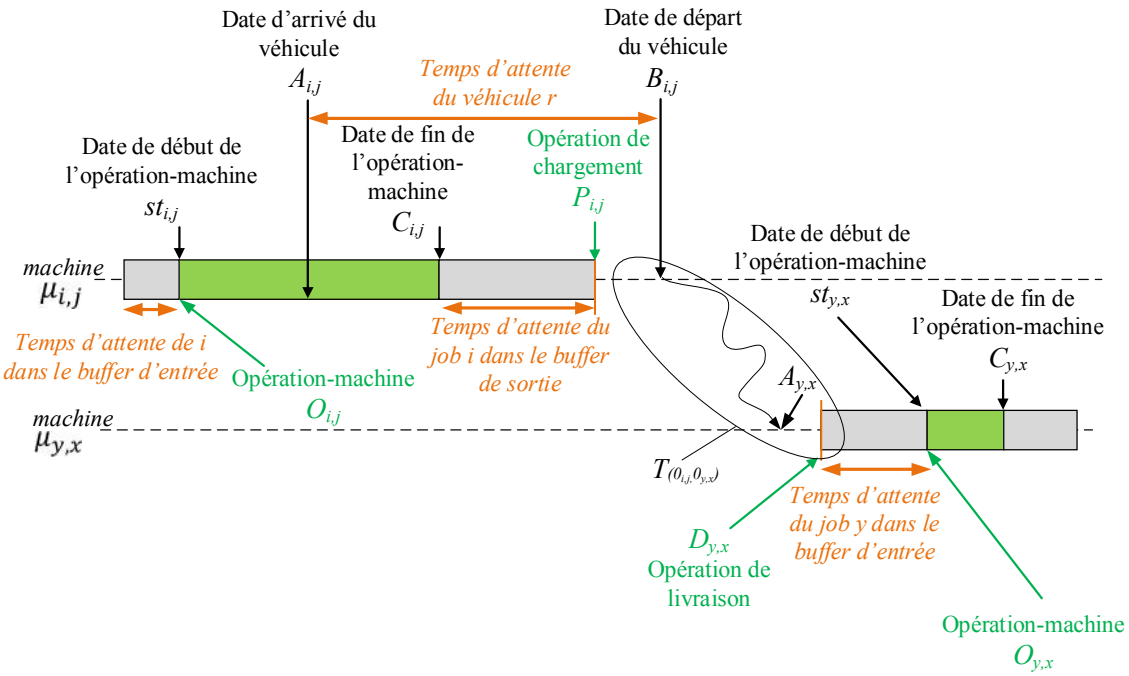


Figure 8 Synchronisation d'une opération de transport et de deux opérations-machines

La différence entre $st_{y,x}$ et $A_{y,x}$ définit le temps d'attente (ou Waiting Time) du job y dans le buffer d'entrée de la machine $\mu_{y,x}$ (Figure 8). De manière similaire, la différence entre $B_{i,j}$ et $C_{i,j}$ définit le temps d'attente du job i dans le buffer de sortie de la machine $\mu_{i,j}$.

Le temps d'attente du véhicule est défini par la différence entre la date d'arrivée $A_{i,j}$ du véhicule sur la machine et la date de départ du véhicule $B_{i,j}$. Il n'y a aucune contrainte qui nécessite que la date de départ $B_{i,j}$ du véhicule soit ordonnancée immédiatement après l'opération de chargement du job, ceci signifie qu'un job peut être chargé et attendre dans le véhicule avant que celui-ci ne commence son déplacement. Des remarques similaires concernent les opérations de livraison $D_{y,x}$ qui ne nécessitent pas d'être commencées immédiatement à la date d'arrivée $A_{y,x}$ du véhicule, ceci signifie que le job pourrait ne pas être autorisé à descendre du véhicule immédiatement à son arrivée. Ces considérations prennent tout leur sens lorsque les problèmes de tournées de véhicules sont soumis à des règlements de sécurité, ce qui est fréquent – notamment – dans les problèmes de transport d'enfants ou de transport de produits dangereux ou précieux. Par exemple, pour un cas de ramassage scolaire, il peut être interdit de déposer un enfant sur son lieu de destination avant la date prévue ou que l'école ouvre. De même, il est interdit à un livreur de déposer sa marchandise devant le dépôt si celui-ci n'est pas encore ouvert.

Afin de réduire les temps d'attente (à la fois des véhicules et des jobs), une approche intégrée doit être privilégiée pour résoudre simultanément les problèmes d'ordonnancement des opérations-machines, de l'affectation des véhicules aux opérations de transport et des tournées de véhicules. Pour les problèmes d'ordonnancement, les dates de début des différentes opérations sont très fréquemment au plus tôt (« left-shifted »), car généralement les solutions recherchées sont semi-actives et la fonction objectif ne prend en compte que le makespan. Ce dernier, aussi dénommé $Cmax$, est la date de début de l'opération fictive finale $*$ dont la date de début est égale à la date de fin de l'opération finissant le plus tard. Le makespan correspond donc au temps nécessaire pour accomplir toutes les opérations :

$Cmax = st_* - st_0$. L'intérêt d'une modélisation explicite des opérations de transport est de pouvoir proposer des dates de début qui ne sont pas au plus tôt. Suivant les dates d'arrivée $A_{i,j}$, de départ $B_{i,j}$ et de début $st_{i,j}$ des opérations-machines, les temps d'attente des véhicules et des jobs sont différents par leur longueur et par leur localisation. Il peut être profitable de retarder la date de début d'une opération (de transport ou machine) afin de réduire des temps d'attente inutiles.

Par analogie avec (Cordeau and Laporte, 2003), il est possible de définir la Qualité de Service (QoS) du Job-shop Scheduling Problem with Routing en considérant le temps nécessaire pour réaliser un job (Total Duration – TD), le temps passé par la pièce dans les transports (Total Riding Time – TRT) et le temps d'attente des pièces dans les buffers d'entrée et de sortie des machines (Total Waiting Time – TWT).

La Figure 9 est une visualisation de ces trois critères, les opérations-machines sont en gris, les opérations de chargement sont en orange et les opérations de livraison sont en vert. Ainsi, le Total Duration (TD) correspond à la notion classique en génie industriel du « flow time » et, il se définit par la différence entre la date de fin de la dernière opération du job et le début de traitement du job. Il s'agit du temps nécessaire pour la gamme complète d'une pièce (modélisée, visuellement, par une ligne dans le graphe disjonctif).

Le Total Riding Time représente le temps passé par le produit (ou le client) entre le moment où il est chargé et celui où il est livré dans le buffer d'entrée de la machine suivante. Cette durée est visuellement représentée sur la Figure 9 (et nommée TRT) par un ensemble constitué par l'ordre suivant : un nœud de chargement (en orange), un nœud de livraison (en vert) et une opération-machine (en gris).

Le Waiting Time TWT d'un job est la somme des temps d'attentes de la pièce dans les buffers d'entrée et de sortie des machines. Sur la Figure 9, ces temps se situent entre les opérations de livraison (en orange) et les opérations-machines (en gris) et entre les opérations-machines et les opérations de chargement (en vert).

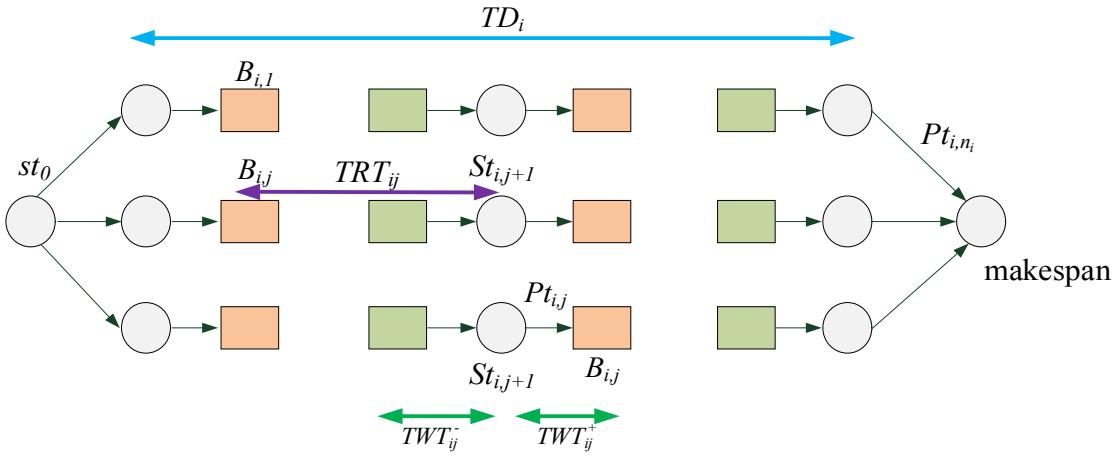


Figure 9 Introduction de nouveaux critères définissant la qualité de service dans le JSPR

La Figure 10, qui est un diagramme de Gantt, présente les différents critères de la qualité de service. Le Total Duration du job J_2 est en bleu, il débute à la première opération de J_2 sur la machine M_2 (opération-machine $O_{2,1}$) et finit à la fin de la dernière opération de J_2 sur la machine M_1 (opération-machine $O_{2,3}$).

Le Riding Time du job J_1 (en violet sur le diagramme de Gantt de la Figure 10) est composé de deux parties, car il y a deux opérations de transport dans la gamme. Le premier Riding Time de J_1 est entre la première opération (opération-machine $O_{1,1}$) sur la machine M_1, ordonnancée à la date 0 et le début de l'opération-machine suivante (opération-machine $O_{1,2}$) commençant à la date 30. La seconde partie du Riding Time de J_1 débute lorsque J_1 est chargé dans le véhicule, à la date 55, et finit lorsque J_1 passe sur la machine M_3 à la date 80. Deux Waiting Times sont illustrés en vert, le premier concerne le Waiting Time de J_3 dans le buffer d'entrée de la machine M_3 à la date 40, la pièce attend jusqu'à la date 55 avant d'être réalisée (opération-machine $O_{3,2}$). Le second Waiting Time illustré est situé dans le buffer de sortie de la machine M_3, le job J_2 finit son passage sur M_3 à la date 55 (opération-machine $O_{2,2}$) et attend l'opération de chargement à la date 70.

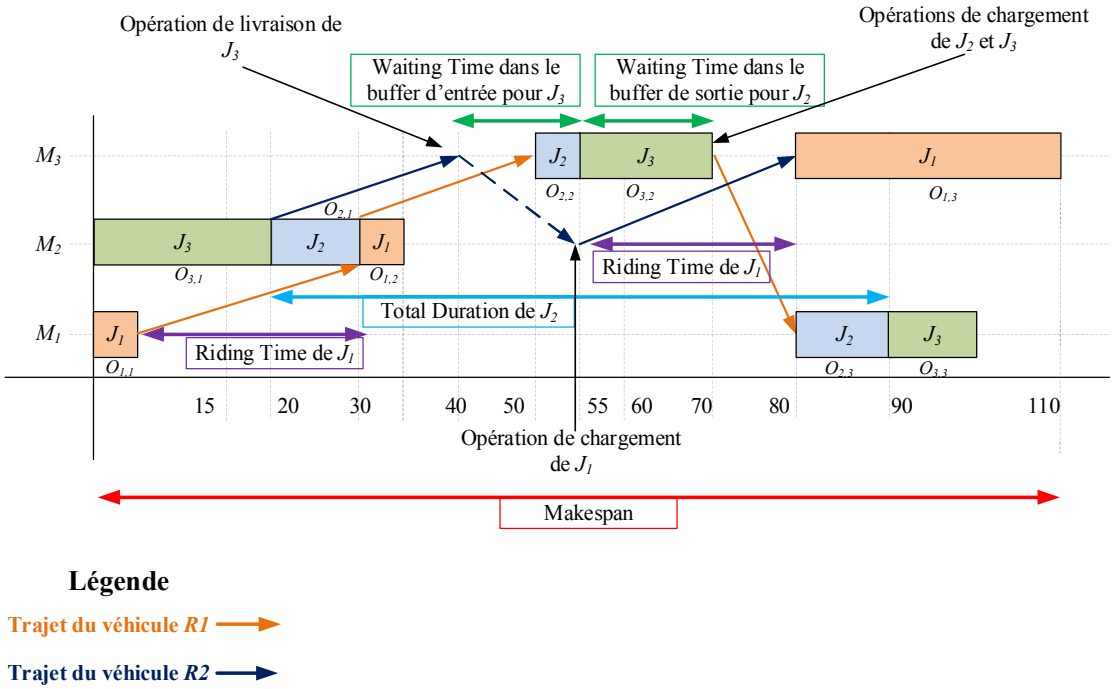


Figure 10 Les critères de la qualité de service dans un diagramme de Gantt

De manière plus formelle, les définitions suivantes sont introduites pour définir la qualité de service (Gondran et al., 2017, 2018a).

- *Total Duration (TD)*. Le Total Duration d'un job i (TD_i) est définit par la différence entre la date de fin de sa dernière opération-machine et la date de début de sa première opération-machine (cette durée est également dénommée « flow time » dans les problèmes d'ordonnancement) : $TD_i = st_{i,n_i} + Pt_{i,n_i} - st_{i,1}$. Le Total Duration TD_i d'un job i est illustré sur la Figure 9. Le Total Duration (TD) est la somme de tous les TD_i : $TD = \sum_{i=1}^{n}(TD_i) = \sum_{i=1}^{n}\left(st_{i,n_i} + Pt_{i,n_i} - st_{i,1}\right)$.
- *Total Riding Time (TRT)*. Un Riding Time est associé à une requête $R_{(O_{i,j},O_{i,j+1})}$ et est calculé par $TRT_{ij} = st_{i,j+1} - B_{i,j}$. Il correspond donc au temps passé entre la date de départ du véhicule r de la machine $\mu_{i,j}$ où le job i est chargé et la date de début du service du job i sur la machine $\mu_{i,j+1}$ (la date de début de l'opération-machine $O_{i,j+1}$), comme le montre la Figure 9. Le Total Riding Time du job i est $TRT_i = \sum_{j \in L_i}(st_{i,j+1} - B_{i,j})$, où L_i

est l'ensemble des opérations de chargement du job i. Le Total Riding Time est la somme de tous les Riding Times de tous les jobs : $TRT = \sum_{i=1}^{n} TRT_i$.

- *Total Waiting Time (TWT).* Le Waiting Time d'un job i dans le buffer de sortie d'une machine $\mu_{i,j}$ est nommé TWT_{ij}^{+} et définit par $B_{i,j} - (st_{i,j} + Pt_{i,j})$. Le Waiting Time d'un job i dans le buffer d'entrée d'une machine $\mu_{i,j}$ est nommé TWT_{ij}^{-} et est défini par $st_{i,j} - A_{i,j}$, (voir la Figure 9). Le Total Waiting Time *(TWT)* est la somme des Waiting Times dans les buffers de sortie $\sum_{i=1}^{i=n} \sum_{j \in L_i} (TWT_{ij}^{+})$ plus les Waiting Times dans les buffers d'entrée $\sum_{i=1}^{i=n} \sum_{j \in U_i} (TWT_{ij}^{-})$ où U_i est l'ensemble des opérations de livraison du job i et L_i est l'ensemble des opérations de chargement du job i. Le Total Waiting Time est la somme de l'ensemble des Waiting Times :
$TWT = \sum_{i=1}^{i=n} \sum_{j \in L_i} (TWT_{ij}^{+}) + \sum_{i=1}^{i=n} \sum_{j \in u_i} (TWT_{ij}^{-})$.

Comme le suggère (Cordeau and Laporte, 2003) pour le DARP, il est possible de définir une fonction objectif F hiérarchique stricte pour le JSPR qui consiste à minimiser d'abord le makespan puis à minimiser en second le TD, suivit du TRT et enfin le TWT : $F = \alpha.C_{max} + \beta.TD + \gamma.TRT + \delta.TWT$ où $(\alpha, \beta, \gamma, \delta)$ respectent les contraintes suivantes :

- $Min(\alpha.C_{max}) > Max(\beta.TD + \gamma.TRT + \delta.TWT)$;
- $Min(\beta.TD) > Max\ (\gamma.TRT)$;
- $Min(\gamma.TRT) > Max\ (\delta.TWT)$.

3. Programme linéaire en nombres entiers pour le JSPR (Gondran et al., 2018a)

Dans cette partie, un Programme Linéaire en Nombres Entiers (PLNE) est introduit pour le JSPR avec une flotte de véhicules à capacité hétérogène (de capacité unitaire ou de capacité non unitaire). La formulation linéaire est une extension de celle introduite par (Larabi, 2010) et contient six ensembles de contraintes :

- Les contraintes disjonctives et conjonctives (1-2) : ces contraintes modélisent à la fois les contraintes de précédence entre les opérations de chargement, de livraison et les opérations-machines dues aux gammes ; et les disjonctions entre les opérations-machines réalisées sur la même machine.
- Des contraintes de type Activity-On-Node Flow (AON-flow) (3-4-5) : la tournée de chaque véhicule est modélisée par un flot. Cet ensemble de contraintes modélise la consommation de ressource des opérations de chargement et de livraison. Les contraintes (3.1) et (3.2) représentent le flot sortant de l'opération fictive initiale 0 et de l'opération fictive finale $*$, tandis que les contraintes (5.1), (5.2), (5.3) et (5.4) représentent la production et la consommation des flots à chaque opération de chargement et de livraison. Les contraintes (4.1) et (4.2) garantissent la faisabilité d'une opération de chargement ou d'une opération de livraison en vérifiant la capacité de véhicule.
- Les contraintes (6-7-8) traduisent les relations entre les contraintes du AON-Flow et les disjonctions des opérations de chargement et de livraison. Ces contraintes définissent les opérations de transport, incluant les dates de départ et d'arrivée des véhicules.
- Les contraintes (9) affectent un véhicule à chaque opération de chargement et de livraison.
- Les contraintes (10) empêchent la formation de sous-tour.
- Les contraintes (11) concernent les critères de QoS.

Puisque les véhicules sont considérés comme des ressources partagées renouvelables, la tournée d'un véhicule peut être formulée comme un RCPSP avec consommation et production de ressource (Artigues et al., 2003), (Carlier et al., 2009), (Artigues, 2017). Le modèle proposé comporte alors autant de flots ϕ^r que de véhicules avec $r = 1 \dots k$ où k est le nombre de véhicules. Pour un véhicule r, sa capacité est notée C^r et la valeur maximale de son flot vaut $C^r + 1$ tandis que sa valeur minimale vaut 1 : un flot $\phi^r = C^r + 1$ signifie que le véhicule se déplace entre deux nœuds du graphe à vide, tandis qu'un flot $\phi^r = 1$ signifie que le véhicule fait un déplacement avec une charge égale à sa capacité maximale. À chaque fois qu'un véhicule visite une opération de chargement, alors la valeur du flot sortant est incrémentée de 1 (le véhicule a déposé un job), tandis qu'à chaque fois qu'il visite une opération de livraison le flot sortant est diminué de 1 (le véhicule a chargé un job).

La Figure 11 est un exemple de flot pour le transport dans un graphe. Le trajet d'un véhicule r est modélisé par les flèches rouges, et il visite les opérations dans l'ordre suivant : 0, 2, 9, 10, 3, 19, 12, 5, 6, 13, 20 et *. La capacité du véhicule est $C^r = 3$, et le flot représentant un transport sans chargement est donc $\phi^r = C^r + 1 = 4$. Le véhicule se déplace sans chargement du sommet 0 (qui est le dépôt de départ) au sommet 2, du sommet 3 au sommet 19, et du sommet 20 au sommet *. Lors de ces déplacements, le flot ϕ^r vaut 4. Lorsque le véhicule arrive au sommet 2, il charge une pièce, et son flot devient $\phi^r = 3$. Ensuite, il charge une nouvelle pièce au sommet 9, et son flot est alors $\phi^r = 2$. Le véhicule continue son trajet en livrant une pièce au sommet 10, et son flot est incrémenté $\phi^r = 3$. En arrivant sur le sommet 3, le véhicule dépose une pièce, le flot devient alors $\phi^r = 4$, signifiant que le véhicule est vide. Entre les sommets 5 et 6, le flot est $\phi^r = 1$: le véhicule transporte 3 pièces et a atteint sa capacité $C^r = 3$.

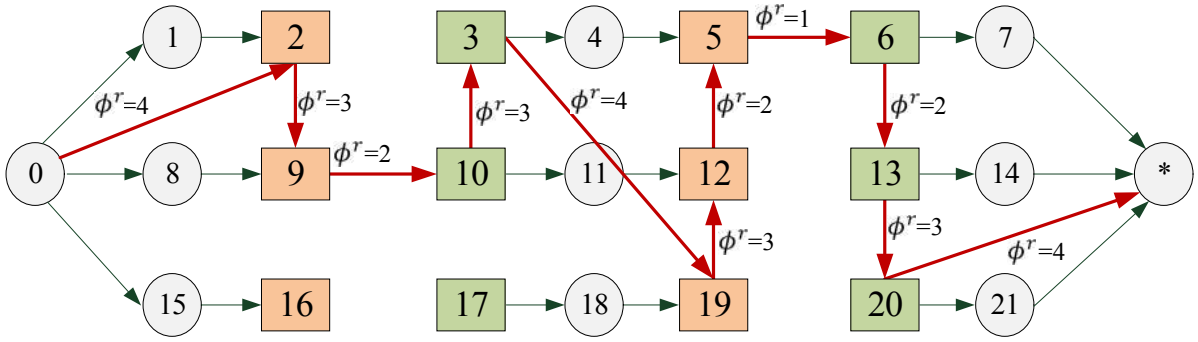


Figure 11 Modélisation du transport par un flot

3.1. Notations

3.1.1. Notations des paramètres

- J : ensemble des jobs ; $J = \{J_1 ; J_2 ; \dots ; J_n\}$.
- M : ensemble des machines ; $M = \{M_1 ; M_2 ; \dots ; M_m\}$.
- R : ensemble des véhicules ; $R = \{R_1 ; R_2 ; \dots ; R_r\}$.
- O_M : ensemble des opérations-machines.
- O_p^1 : première opération du job J_p.
- O_M^{f} : ensemble contenant la première opération-machine de chaque job, $O_M^{\mathrm{f}} = \{O_p^1, \forall p \in J\}$.
- O_M^{l} : ensemble contenant la dernière opération-machine de chaque job.

- O : ensemble des opérations-machines.
- L : ensemble des opérations de chargement.
- L_p : ensemble des opérations de chargement du job J_p.
- U : ensemble des opérations de livraison.
- T : $T = L \cup U$.
- W : $W = O \cup T$.
- Pt_i : durée de traitement de l'opération-machine i.
- μ_i : machine sur laquelle l'opération i doit être exécutée.
- C^r : capacité du véhicule $r \in R$.
- c_i^r : volume du job i dans le véhicule r, dans cette partie $c_i^r = 1, \forall i \in O_M, \forall r \in R$.
- $td_{m_1,m_2}^{r,c}$: durée de déplacement du véhicule r entre les machines m_1 et m_2 chargé de c jobs.
- g_i : job de l'opération i.
- H : constante entière supérieure à 10000.
- ϵ : constante inférieure à 0.01.
- O : l'opération d'origine du graphe.
- $*$: l'opération finale du graphe.

3.1.2. Notations des variables

La formulation linéaire est basée sur trois ensembles de variables binaires, ainsi que sur neuf variables de décision continues :

- $a_i^r = \begin{cases} 1 \text{ si l'opération } i \in T \text{ est affectée au véhicule } r \\ 0 \text{ sinon} \end{cases}$
- $b_{ij} = \begin{cases} 1 \text{ si } i \in O \text{ est réalisée avant } j \in O \\ 0 \text{ sinon} \end{cases}$
- $y_{i,j}^{r,c} = \begin{cases} 1 \text{ si un flot, chargé de } c \text{ jobs, est affecté au véhicule } r \text{ entre } i \in T \text{ et } j \in T \\ 0 \text{ sinon} \end{cases}$
- $\phi_{i,j}^r$: le flot affecté au véhicule r reliant $i \in T$ à $j \in T$.
- st_i : la date de début de l'opération $i \in W$.
- TD_p : le Total Duration du job $p \in J$.
- TD : le Total Duration de l'ensemble des jobs.
- TRT_p : le Total Riding Time du job $p \in J$.
- TRT : le Total Riding Time de l'ensemble des jobs.
- TWT_i^+ : le temps d'attente dans le buffer de sortie de la machine après l'opération-machine $i \in O_M \backslash O_M^{\text{l}}$.
- TWT_i^- : le temps d'attente dans le buffer d'entrée de la machine après l'opération-machine $i \in O_M \backslash O_M^{\text{f}}$.
- TWT : Total Waiting Time de l'ensemble des opérations-machines.

3.2. Les contraintes

Afin de faciliter la lecture du PLNE, toutes les opérations sont numérotées sans distinction entre les opérations-machines, les opérations de chargement et de livraison. Si i est une opération-machine, alors $i-1$ est une opération de livraison et $i+1$ est une opération de chargement. Si i est une opération de livraison alors $i+1$ est une opération-

machine et $i-1$ est une opération de chargement. Et enfin si i est une opération de chargement alors $i+1$ est une opération de livraison et $i-1$ est une opération-machine.

Contraintes conjonctives/disjonctives

Les contraintes conjonctives : ces contraintes garantissent qu'une opération (machine, chargement ou livraison) ne débute pas avant que l'opération précédente du job soit terminée. Ces contraintes assurent que l'ordre des opérations respecte la gamme de chaque job.

$$st_i \geq st_{i-1} + Pt_{i-1} \qquad \forall i \in L \qquad (1.1)$$

$$st_i \geq st_{i-1} \qquad \forall i \in U \qquad (1.2)$$

$$st_i \geq st_{i-1} \qquad \forall i \in O_M \backslash O_M^f \qquad (1.3)$$

Les contraintes disjonctives pour les opérations-machines : ces contraintes garantissent que deux opérations-machines i et j exécutées sur la même machine ne peuvent pas être réalisées en même temps : l'opération-machine i ne peut pas débuter tant que l'opération j n'est pas accomplie et que j ne se situe pas dans le buffer de sortie de la machine (ou inversement).

$$st_i \geq st_j + Pt_j - Hb_{ij} \qquad \forall i \in O_M, \forall j \in O_M \mid \mu_i = \mu_j \qquad (2.1)$$

$$st_j \geq st_i + Pt_i - H(1 - b_{ij}) \qquad \forall i \in O_M, \forall j \in O_M \mid \mu_i = \mu_j \qquad (1.2)$$

$$b_{ij} + b_{ji} = 1 \qquad \forall i \in O_M, \forall j \in O_M \mid \mu_i = \mu_j \qquad (2.3)$$

Les contraintes AON-Flow

Les contraintes d'Activity-On-Node Flow (AON-Flow) sont introduites pour le RCPSP par (Artigues et al., 2003).

Les flots aux extrémités du graphe : ces contraintes définissent le nombre d'unités de chaque ressource r, qui sont transférées de l'opération fictive 0 à une opération de chargement (ou l'opération fictive *) avec la contrainte (3.1) ; et le nombre d'unités de ressources transférées d'une opération de livraison (ou l'opération fictive 0) à l'opération fictive finale * avec la contrainte (3.2).

$$\sum_{i \in L \cup \{*\}} \phi_{0,i}^{r} = C^r + 1 \qquad \forall r \in R \qquad (3.1)$$

$$\sum_{i \in U \cup \{0\}} \phi_{i,*}^{r} = C^r + 1 \qquad \forall r \in R \qquad (3.2)$$

Les contraintes de capacité de véhicule : ces contraintes garantissent que le nombre de jobs transportés par le véhicule ne dépasse pas la capacité du véhicule. Si $a_i^r = 1$ (l'opération de livraison est affectée au véhicule r), la contrainte (4.1) peut se réécrire : $\sum_{j \in E \cup \{0\}} \phi_{i,j}^r \leq C^r + 1 - \epsilon$ impliquant que $\sum_{j \in E \cup \{0\}} \phi_{i,j}^r < C^r + 1$, c'est-à-dire qu'après le passage du véhicule sur un nœud livraison, un job est retiré du véhicule. Si $a_i^r = 1$, alors la contrainte (4.2) s'écrit $\sum_{j \in E \cup \{0\}} \phi_{i,j}^r \geq 1 + \epsilon$, impliquant que $\sum_{j \in E \cup \{0\}} \phi_{i,j}^r > 1$: le nombre de

jobs transportés par le véhicule r ne peut pas excéder la capacité C^r du véhicule. Ces deux contraintes sont illustrées sur la Figure 12.

$$\sum_{j \in E \cup \{0\}} \phi_{i,j}^r \leq C^r + 1 - \epsilon + (1 - a_i^r)H \qquad \forall i \in L, \forall j \in E, \forall r \in R \qquad (4.1)$$

$$\sum_{j \in E \cup \{0\}} \phi_{i,j}^r \geq 1 + \epsilon - (1 - a_i^r)H \qquad \forall i \in U, \forall j \in E, \forall r \in R \qquad (4.2)$$

Les contraintes de production et de consommation de flot : ces contraintes définissent la consommation du flot sur les opérations de chargement (5.1) et (5.2) ou de production de flot sur les opérations de livraison (5.3) et (5.4).

$$\sum_{j \in T} \phi_{i,j}^r \geq \sum_{l \in T \cup \{0\}} \phi_{l,i}^r - c_i^r + (a_i^r - 1)H \qquad \forall i \in L, \forall r \in R \qquad (5.1)$$

$$\sum_{j \in T} \phi_{i,j}^r \leq \sum_{l \in E \cup \{0\}} \phi_{l,i}^r - c_i^r + (1 - a_i^r)H \qquad \forall i \in L, \forall r \in R \qquad (5.2)$$

$$\sum_{j \in T \cup \{*\}} \phi_{i,j}^r \geq \sum_{l \in E} \phi_{l,i}^r + c_i^r + (a_i^r - 1)H \qquad \forall i \in U, \forall r \in R \qquad (5.3)$$

$$\sum_{j \in T \cup \{*\}} \phi_{i,j}^r \leq \sum_{l \in E} \phi_{l,i}^r + c_i^r + (a_i^r - 1)H \qquad \forall i \in U, \forall r \in R \qquad (5.4)$$

Lorsque $a_i^r = 1$ (le véhicule r est affecté à l'opération i), les contraintes (5.1)-(5.4) s'écrivent alors (5.1')-(5.4') :

$$\sum_{j \in T} \phi_{i,j}^r \geq \sum_{l \in T \cup \{0\}} \phi_{l,i}^r - c_i^r \qquad \forall i \in L, \forall r \in R \qquad (5.1')$$

$$\sum_{j \in T} \phi_{i,j}^r \leq \sum_{l \in E \cup \{0\}} \phi_{l,i}^r - c_i^r \qquad \forall i \in L, \forall r \in R \qquad (5.2')$$

$$\sum_{j \in T \cup \{*\}} \phi_{i,j}^r \geq \sum_{l \in E} \phi_{l,i}^r + c_i^r \qquad \forall i \in U, \forall r \in R \qquad (5.3')$$

$$\sum_{j \in T \cup \{*\}} \phi_{i,j}^r \leq \sum_{l \in E} \phi_{l,i}^r + c_i^r \qquad \forall i \in U, \forall r \in R \qquad (5.4')$$

Ces contraintes impliquent que :

- $\sum_{j \in T} \phi_{i,j}^r = \sum_{l \in T \cup \{0\}} \phi_{l,i}^r - c_i^r$ d'après les contraintes (5.1)-(5.2) et imposent que le nombre de jobs transportés par le véhicule décroit d'une unité à chaque opération de livraison.
- $\sum_{j \in T \cup \{*\}} \phi_{i,j}^r = \sum_{l \in E} \phi_{l,i}^r + c_i^r$ d'après les contraintes (5.3)-(5.4) et imposent que le nombre de jobs transportés par le véhicule croît d'une unité à chaque opération de chargement.

Ces contraintes sont représentées graphiquement sur la Figure 12.

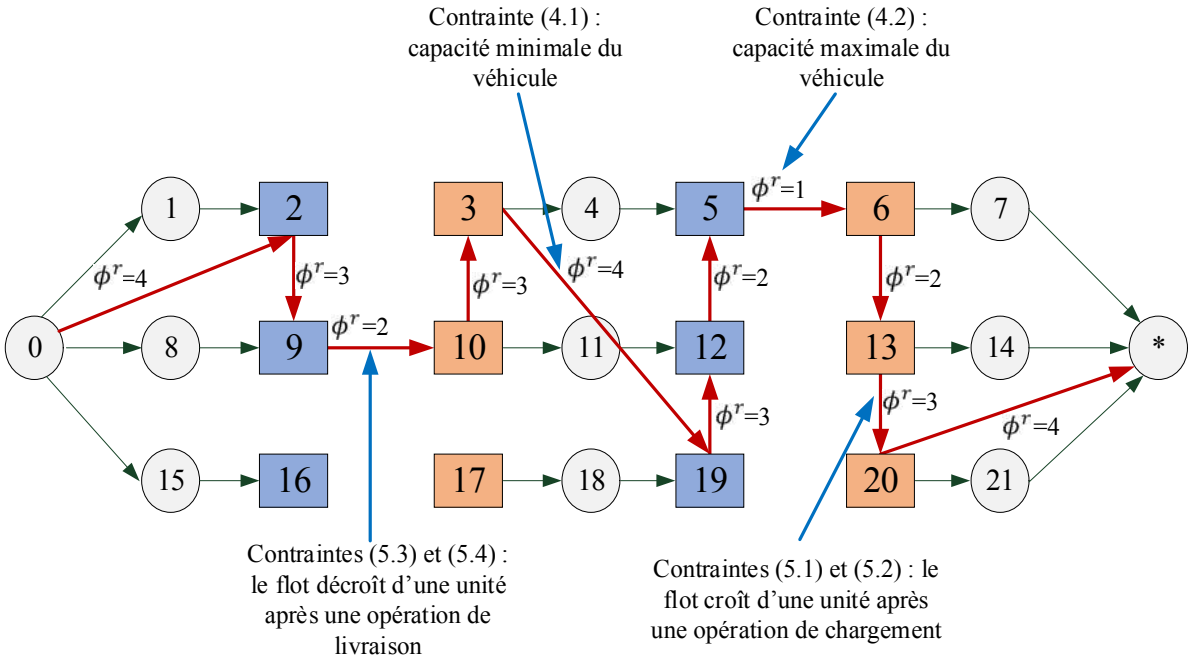


Figure 12 Représentation graphique des contraintes 4 et 5

Relation entre les contraintes de type AON-Flot et les disjonctions de transport

La difficulté des contraintes entre le transport et les dates de début ou entre le transport et l'unicité des opérations de transport sur chaque sommet, est que les contraintes portantes sur les dates de début ou l'unicité du transport utilisent des variables binaires $y_{i,j}^{r,k}$ qui permettent de savoir si un véhicule r réalise une opération de transport de k jobs entre les opérations i et j ; tandis que les contraintes de flots (pour le transport) sont écrites avec des variables entières $\phi_{i,j}^r$. La contrainte (6) permet de passer d'un ensemble de variables à un autre par linéarisation des variables $\phi_{i,j}^r$.

Les opérations de transport : la contrainte est une décomposition du flot $\phi_{i,j}^r$ en variable binaire $y_{i,j}^{r,c}$, elle assure également qu'un flot non nul $\phi_{i,j}^r \in [1\,;\, C^r + 1]$ conduit à une expression unitaire en considérant les variables binaires $y_{i,j}^{r,c}$.

$$\sum_{k=0}^{C^r} (C^r + 1 - k) y_{i,j}^{r,k} = \phi_{i,j}^r \qquad \forall r \in R\,;\, i \in T \cup \{0\,;\ *\};\, j \in T \cup \{0\,; *\} \qquad (6)$$

Par exemple, dans le cas d'un véhicule de capacité 2, la contrainte (6) s'écrit : $3 \times y_{i,j}^{r,0} + 2y_{i,j}^{r,1} + y_{i,j}^{r,2} = \phi_{i,j}^r$. Si $\phi_{i,j}^r = 2$, alors le véhicule r transport un job de l'opération i à l'opération j (car la capacité du véhicule est de 2). $3y_{i,j}^{r,0} + 2y_{i,j}^{r,1} + y_{i,j}^{r,2} = 2$ si et seulement si $y_{i,j}^{r,1} = 1$, $y_{i,j}^{r,0} = 0$ et $y_{i,j}^{r,2} = 0$ car $y_{i,j}^{r,0}$, $y_{i,j}^{r,1}$ et $y_{i,j}^{r,2}$ sont des variables binaires

Unicité des opérations de transport sur les opérations de chargement et livraison : ces contraintes, (7.1) et (7.2), garantissent que pour chaque opération i de chargement ou

livraison, une et une seule opération de transport a pour destination i et une et une seule opération de transport a pour origine i.

$$\sum_{r\in R}\sum_{c=0}^{Cr+1}\sum_{j\in T\cup\{0\}} y_{j,i}^{r,c} = 1 \qquad \forall i \in T \qquad (7.1)$$

$$\sum_{r\in R}\sum_{c=0}^{Cr+1}\sum_{j\in T\cup\{*\}} y_{i,j}^{r,c} = 1 \qquad \forall i \in T \qquad (7.2)$$

Contraintes disjonctives pour les opérations de chargement et livraison : cette contrainte gère les dates de début de deux opérations (de chargement ou livraison) qui se succèdent.

$$st_j \geq st_i + td_{\mu_i,\mu_j}^{r,c} - \left(1 - y_{i,j}^{r,c}\right)H \qquad \forall i,j \in T^2, \forall r \in R, \forall c \in [0\,;\, C^r] \qquad (8)$$

Si $y_{i,j}^{r,c} = 1$, signifiant qu'il y a une opération de transport de l'opération i vers l'opération j, alors l'opération j doit commencer après l'opération i. La contrainte (8) s'écrit : $st_j \geq st_i + td_{\mu_i,\mu_j}^{r,c}$, imposant que l'opération j ne peut pas débuter avant que le véhicule soit arrivé sur l'opération i plus le temps de transport de la machine μ_i à la machine μ_j.

Affectation des véhicules

Unicité de l'affectation des véhicules : un et un seul véhicule doit être affecté à chaque opération de chargement et livraison, contrainte (9.1). De plus, une opération de chargement i doit être affectée au même véhicule que l'opération de livraison $i+1$, contrainte (9.2). Si $a_i^r = 1$, cela signifie que l'opération i est affectée au véhicule r et il existe une opération j telle que $y_{i,j}^{r,c} = 1$ si i est une opération de chargement et 0 sinon (9.3) ; si i est une opération de livraison, alors il existe une opération j telle que $y_{j,i}^{r,c} = 1$: contrainte (9.4).

$$\sum_{r\in R} a_i^r = 1 \qquad \forall i \in T \qquad (9.1)$$

$$a_i^r = a_{i+1}^r \qquad \forall i \in L, \forall r \in R \qquad (9.2)$$

$$a_i^r = \sum_{j\in T\cup\{*\}}\sum_{c=0}^{C_r} y_{i,j}^{r,c} \qquad \forall i \in L, \forall r \in R \qquad (9.3)$$

$$a_i^r = \sum_{j\in T\cup\{0\}}\sum_{c=0}^{C_r} y_{j,i}^{r,c} \qquad \forall i \in U, \forall r \in R \qquad (9.4)$$

Les contraintes de sous-tours

La contrainte (10) empêche la formation de sous-tour en utilisant la contrainte classique d'élimination des sous-tours de Dantzig-Fulkerson-Johnson (Dantzig et al., 1954; Chvátal et al., 2010).

$$\sum_{i \in P_m^{k,i}, j \in P_m^{k,i}} b_{i,j} < |P_m^{k,i}| \qquad \forall P_m^{k,i}\ \forall m \in M, \forall\, k, i \in \mathbb{N}^2 \qquad (10)$$

Où $P_m = \{i \in U | \mu_{i+1} = m\} \cup \{i \in L | \mu_{i-1} = m\}$ $\forall m \in M$, où $P_m^{k,i}$ est le i^{em} sous ensemble contenant exactement k opérations. Dans chaque sous-ensemble $P_m^{k,i}$, les sous-tours sont évités en s'assurant que le nombre d'arcs ($b_{i,j} = 1$) dans $P_m^{k,i}$ est plus petit que le nombre d'opérations contenues dans le sous-ensemble $P_m^{k,i}$.

Les contraintes de Qualité de Service (QoS)

Contraintes pour le Total Duration : le Total Duration d'un job (contrainte 11.1) est la différence entre la date de fin de sa dernière opération-machine et la date de début de sa première opération-machine (voir section 2.4). Le Total Duration (TD) est la somme de l'ensemble des Total Durations (contrainte 11.2).

$$TD_p = st_{n_p} + Pt_{n_p} - st_{O_p^1} \qquad \forall p \in J \qquad (11.1)$$

$$TD = \sum_{\forall p \in J} TD_i \qquad (11.2)$$

Contraintes pour le Total Riding Time : le Riding Time d'un job (contrainte 11.3) est le temps que le job passe dans un véhicule (voir section 2.4). Le Total Riding Time (TRT) est la somme de tous les Riding Times (contrainte 11.4).

$$TRT_p = \sum\nolimits_{j \in L_p} (st_j - st_{j-2}) \qquad \forall p \in J \qquad (11.3)$$

$$TRT = \sum_{\forall p \in J} TRT_i \qquad (11.4)$$

Contraintes pour le Total Waiting Time : TWT_i^+ (contrainte 11.5) est le temps d'attente entre la date de fin de l'opération-machine i et sa date de chargement (réalisé par l'opération $i+1$). TWT_i^- (contrainte 11.6) est le temps d'attente entre la date de fin de livraison correspondant à l'opération $i-1$ et la date de début de l'opération-machine i (voir section 2.4). Le Total Waiting Time (TWT) est la somme de l'ensemble des Waiting Times (contrainte 11.7). La Figure 13 est un rappel de la représentation graphique des critères de la qualité de service pour le JSPR.

$$TWT_i^+ = st_{i+1} - (st_i + Pt_i) \qquad \forall i \in O \backslash O_M^{\text{l}} \qquad (11.5)$$

$$TWT_i^- = st_i - st_{i-1} \qquad \forall i \in O \backslash O_M^{\text{f}} \qquad (11.6)$$

$$TWT = \sum_{i \in O \setminus O_M^1} (TWT_i^+) + \sum_{j \in O \setminus O_M^f} (TWT_i^-) \quad (11.7)$$

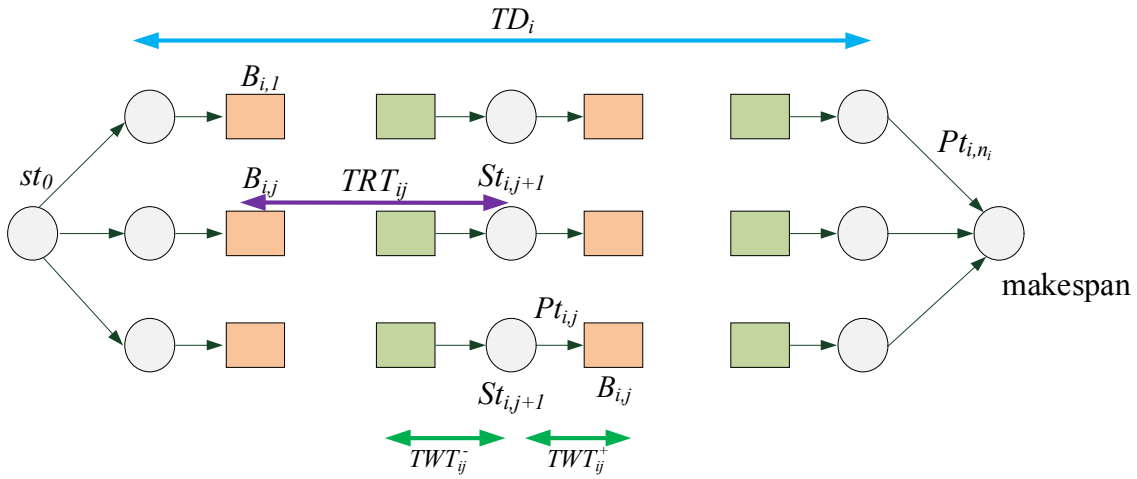


Figure 13 Rappel des critères de la qualité de service pour le JSPR

3.3. La fonction objectif

La fonction objectif concerne à la fois la minimisation du makespan et la maximisation de la qualité de service, ce dernier est défini par $QoS = \frac{1}{\beta.TD+\gamma.TRT+\delta.TWT}$. Par conséquent, la fonction multi-objectifs peut être formulée comme suit (cf. section 2.3) :

$$F(x) = \alpha C_{max} + \beta \frac{1}{QoS} \text{ avec } C_{max} = \max_{i \in J}\left(st_{i,n_i} + pt_{i,n_i}\right)$$

4. Proposition d'une approche de résolution du JSPR (Gondran et al., 2018a)

Les méthodes de résolution approchée du JSPR sont basées sur l'évaluation du graphe disjonctif. Les fonctions d'évaluation classiques et usuelles par un algorithme de plus long chemin, de type Bellmand-Ford (Ford and Lester, 1956; Bellman, 1958; Moore, 1959) ou Dijkstra (Dijkstra, 1959), d'un graphe disjonctif conduisent à une solution semi-active minimisant le makespan où toutes les opérations débutent le plus tôt possible (Roy and Sussmann, 1964). Or l'objectif du JSPR est de simultanément minimiser le makespan et de maximiser la qualité de service (QoS). La prise en compte de ce second critère impose que la solution ne soit pas semi-active.

La proposition originale de cette section est une nouvelle fonction d'évaluation permettant d'obtenir une solution qui n'est pas semi-active pour le JSPR. Cette fonction d'évaluation (TLH – Time-Lag Heuristic) est basée sur celle de (Cordeau and Laporte, 2003) introduite initialement pour un problème de tournées de véhicules (Dial-A-Ride Problem – DARP) et qui est adaptée au JSPR. La première partie de cette section présente l'algorithme de (Cordeau and Laporte, 2003) et comment cet algorithme peut être généralisé en utilisant des time-lags maximaux. La seconde partie expose comment l'algorithme de (Cordeau and

Laporte, 2003), qui est conçu pour des tournées indépendantes les unes des autres, est étendu au JSPR qui possède des tournées dépendantes les unes des autres. Les sections suivantes développent l'impact de l'insertion des time-lags maximaux dans le graphe, la fonction TLH en détail, la recherche de la valeur minimale d'un time-lag maximal, son insertion dans le graphe disjonctif, et enfin la proposition d'un schéma global de résolution incluant la fonction TLH.

4.1. L'algorithme de (Cordeau and Laporte, 2003)

(Cordeau and Laporte, 2003) introduisent une fonction d'évaluation gloutonne pour le Dial-A-Ride Problem (DARP) qui minimise consécutivement la durée totale (Total Duration) de la tournée et la durée du trajet des clients (Riding Time) (voir chapitre 1, section 6.4.2). Cette évaluation permet d'obtenir une solution non semi-active et qui est un compromis entre les différents critères. Il est important de noter que cet algorithme est conçu pour évaluer des tournées indépendantes les unes des autres.

La force de leur approche est qu'à chaque étape de leur fonction d'évaluation, un critère est minimisé et celui-ci ne peut plus être dégradé lors des étapes suivantes. Les critères sont minimisés dans l'ordre du plus englobant au plus particulier, l'algorithme consiste donc par minimiser le Total Duration en premier, puis les Riding Times en second.

L'idée de l'algorithme de (Cordeau and Laporte, 2003) est de planifier les dates de début de service sur les nœuds de chargement au plus tard, et de planifier les dates de début de service sur les nœuds de livraison au plus tôt afin de minimiser la durée de transport des clients et la durée totale de la tournée. Leur algorithme est composé de cinq grandes étapes :

- Étape 1 : évaluation des dates au plus tôt.
- Étape 2 : évaluation des dates au plus tard.
- Étape 3 : évaluation des dates au plus tôt en imposant que la date de début du véhicule de son dépôt de départ soit le plus tard.
- Étape 4 : minimisation des temps de trajets des clients, en parcourant le graphe dans le sens de la tournée.

Le principe de l'algorithme de (Cordeau and Laporte, 2003) est illustré avec la tournée de la Figure 14. Cette tournée est constituée de 5 clients (A, B, C, D et E), que le véhicule doit « visiter » dans cet ordre, le sommet avec un « + » correspond au sommet de départ du client et le sommet avec un « - » correspond au sommet de livraison du client. La tournée commence au dépôt 0, puis le véhicule charge un client au niveau du sommet A+ (celui-ci doit être déchargé au niveau du sommet A-), puis le véhicule charge un nouveau client au niveau du sommet B+ (le sommet B- est le sommet de destination de ce nouveau client), etc., jusqu'au dépôt final *. Une fenêtre de temps est associée à chaque sommet : par exemple, au sommet A+, le client doit être chargé dans la fenêtre de temps [0 ; 20]. Les durées de service sont nulles.

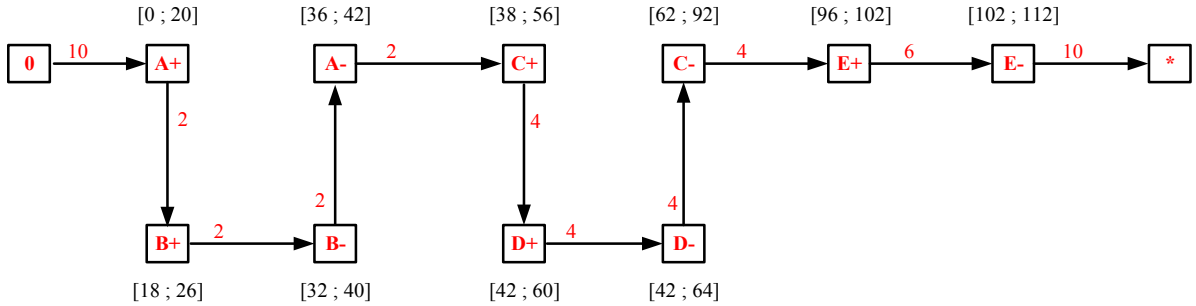


Figure 14 Tournée pour l'algorithme de (Cordeau and Laporte, 2003)

La première étape de l'algorithme de (Cordeau and Laporte, 2003) consiste en une évaluation au plus tôt des dates des visites. Une visite commence lorsque les deux conditions suivantes sont simultanément remplies : le véhicule est arrivé et la fenêtre de temps est atteinte.

Le Tableau 2 présente, pour chaque visite, la date A_i qui est la date d'arrivée du véhicule ; la date st_i qui est la date de début du service ; la date C_i qui est la date de fin du service ; et la date D_i qui est la date de départ du véhicule (voir la section 0). Par exemple, le véhicule part de la visite 0 (qui est le dépôt de départ) à la date $D_i = 0$, ensuite il arrive à la visite A+ à la date $A_i = 10$, il débute son service immédiatement à la date $st_i = 10$, le service est de durée nulle et finit donc à la date $C_i = 10$, finalement le véhicule repart tout de suite après à la date $D_i = 10$. Le véhicule arrive au sommet suivant (B+) à la date $A_i = 12$. La fenêtre de temps du sommet B+ est $[18\,;26]$, le véhicule doit donc attendre la date $st_i = 18$ avant de commencer son service, il finit ensuite celui-ci à la date $C_i = 18$ et part à la date $D_i = 18$.

Tableau 2 Dates des visites après l'évaluation au plus tôt de la tournée

i	Fenêtre de temps	Durée de service	Distance du successeur	A_i	$st_i(ES)$	C_i	D_i
0	/	0	10	/	0	0	0
A+	[0 ; 20]	0	2	10	10	10	10
B+	[18 ; 26]	0	2	12	18	18	18
B-	[32 ; 40]	0	2	20	32	32	32
A-	[36 ; 42]	0	2	34	36	36	36
C+	[38 ; 56]	0	4	38	38	38	38
D+	[42 ; 60]	0	4	42	42	42	42
D-	[42 ; 64]	0	4	46	46	46	46
C-	[62 ; 92]	0	4	50	62	62	62
E+	[96 ; 102]	0	6	66	96	96	96
E-	[102 ; 112]	0	10	102	102	102	102
*	/	0	/	112	112	112	/

La Figure 15 présente la tournée avec les dates de début au plus tôt. (Cordeau and Laporte, 2003) définissent une solution par le vecteur T = (date de fin de la tournée ; date de début de la tournée ; somme des durées de transport ; durée totale de trajet des clients ; durée totale de la tournée). Le coût de la solution ES au plus tôt est : $T(ES) = (112\,;0\,;50\,;(36-10)+(32-18)+(62-38)+(46-42)+(102-96)\,;112) = (112\,;0\,;50\,;74\,;112)$.

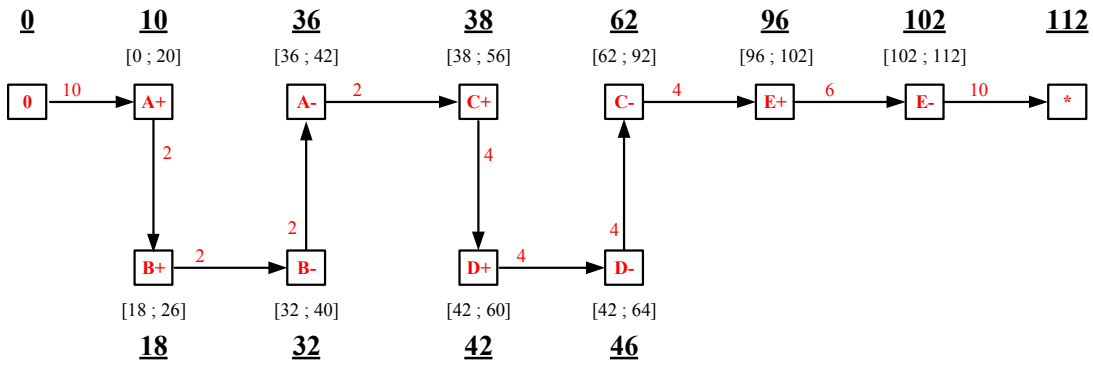


Figure 15 Dates de service au plus tôt

La seconde étape de l'algorithme de (Cordeau and Laporte, 2003) consiste à fixer la date de début au plus tard du sommet * au plus tôt (donc à 112 dans notre exemple), puis d'évaluer les dates des visites au plus tard en remontant la tournée du dépôt final au dépôt de départ.

Le véhicule doit donc partir de la visite E- à la date $D_i = 102$ (car le temps de trajet de E- au dépôt * est de 10 unités de temps). La date de fin de service est $C_i = 102$, et la date de début de service est $st_i = 102$. La Figure 16 illustre les dates de début st_i au plus tôt et au plus tard (ces dernières sont en rouge) des visites. Pour la visite A+, la date de début au plus tôt est $st_i = 10$ et celle au plus tard est $st_i = 20$.

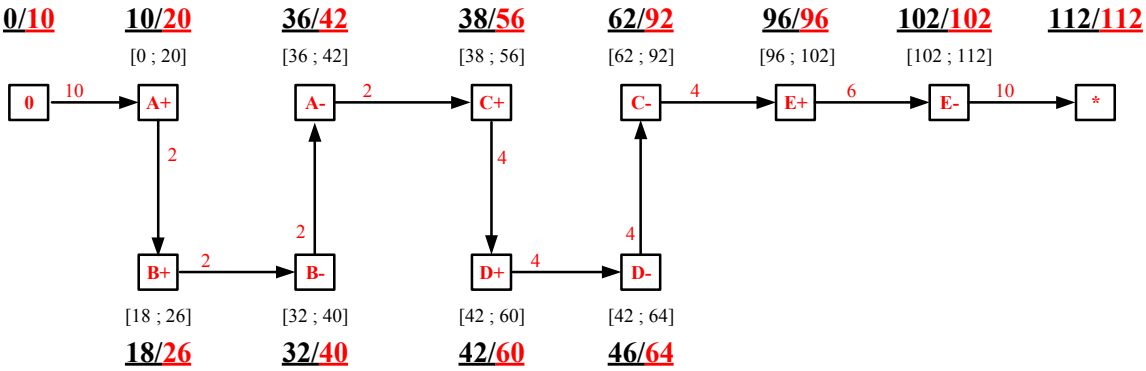


Figure 16 Dates de service au plus tard

Le Tableau 3 propose les dates des visites au plus tard. Le coût (T = (date de fin de la tournée ; date de début de la tournée ; somme des durées de transport ; durée totale de trajet des clients ; durée totale de la tournée)) de la solution LS associée est : $T(LS) = (112 ; 10 ; 50 ; (42 - 20) + (40 - 26) + (92 - 56) + (64 - 60) + (102 - 96) ; 102) = (112 ; 10 ; 50 ; 82 ; 102)$.

Tableau 3 Dates des visites après l'évaluation au plus tard de la tournée

i	Fenêtre de temps	Durée de service	Distance du successeur	A_i	$st_i(LS)$	C_i	D_i
0	/	0	10	/	10	10	10
A+	[0 ; 20]	0	2	20	20	20	20
B+	[18 ; 26]	0	2	22	26	26	26
B-	[32 ; 40]	0	2	28	40	40	40
A-	[36 ; 42]	0	2	42	42	42	42
C+	[38 ; 56]	0	4	44	56	56	56
D+	[42 ; 60]	0	4	60	60	60	60
D-	[42 ; 64]	0	4	64	64	64	64
C-	[62 ; 92]	0	4	68	92	92	92
E+	[96 ; 102]	0	6	96	96	96	96
E-	[102 ; 112]	0	10	102	102	102	102
*	/	0	/	112	112	112	/

Lors de l'étape 3 de l'algorithme de (Cordeau and Laporte, 2003), la date de départ du dépôt D_i est fixée à la valeur de la date obtenue par l'évaluation au plus tard. D'après les deux premières étapes de l'algorithme, la date de fin au plus tôt de la tournée est connue, et ne peut pas être diminuée. Afin de réduire la durée totale du trajet, l'unique possibilité est de démarrer la tournée plus tard. Sur l'exemple, de la Figure 17, la date de début du dépôt est 10. Les dates de début des autres visites sont fixées en fonction de celle-ci.

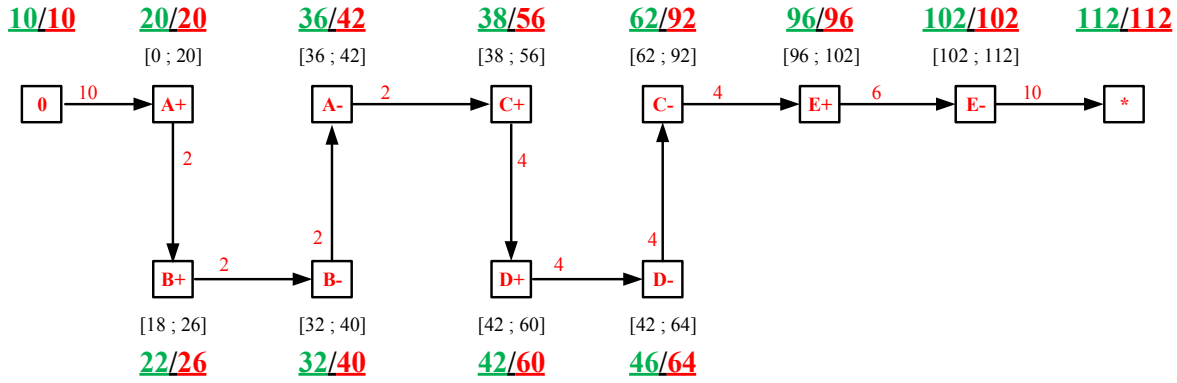


Figure 17 Dates de service lors de la troisième étape

Le Tableau 4 présente les différentes dates de toutes les visites à la fin de la troisième étape de l'algorithme. Le coût de la solution est donné par : $T(3E) = (112\,;10\,;50\,;(36-20)+(32-22)+(62-38)+(46-42)+(102-96)\,;102) = (112\,;10\,;50\,;60\,;102)$.

Tableau 4 Dates des visites après la troisième étape

i	Fenêtre de temps	Durée de service	Distance du successeur	A_i	$st_i(3E)$	C_i	D_i
0	/	0	10	/	10	10	10
A+	[0 ; 20]	0	2	20	20	20	20
B+	[18 ; 26]	0	2	22	22	22	22
B-	[32 ; 40]	0	2	24	32	32	32
A-	[36 ; 42]	0	2	34	36	36	36
C+	[38 ; 56]	0	4	38	38	38	38
D+	[42 ; 60]	0	4	42	42	42	42
D-	[42 ; 64]	0	4	46	46	46	46
C-	[62 ; 92]	0	4	50	62	62	62
E+	[96 ; 102]	0	6	66	96	96	96
E-	[102 ; 112]	0	10	102	102	102	102
*	/	0	/	112	112	112	/

La quatrième étape de l'algorithme de (Cordeau and Laporte, 2003) consiste à minimiser les temps de trajet (Riding Time) des clients en affectant aux sommets de chargement des dates le plus tard possible et aux sommets de livraison des dates le plus tôt possible. Pour cette étape, (Cordeau and Laporte, 2003) calculent la marge et le temps d'attente de chaque visite en fonction de la date au plus tôt et la date au plus tard de début du service de la visite.

La marge est définie pour chaque sommet par $M_i = st_i(LS) - st_i(3E)$. Par exemple, pour le sommet B+, la marge vaut $M_i = 4$. La marge correspond au délai maximal qu'une visite peut être retardée tout en conservant la tournée faisable. Par exemple, pour le sommet B+, d'après le Tableau 5, la date de début $st_i(3E) = 22$, si cette dernière est retardée de plus de 4 unités de temps (car $M_i = 4$), alors la date de début de service de B+ est strictement supérieure à 26 et elle ne respecte pas la fenêtre de temps $[18\,;26]$ de ce sommet. La visite A+ ne peut pas être retardée, et sa marge est donc nulle.

L'attente correspond au temps que passe le client dans le véhicule entre son point de chargement et son point de livraison, lorsque le véhicule est à l'arrêt. L'attente Att_{i-} est donc définie, pour un sommet de chargement $i-$ jusqu'au sommet de livraison $i+$, où $d_{u,v}$ est la distance entre les sommets u et v, par :

$$Att_{i-} = \sum_{j=i-}^{i+} \left(st_{j+1}(3E) - st_j(3E) - d_{j,j+1}\right)$$

Par exemple, pour le sommet A+, la date $st_i(3E)$ est 20, la date sur le sommet B+ est $st_i(3E) = 22$, et il y a 2 unités de temps de transport, donc entre ces deux opérations il n'y a pas d'attente. Par contre, ensuite, la date de départ (équivalente à la date de début de service) du sommet B+ est $st_i(3E) = 22$, et la date de début de service au sommet B- est $st_i(3E) = 32$, le temps de transport est de 2 unités de temps, il y a donc 8 unités de temps d'attente. De manière similaire, entre les sommets B- et A-, il y a 2 unités de temps d'attente, donc le temps d'attente total du sommet A- est $Att_i = 0 + 8 + 2 = 10$.

Le Tableau 5 présente la marge et l'attente de chacun des sommets.

Tableau 5 Calcul de la marge et de l'attente

i	Fenêtre de temps	$st_i(ES)$	$st_i(3E)$	$st_i(LS)$	Marge M_i	Attente Att_i
0	/	0	10	10	/	/
A+	[0 ; 20]	10	20	20	0	10
B+	[18 ; 26]	18	22	26	4	8
B-	[32 ; 40]	32	32	40	8	/
A-	[36 ; 42]	36	36	42	6	/
C+	[38 ; 56]	38	38	56	18	12
D+	[42 ; 60]	42	42	60	18	0
D-	[42 ; 64]	46	46	64	18	/
C-	[62 ; 92]	62	62	92	30	/
E+	[96 ; 102]	96	96	96	0	0
E-	[102 ; 112]	102	102	102	0	/
*	/	112	112	112	/	/

Ensuite, l'algorithme de (Cordeau and Laporte, 2003) parcourt la tournée du dépôt de départ vers le dépôt final et recalcule les dates de début de services $st_i(4E)$ par $st_i(4E) = st_i(3E) + \min(M_i, Att_i)$. La valeur minimale est toujours positive ou nulle. Lorsque $\min(M_i, Att_i)$ est nul, cela signifie qu'il est impossible de retarder la visite i sans violer une contrainte de fenêtre de temps ou d'augmenter la durée totale de la tournée.

Par exemple pour le sommet A+, $\min(M_i, Att_i) = \min(0, 10) = 0$ (d'après le Tableau 5). Il est impossible donc de retarder la date de début de service, donc $st_i(4E) = st_i(3E)$. Pour le sommet B+, $\min(M_i, Att_i) = \min(4, 8) = 4$, il est possible de retarder la date de début de service à $st_i(4E) = st_i(3E) + \min(4, 8) = 22 + 4 = 26$. Les dates des visites suivantes, ainsi que les marges et les attentes doivent être recalculées (Figure 18).

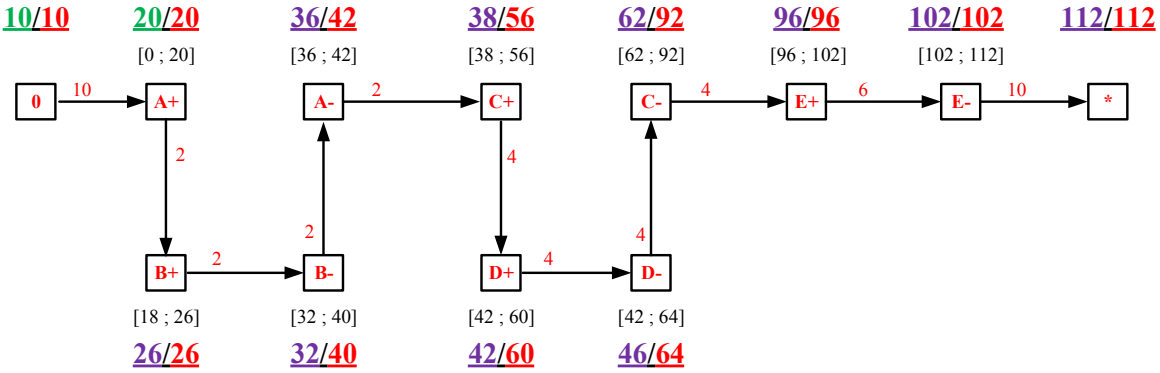


Figure 18 Dates de service au cours de la quatrième étape

Le Tableau 6 présente les nouvelles dates et les valeurs des marges et des attentes lors de l'étape 4 de l'algorithme. Une date avec * signifie que l'algorithme n'a pas encore fixé la date définitive de cette visite. La marge sur le sommet B+ devient nulle avec la nouvelle date $st_i(4E) = 26$.

Tableau 6 Calcul des dates durant la quatrième étape

i	Fenêtre de temps	$st_i(ES)$	$st_i(4E)$	$st_i(LS)$	Marge M_i	Attente Att_i
0	/	0	10	10	/	/
A+	[0 ; 20]	10	20	20	0	10
B+	**[18 ; 26]**	**18**	**26**	**26**	**0**	**4**
B-	[32 ; 40]	32	32*	40	8	/
A-	[36 ; 42]	36	36*	42	6	/
C+	[38 ; 56]	38	38*	56	18	12
D+	[42 ; 60]	42	42*	60	18	0
D-	[42 ; 64]	46	46*	64	18	/
C-	[62 ; 92]	62	62*	92	30	/
E+	[96 ; 102]	96	96*	96	0	0
E-	[102 ; 112]	102	102*	102	0	/
*	/	112	112*	112	/	/

La quatrième étape consiste à parcourir la tournée et à mettre à jour les dates en fonction de la marge et des attentes. La date de la visite C+ peut être retardée car celle-ci a une marge et une attente strictement positives.

Le Tableau 7 présente les dates de début de service à la fin de la quatrième étape. Pour toutes les visites, soit la marge soit l'attente est nulle.

Tableau 7 Date de début de service à la fin de la quatrième étape

i	Fenêtre de temps	$st_i(ES)$	$st_i(4E)$	$st_i(LS)$	Marge M_i	Attente Att_i
0	/	0	10	10	/	/
A+	[0 ; 20]	10	20	20	0	10
B+	[18 ; 26]	18	26	26	0	4
B-	[32 ; 40]	32	32	40	8	/
A-	[36 ; 42]	36	36	42	6	/
C+	[38 ; 56]	38	50	56	6	0
D+	[42 ; 60]	42	54	60	6	0
D-	[42 ; 64]	46	58	64	6	/
C-	[62 ; 92]	62	62	92	30	/
E+	[96 ; 102]	96	96	96	0	0
E-	[102 ; 112]	102	102	102	0	/
*	/	112	112	112	/	/

La Figure 19 présente les dates de service finales de la tournée. Le coût de cette solution ($T =$ (date de fin de la tournée ; date de début de la tournée ; somme des durées de transport ; durée totale de trajet des clients ; durée totale de la tournée)) est $T(4E) = (112 ; 10 ; 50 ; (36 - 20) + (32 - 26) + (62 - 50) + (58 - 54) + (102 - 96) ; 102) = (112 ; 10 ; 50 ; 44 ; 102)$.

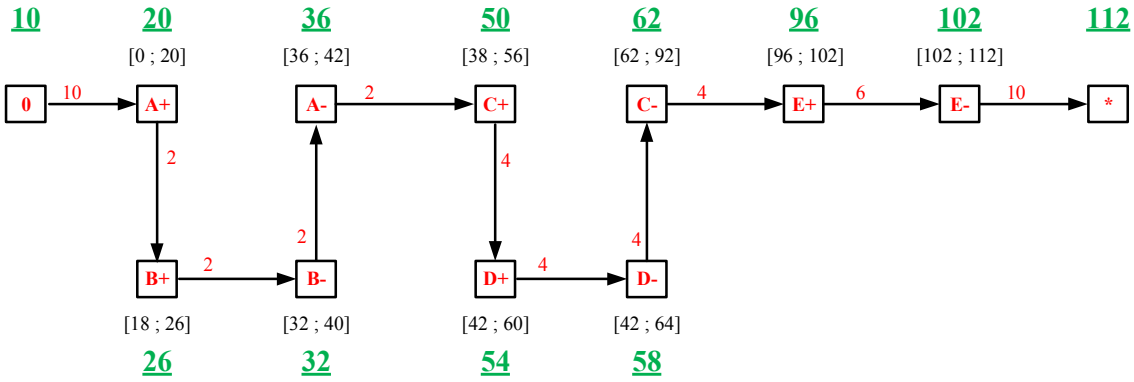


Figure 19 Dates de service finales

Le Tableau 8 résume l'évolution des critères du coût de la solution à chaque étape de l'algorithme. Les valeurs en gras correspondent au critère minimisé durant l'étape. Il est important de noter qu'un critère minimisé durant une étape ne peut pas être dégradé durant les étapes suivantes.

Tableau 8 Évolution des critères suivant l'étape de l'algorithme

		Date de fin de la tournée	**Date de début de la tournée**	**Somme des durées de transport**	**Durée totale de trajet des clients**	**Durée totale de la tournée**
1	Calcul des dates au plus tôt $st_i(ES)$	112	0	50	74	112
2	Calcul des dates au plus tard $st_i(LS)$	112	10	50	82	102
3	Calcul des dates $st_i(3E)$	112	10	50	60	**102**
4	Calcul des dates $st_i(4E)$	112	10	50	**44**	102

Dans ce manuscrit, l'algorithme de (Cordeau and Laporte, 2003) est généralisé en utilisant le concept de time-lag maximal entre les visites. En effet, le décalage de la date de début de service de la visite B+ se modélise dans le graphe de la tournée par l'insertion d'un time-lag maximal entre les visites B- et B+ de valeur 6 lors de la troisième étape de la procédure. Un time-lag maximal (ou time-lag max) est une contrainte de délai maximal entre deux visites (Caumond et al., 2008; Artigues et al., 2011). Le time-lag maximal est modélisé dans un graphe par un arc de valeur négatif.

La Figure 20 correspond à la tournée après la mise à jour des dates de service et l'insertion d'un time-lag entre de la visite B+ à la visite B-. Le time-lag empêche que les dates de début des visites « s'éloignent » de nouveau.

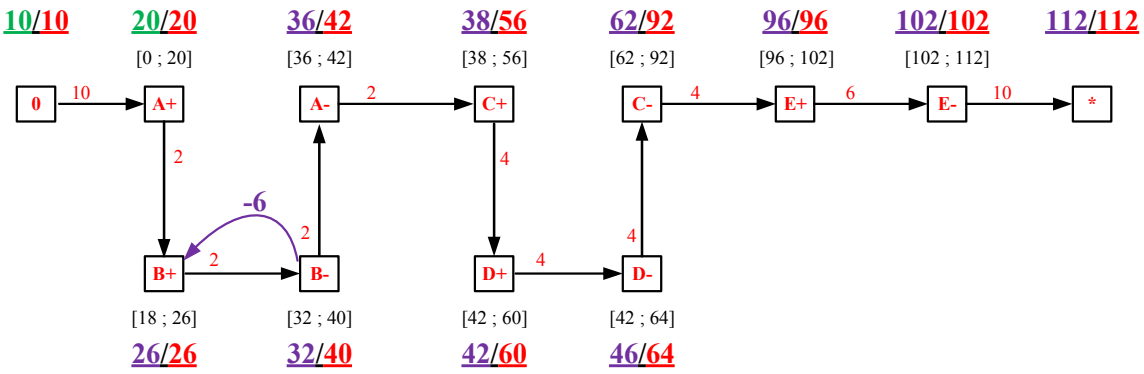


Figure 20 Dates de service et insertion d'un time-lag max au cours de la quatrième étape

L'algorithme de (Cordeau and Laporte, 2003) n'utilise pas explicitement le concept de time-lag maximal car les tournées sont indépendantes les unes des autres. Pour le JSPR, les tournées sont interconnectées et l'utilisation de time-lags maximaux est primordiale afin d'éviter qu'un critère minimisé pour une tournée soit dégradé lors de la minimisation d'un autre critère ou d'une autre tournée.

4.2. Principe de l'évaluation TLH

Il est impossible d'appliquer directement l'approche de (Cordeau and Laporte, 2003) aux tournées présentes dans un graphe disjonctif pour le JSPR car les tournées ne sont pas indépendantes les unes des autres. L'algorithme de (Cordeau and Laporte, 2003) est dédié à l'évaluation d'une tournée unique et indépendante.

Pour le JSPR, il faut concevoir une procédure d'évaluation qui prend en compte simultanément l'ensemble du graphe qui contient toutes les tournées. La dépendance entre les tournées est liée au fait qu'un véhicule réalise des opérations de transport appartenant à des jobs différents ce qui rajoute des dépendances temporelles entre les gammes des jobs. Ce point particulier est illustré sur la Figure 21, où le véhicule $R1$, dont la tournée est modélisée par les arcs en orange, réalise, par exemple une opération de transport pour le job J_1, puis pour le job J_2.

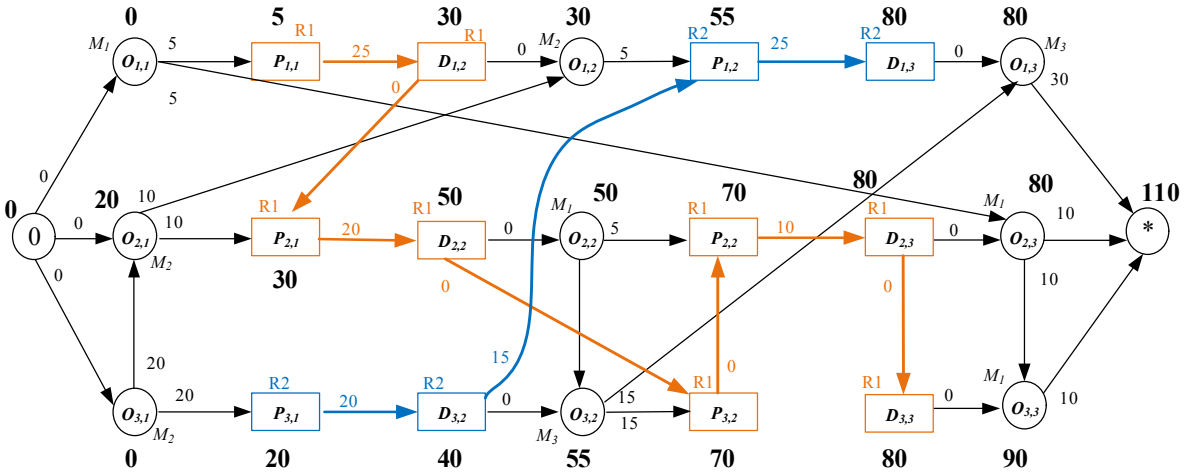


Figure 21 Dépendance des tournées

La nouvelle fonction d'évaluation, proposée dans ce manuscrit, *TLH* – Time-Lag insertion Heuristic – repose sur le même principe que l'approche de (Cordeau and Laporte, 2003) et est basée sur l'insertion heuristique de time-lags dans le graphe disjonctif du JSPT. La fonction d'évaluation TLH minimise le makespan – en calculant les dates de début au plus tôt des opérations – puis TLH minimise le Total Duration (TD), ensuite le Total Riding Time (TRT) et enfin le Total Waiting Time (TWT) (voir définitions section 3.3). La détérioration d'un critère précédemment optimisé est empêchée par l'ajout itératif de time-lags maximaux dans le graphe. La fonction d'évaluation TLH est composée de quatre étapes :

- Étape 1 : minimisation du makespan.
- Étape 2 : minimisation du Total Duration.
- Étape 3 : minimisation du Total Riding Time.
- Étape 4 : minimisation du Total Waiting Time.

Les étapes de l'évaluation TLH vont des time-lags les plus englobants aux time-lags les plus « petits », car insérer les time-lags les plus englobants permet de contraindre les futurs time-lags. Par ailleurs, insérer au début de la procédure des time-lags englobant peu d'opérations risque de créer une croissance des autres critères, car ceux-ci ne seront pas contraints. Par exemple, la réduction des Riding Times peut entrainer un accroissement des Total Durations des jobs, en effet, afin de minimiser les Riding Times, les jobs ne seront chargés que lorsque la machine suivante est libre et seront réalisés sans attente. Dans ce cas, le transport devient la ressource critique.

4.3. Impact de l'insertion des time-lags maximaux dans un graphe

L'évaluation d'un graphe orienté (l'ordre des opérations est connu) affecte une date de début st_i à chaque opération i avec $st_i \in [ES_i\ ;\ LS_i]$, où ES_i est la date de début au plus tôt de i et LS_i est sa date de début au plus tard. Soit LP_{ij} le plus long chemin entre deux opérations i et j et sa taille est donnée par $|LP_{ij}| = LS_j - ES_i$; soit SP_{ij} le chemin le plus court théorique et sa longueur est $|SP_{ij}| = ES_j - LS_i$. Il est important de noter que le chemin SP_{ij} peut être impossible dans le graphe à cause des contraintes disjonctives ou alors, il peut conduire à l'accroissement du makespan. Soit l_{ij} le chemin entre deux opérations i et j, tel que $l_{ij} = st_j - st_i$ avec $l_{ij} \in [SP_{ij}\ ;\ LS_{ij}]$. Une évaluation dite classique d'un graphe avec un algorithme de type Bellman-Ford donne $st_i = ES_i$ et $st_j = ES_j$. Le chemin de Bellman-Ford BFP_{ij} (*Bellman-Ford Path*) est alors défini de l'opération i à l'opération j par $BFP_{ij} = ES_j - ES_i$ et $BFP_{ij} \in [SP_{ij}\ ;\ LP_{ij}]$. Puisque SP_{ij} est une borne inférieure théorique, il existe une borne inférieure minimale $SPM_{ij} \in [SP_{ij}\ ;\ BFP_{ij}]$ telle que $PM_{ij} = st_j - st_i$ soit le plus court chemin possible entre i et j qui maintient toutes les contraintes disjonctives sans accroître le makespan. La minimisation de la qualité de service est équivalente à trouver st_i et st_j qui minimise l_{ij}, c'est-à-dire tel que $l_{ij} = SPM_{ij}$. Cette minimisation est réalisée par l'insertion de time-lags dans le graphe, et est un des points clés de l'évaluation TLH.

La Figure 21 illustre les conséquences de l'insertion d'un time-lag maximal entre deux opérations qui sont ordonnancées au plus tôt ($st_i = ES_i$ et $st_j = ES_j$). Le chemin initial l_{ij}, qui est également le chemin de Bellman-Ford BFP_{ij}, a pour valeur $l_{ij} = BFP_{ij} = st_j - st_i$ (partie 1 de la Figure 21). L'ajout d'un time-lag maximal de valeur $-TL_{ji}$ induit un décalage à droite de i d'une valeur δ (avec $\delta = Max(0\,;st_j - TL_{ji} - st_i)$), signifiant que la date de début de i doit être mise à jour. Si la valeur $l_{ij} = st_j - st_i$ du chemin de i à j est plus petite

que TL_{ji} ($\delta = 0$), la différence entre les deux dates st_i et st_j respecte le time-lag TL_{ji} et aucune mise à jour de st_i n'est requise. Une mise à jour de st_i est nécessaire lorsque le time-lag maximal est plus petit que l_{ij}, partie 2 de la Figure 21, où $st_i + TL_{ji} < st_j$. Par conséquent, si le graphe reste acyclique, alors la différence entre la date de début de j (et, par conséquent, le date de fin) et la date de début de i est réduite. Afin de trouver la valeur optimale SPM_{ij}, une heuristique gloutonne est utilisée lors des étapes de l'évaluation TLH.

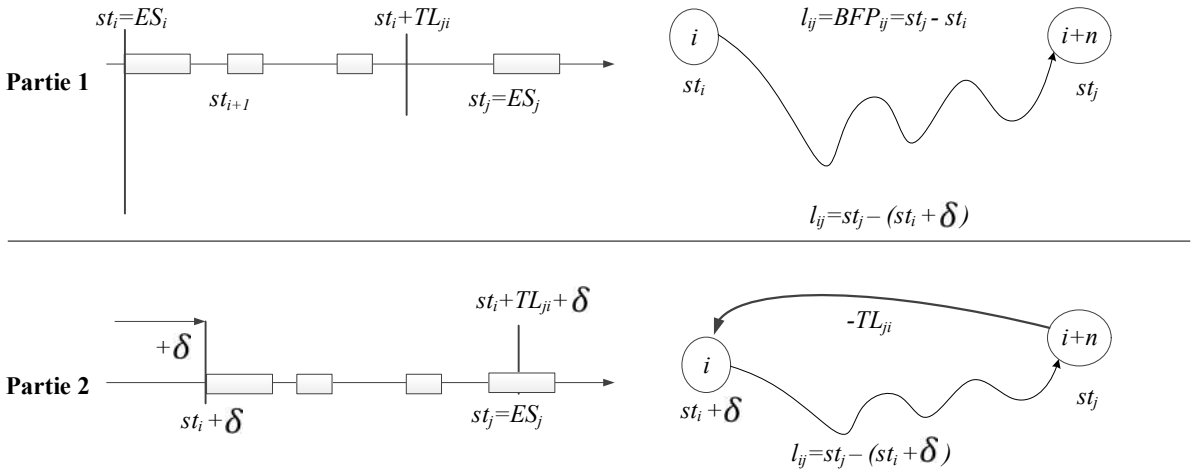


Figure 22 Le délai entre deux opérations est réduit grâce au time-lag maximal

La présence d'arcs pondérés négativement dans un graphe rend impossible son évaluation par un algorithme de complexité linéaire. Il est important de garder à l'esprit que les time-lags maximaux sont des outils pour décaler des dates sachant que celles-ci restent toujours calculées par un algorithme de plus long chemin.

4.4. La fonction d'évaluation TLH

L'évaluation TLH – Time-Lag Heuristic – comporte quatre étapes, chacune permettant la minimisation d'un critère (dans l'ordre, makespan, TD, TRT, et TWT) par insertion de time-lags maximaux dans le graphe. La Figure 23 représente le schéma global de la fonction d'évaluation TLH avec ses quatre étapes.

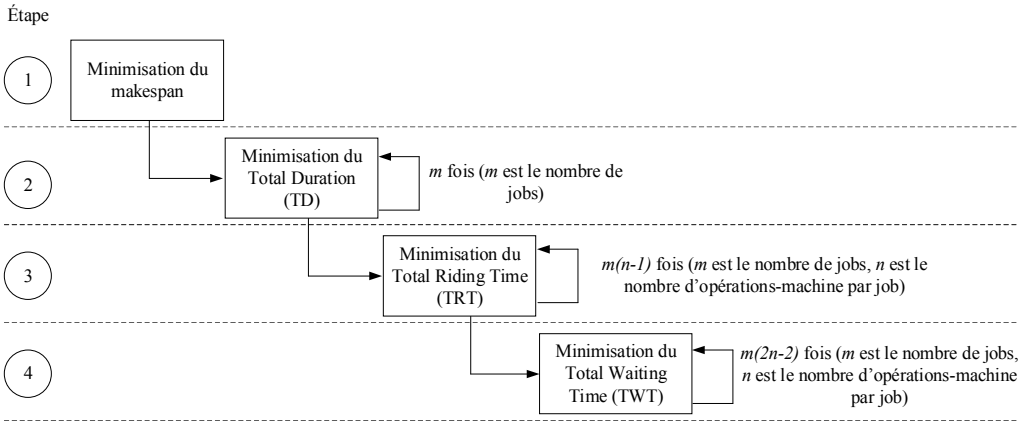


Figure 23 Principe de l'évaluation TLH

La Figure 24 illustre – sur un graphe simplifié – quels time-lags sont ajoutés à chaque étape de l'évaluation TLH. Chaque étape est détaillée par la suite.

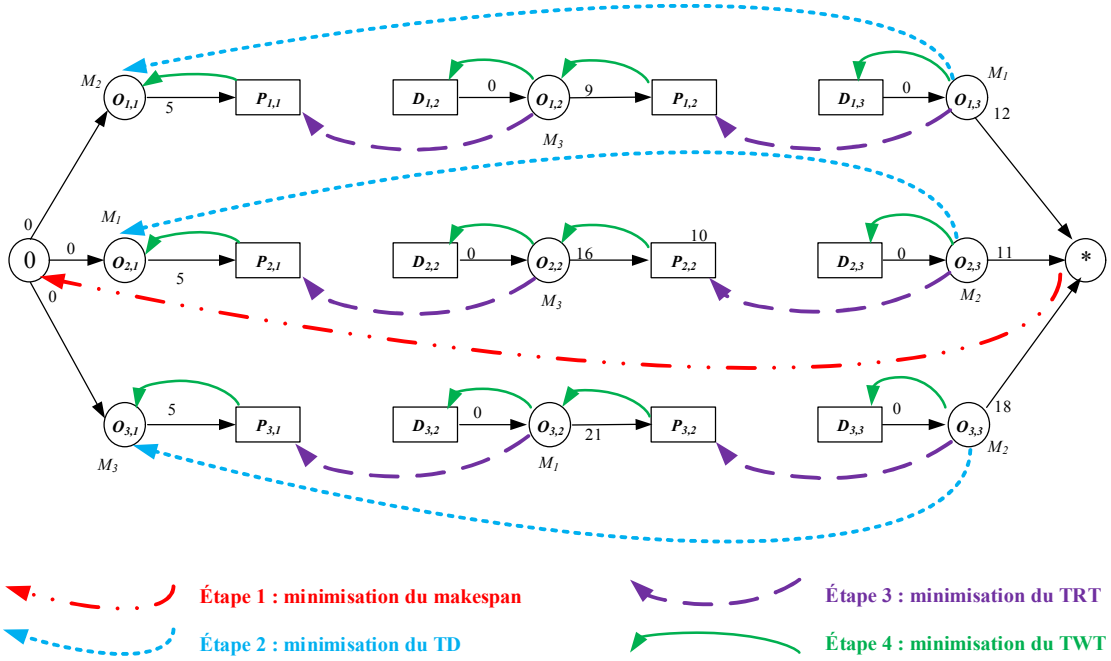


Figure 24 Ordre de minimisation des critères durant TLH

La première étape de l'algorithme TLH évalue les dates de début au plus tôt de chaque opération grâce à un algorithme de plus long chemin de type Bellman-Ford, définissant ainsi une solution semi-active et optimale pour le critère du makespan (voir la Figure 25 où les dates de début au plus tôt ES_i de chaque opération i sont en gras). Les valeurs des différents critères sont données à titre indicatif sur la Figure 25.

Le makespan (ou $Cmax$) est la différence entre la date de l'opération * et la date de l'opération 0. Sur la Figure 25, le makespan est $Cmax = 75 - 0 = 75$.

Le Total Duration (TD_i) d'un job est la différence entre la date de début de sa première opération et la date de fin de sa dernière opération. Sur la Figure 25, le Total Duration du job J_1 (première ligne) est donc $TD_{J_1} = \left(st_{O_{1,3}} + Pt_{O_{1,3}}\right) - st_{O_{1,1}} = (58 + 12) - 0 = 70$. Le Total Duration ($TD$) est la somme de toutes les Total Durations des jobs : $TD = D_{J_1} + D_{J_2} + D_{J_3} = 70 + 75 + 64 = 209$.

Le Riding Time d'une opération-machine (RT) est la différence entre la date d'une opération de chargement type $P_{i,j-1}$ et de la date de l'opération-machine $O_{i,j+1}$. Par exemple, le premier Riding Time du job J_1 (Figure 25) correspond à la différence entre la date de $O_{1,2}$ et celle de $P_{1,1}$, $RT_{J_1}^1 = 23 - 15 = 8$. Le second Riding Time du job J_1 est entre les opérations $O_{1,3}$ et $P_{1,2}$, $RT_{J1}^2 = 58 - 34 = 24$. Le Riding Time du job J_1 est donc $RT_{J_1} = RT_{J_1}^1 + RT_{J_1}^2 = 19 + 24 = 43$. Le Riding Time Total ($TRT$) est la somme de tous les Riding Times, dans l'ordre des jobs, cela donne : $TRT = RT_{J_1}^1 + RT_{J_1}^2 + RT_{J_2}^1 + RT_{J_2}^2 + RT_{J_3}^1 + RT_{J_3}^2 = 8 + 24 + 27 + 16 + 4 + 16 = 95$.

Le Waiting Time sur un nœud de chargement ($WT^{+}_{P_{i,j}}$) est la différence entre sa date de début et la date de fin de l'opération-machine ($O_{i,j}$) précédente. Par exemple, sur la Figure 25, le Waiting Time $WT^{+}_{P_{1,2}} = st_{P_{1,2}} - \left(st_{O_{1,2}} + Pt_{O_{1,2}}\right) = 34 - (23 + 9) = 2$. Le Waiting Time sur un nœud de livraison ($WT^{-}_{D_{i,j}}$) est la différence entre la date de livraison et la date de début de l'opération-machine ($O_{i,j}$) suivante. Par exemple (Figure 25), le Waiting Time $WT^{-}_{D_{1,3}} = st_{O_{1,3}} - \left(st_{D_{1,3}} + Pt_{D_{1,3}}\right) = 58 - (58 + 0) = 0$. Le Total Waiting Time est la somme de tous les Waiting Times de tous les jobs sur toutes les opérations de chargement et de livraison. Pour l'exemple de la Figure 25, $TWT = 35$.

Le coût de la solution de la Figure 25 est résumé par $S = (75\,;209\,;95\,;35)$.

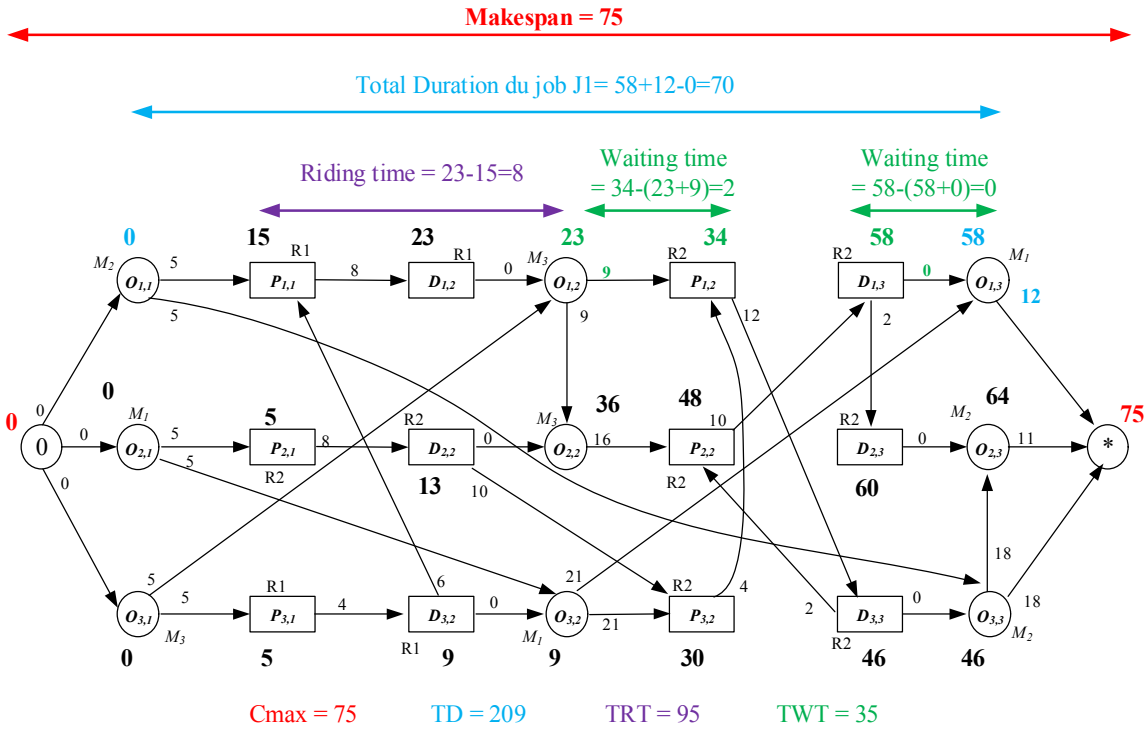


Figure 25 Valeurs des critères après l'évaluation au plus tôt

À la fin de la première étape de l'évaluation TLH, un time-lag maximal est inséré de l'opération fictive finale $*$ à l'opération fictive initiale 0 afin d'éviter une croissance du makespan durant les futures étapes d'optimisation (time-lag en rouge sur la Figure 24 et la Figure 26). Ce time-lag maximal est modélisé par un arc avec une valeur négative égale au makespan. La Figure 26 illustre l'insertion de ce time-lag, les dates de début au plus tôt ES_i et les dates de début au plus tard LS_i de chaque opération i sont en gras.

Cette étape n'aboutit à la minimisation d'aucun critère, mais elle revient à ajouter une contrainte au problème en imposant au makespan de prendre une valeur inférieure ou égale à 75.

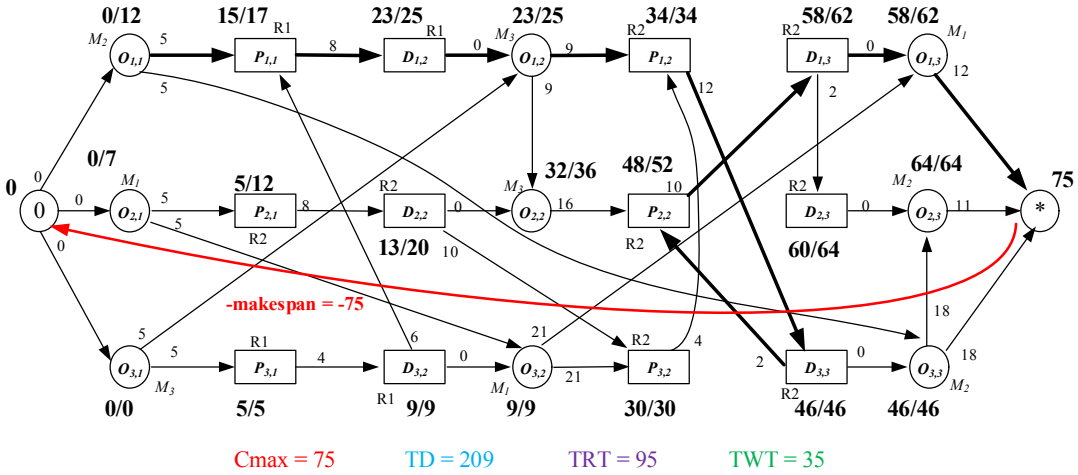


Figure 26 Time-lags après l'étape 1

La seconde étape de l'évaluation TLH consiste à minimiser le Total Duration par ajout itératif de time-lags maximaux de valeurs minimales entre la première opération-machine et la dernière opération-machine de chaque job (arcs en bleu sur la Figure 24). L'objectif est de rapprocher les dates de début des premières opérations de chaque job avec la date de fin de la dernière opération de chaque job. Afin de provoquer la minimisation la plus grande possible du TD, il est nécessaire d'ajouter les time-lags avec les valeurs les plus petites possible pour maximiser le décalage des dates. Évidemment, il existe une valeur du time-lag maximal en dessous de laquelle son ajout créer un cycle dans le graphe. Chaque ajout d'un time-lag maximal augmente le nombre de contraintes supplémentaires dans le graphe. Il est donc évident que l'ordre d'ajout des time-lags influence la solution obtenue à la fin de l'algorithme. Sauf à disposer d'information supplémentaire, un ordre quelconque peut être envisagé tel que par exemple un ordre lexicographique (cet ordre est l'objet de l'étude de la section 5.4.2).

Puisque toutes les opérations sont ordonnancées au plus tôt lors de l'étape 1, l'ajout d'un time-lag maximal TL_{ji} de j à i avec une valeur positive (mais modélisé par un arc avec une valeur négative) peut potentiellement retarder la date de début de l'opération i, tandis que l'opération j ne peut pas commencer plus tôt (car j est déjà au plus tôt). La valeur minimale assignée au time-lag est un problème traité dans la section 4.5.

La troisième étape de l'évaluation TLH concerne la minimisation du Total Riding Time par l'ajout successif de time-lags maximaux entre les opérations-machines $O_{i,j}$ et les opérations de chargement $P_{i,j-1}$ (arcs violets en pointillé sur la Figure 24). De manière similaire à l'étape 2, la valeur des time-lags maximaux doit être minimale et ne pas créer de cycle dans le graphe. Comme cela a été souligné, les opérations ne peuvent pas débuter plus tôt, mais peuvent uniquement être retardées (car lors de la première étape, ce sont les dates au plus tôt qui sont calculées), il semble alors naturel qu'il soit plus intéressant d'abord d'insérer les time-lags maximaux en fin de tournées des véhicules, puis suivant le chemin inverse des véhicules.

La quatrième étape de la procédure consiste à minimiser le Total Waiting Time, similairement aux deuxième et troisième étapes. Les time-lags maximaux sont ajoutés entre les opérations-machines $O_{i,j}$ et leur opération de livraison $D_{i,j}$ et entre les opérations de chargement $P_{i,j}$ et les opérations-machines $O_{i,j}$ (arcs pleins en vert sur la Figure 24). Comme pour l'étape 3, les time-lags sont insérés itérativement suivant le trajet inverse des véhicules.

Il est important de noter qu'à chaque insertion (quelle que soit l'étape de l'évaluation TLH) d'un time-lag, sa valeur doit être la plus petite possible afin de minimiser le critère concerné. L'affectation de valeurs aux time-lags maximaux peut conduire à un graphe cyclique (un graphe cyclique ne modélisant pas une solution), et trouver une valeur « correcte » pour un time-lag est un problème difficile qui est traité dans les sections 0 et 4.6.

Le Tableau 9 résume les différents paramètres de l'évaluation TLH au cours des différentes étapes. c est le numéro de l'étape concernée (entre 1 et 4, puisque TLH possède quatre étapes), le critère minimisé au cours de l'étape c est donnée par la seconde colonne du Tableau 9. φ_{TL}^{c} est l'ensemble des time-lags concernés au cours de l'étape c et θ_{TL}^{c} est l'ordre d'insertion des time-lags durant l'étape c de TLH.

Tableau 9 Les paramètres des différentes étapes de TLH

Étape c	**Critère minimisé**	**Time-lags concernés φ_{TL}^{c}**	**Ordre d'insertion θ_{TL}^{c}**
1	Makespan	Entre les * et 0	/
2	Total Duration	Entre la dernière opération et la première de chaque gamme	Lexicographique
3	Total Riding Time	Entre $O_{i,j}$ et $P_{i,j-1}$	Chemin inverse des tournées des véhicules
4	Total Waiting Time	Entre $O_{i,j}$ et $D_{i,j}$ et entre $P_{i,j}$ et $O_{i,j}$	Chemin inverse des tournées des véhicules

L'Algorithme 1 est l'algorithme de principe de minimisation d'un critère au cours d'une étape c (voir Tableau 9). Il est valable, quel que soit le critère optimisé (et donc quel que soit l'étape c considérée de la fonction TLH), ses deux paramètres sont φ_{TL}^{c} et θ_{TL}^{c}. φ_{TL}^{c} est l'ensemble des time-lags maximaux à insérer lors d'une étape c de la TLH, suivant l'ordre d'insertion θ_{TL}^{c}. L'algorithme *insertion_time_lag* (ligne 11) est l'objet de la section 4.6.

Algorithme 1 : Principe de la minimisation du critère à l'étape c

1. **Sortie**
2. G : le graphe avec critère i minimisé
3. **Entrée**
4. st : les dates de début des opérations
5. G : le graphe
6. c : le critère à minimiser
7. φ_{TL}^{c} : time-lags maximaux à insérer lors de l'étape c
8. θ_{TL}^{c} : ordre d'insertion des time-lags maximaux lors de l'étape c
9. **Début**
10. | **Pour** $\forall \phi \in \varphi_{TL}^{c}$ d'après l'ordre d'insertion θ_{TL}^{c} **Faire**
11. | | insertion_time_lag(G, st, v_{ϕ}^{-}) // voir Algorithme 2
12. | **FinPour**
13. | **Retourner** TL_{ji}
14. **Fin**

4.5. Valeur minimale d'un time-lag maximal

La sélection de la valeur du time-lag maximal est un point clé de l'évaluation TLH et un problème difficile, car la valeur du time-lag doit être minimale, mais l'ajout d'un time-lag de valeur trop petite dans le graphe disjonctif peut conduire à un graphe cyclique. Cette situation est à éviter.

La valeur minimale SP_{ij} d'un time-lag maximal TL_{ji} (cette valeur est positive, mais un time-lag est modélisé par un arc avec une valeur négative), de l'opération j à l'opération i est la différence entre la date de début au plus tard LS_i de i et la date de début au plus tôt ES_j de j.

Soit une position d'insertion d'un time-lag maximal TL_{ji} de j à i, la valeur minimale (qui est égale à SPM_{ij}) du time-lag maximal, sans générer de cycle dans le graphe, est $L_{ji} = ES_j - LS_i$ où LS_i et ES_j vérifient que :

1) $LS_* \coloneqq ES_*$ et $LS_j \coloneqq ES_j$

2) $\forall k\ LS_k \coloneqq \min_{u \in Succ(k)} (LS_u - l_{uk})$ où $Succ(k)$ est l'ensemble des successeurs de k, et l_{uk} est la longueur du chemin entre u et k

avec $l_{uk} = Pt_k \,||\, l_{uk} = TL_{ku}$

Cette proposition est directement déduite de la définition des dates au plus tôt et au plus tard des opérations.

De manière générale, entre deux opérations i et j de la même gamme, quels que soient leurs types (chargement, livraison ou opération-machine), il existe un ou plusieurs chemins allant de i à j. Ces chemins sont composés d'une succession d'arcs dont les extrémités n'appartiennent pas forcément à la même gamme que i et j.

Afin d'éviter la création d'un cycle dans le graphe par l'insertion d'un time-lag maximal entre j et i, il faut trouver le chemin le plus long. La longueur de ce chemin est la valeur minimale que le time-lag entre ces deux opérations peut prendre afin d'éviter la création de cycle. L'Algorithme 2 permet de calculer la valeur minimale d'un time-lag.

Algorithme 2 : Calcul de la valeur d'un time-lag minimal

Sortie
TL_{ji} : la valeur du time-lag de j vers i
Entrée
ES : les dates de début au plus tôt des opérations
G : le graphe
i et j : deux opérations reliées par un time-lag maximal de j vers i
Début
  $LS_* = ES_*$ // l'opération fictive finale
  $LS_j = ES_j$
  Pour $\forall k \in W$ **Faire**
    $l_{uk} = p_{tk}$ || $l_{uk} = TL_{ku}$
    $LS_k = \min_{u \in Succ(k)} (LS_u - l_{uk})$
  FinPour
  $TL_{ji} = ES_j - LS_i$
  Retourner TL_{ji}
Fin

La Figure 27 présente trois chemins $C2$, $C3$ et $C4$ reliant i à j. Sans perte de généralité, ces chemins sont classés suivant leur longueur ($C4$ est le chemin le plus long et $C2$ est le chemin le plus court). L'Algorithme 2 permet d'obtenir la longueur du plus long chemin, toutefois cet algorithme est gourmand en temps de calcul. Nous verrons par la suite qu'il existe un chemin théorique (C1 sur la Figure 27) de taille minimale qui peut être une borne inférieure à la valeur du time-lag maximal (voir section 4.3) et qui est très facile à calculer.

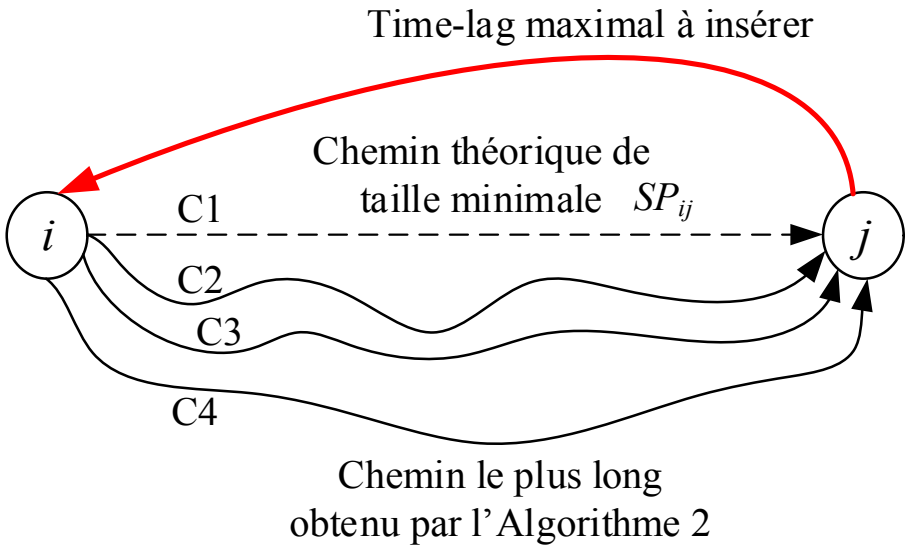


Figure 27 Chemins reliant deux opérations

La stratégie préconisée, durant l'évaluation TLH, est une stratégie ascendante où les time-lags maximaux sont d'abord initialisés avec la valeur la plus basse possible (la valeur est positive) puis itérativement augmentés. Cette stratégie permet d'obtenir des time-lags maximaux avec une valeur minimale et de s'assurer de toujours conserver une solution réalisable : si un cycle est détecté dans le graphe alors la valeur du dernier time-lag maximal inséré est augmentée. Par exemple, si un time-lag maximal est inséré de j vers i, de valeur

$TL_{ji} = 10$, alors celui-ci est modélisé par un arc de coût -10. Si un cycle est détecté suite à l'insertion de ce time-lag, alors la valeur de celui-ci est incrémentée et devient $TL_{ji} = 11$.

Puisque les solutions sont basées sur le vecteur de Bierwirth, aucun cycle dans le graphe ne peut être dû à une orientation incorrecte des arêtes. L'apparition d'un cycle est donc la conséquence d'une affectation incorrecte de la valeur d'un time-lag maximal. Ainsi, pour éviter la création d'un cycle, le graphe est évalué à chaque insertion d'un time-lag maximal, et si un cycle est détecté alors la valeur du time-lag maximal est augmentée de δ qui est la valeur de la longueur du cycle. La valeur δ est la valeur minimale requise pour mettre à jour le décalage temporel maximal, mais il n'y a aucune garantie qu'une seule itération soit requise puisque plusieurs cycles peuvent exister simultanément.

4.6. Insertion d'un time-lag maximal

L'Algorithme 3 illustre le principe d'insertion d'un time-lag maximal TL_{ji} dans le graphe G. Le time-lag est initialisé à sa valeur initiale v_{ji} (ligne 9) puis le graphe est évalué (ligne 10) grâce à un algorithme de plus long chemin de type Bellman-Ford. Si un cycle est détecté alors la valeur de la longueur du cycle δ est positive ($\delta > 0$). La valeur du time-lag est augmentée de δ (ligne 12), jusqu'à ce qu'il n'y ait plus de cycle détecté.

Algorithme 3 : Insertion d'un time-lag

1. **Sortie**
2. G : le graphe avec le time-lag inséré
3. v_{ji} : valeur du time-lag
4. **Entrée**
5. G : le graphe
6. TL_{ji} : le time-lag à insérer entre j et i
7. **Début**
8. | v_{ji}^{-} = calcul_valeur_initial()
9. | $v_{ji} = v_{ji}^{-}$
10. | evaluer_graphe($G \cup TL_{ji}, \delta, v_{ji}$)
11. | **Tant que** $\delta > 0$ **Faire**
12. | | $v_{ji} = v_{ji} + \delta$
13. | | evaluer_graphe($G \cup TL_{ji}, \delta, v_{ji}$)
14. | **FinTant que**
15. | **Retourner** $G \cup TL_{ji}$
16. **Fin**

La valeur minimale initiale v_{ji}^{-}, lors de la minimisation du Total Duration (deuxième étape de l'évaluation TLH), du time-lag maximal d'un job J_i est la somme de toutes les opérations $O_{i,j}, \forall j \in \{1 \dots n_i\}$ appartenant à celui-ci, plus les temps de transport entre les machines où sont réalisées les opérations $O_{i,j}, \forall j \in \{1 \dots n_i\}$.

La Figure 28 illustre l'insertion d'un time-lag maximal (en pointillé bleu) durant la seconde étape de l'évaluation TLH, entre les opérations $O_{1,3}$ et $O_{1,1}$ avec une valeur : $-TL_{O_{1,3},O_{1,1}} = -(ES_{O_{1,3}} - LS_{O_{1,1}}) = -46$. La valeur 46 affectée à $TL_{O_{1,3},O_{1,1}}$ est la somme des arcs du chemin (en gras) partant de $O_{1,1}$ et arrivant en $O_{1,3}$ sur la Figure 28, il est calculé

comme suit : $-TL_{O_{1,3},O_{1,1}} = -(5+8+0+9+12+2+10+0) = -46$. Il correspond à la durée minimale entre les deux opérations, cette durée est due : aux processing times ($Pt_{O_{1,1}} = 5$, $Pt_{O_{1,2}} = 9$) ; aux opérations de transport ($T_{P_{1,1},D_{3,3}} = 12$, $T_{D_{3,3},P_{2,2}} = 2$, $T_{P_{2,2},D_{1,3}} = 10$) ; et aux opérations de déchargement qui ont une durée nulle ($Pt_{D_{1,2}} = 0$, $Pt_{D_{1,3}} = 0$). Les arcs portant ces poids sont dessinés en bleu sur la Figure 28.

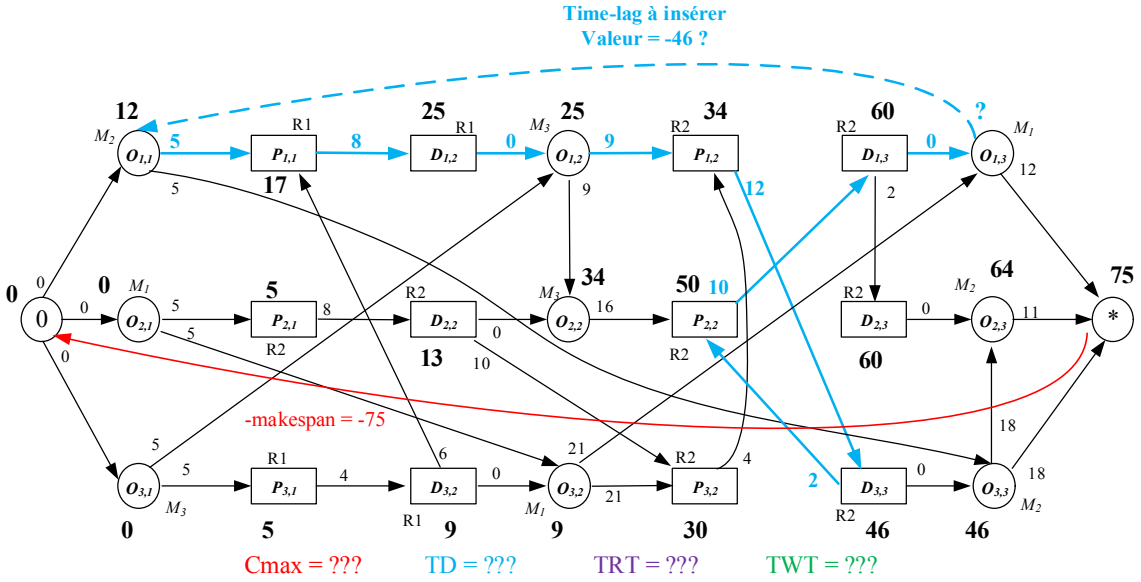


Figure 28 Chemin imposant la valeur minimale du time-lag

L'insertion du time-lag maximal entre les opérations $O_{1,3}$ et $O_{1,1}$ avec pour valeur : $-TL_{O_{1,3},O_{1,1}} = -46$ provoque l'apparition d'un cycle (Figure 29) qui comprend les opérations $O_{1,3}$, $O_{1,1}$, $P_{1,1}$, $D_{1,2}$, $O_{1,2}$, $O_{2,2}$, $P_{2,2}$, $D_{1,3}$. Sa longueur δ est la somme de tous les arcs, $\delta = -46+5+8+0+9+13+10+0 = 2$. Ce cycle est représenté en pointillé bleu sur la Figure 29. Comme le cycle est de valeur 2, il est nécessaire d'augmenter la valeur du time-lag maximal d'au moins deux unités pour que ce cycle n'existe plus dans le graphe. Si la valeur du time-lag de $O_{1,3}$ à $O_{1,1}$ est augmenté de 2, alors ce cycle est évité (Figure 30). La date de début de l'opération $O_{1,1}$ ne peut, donc, pas être 12 (Figure 29), mais est 10 (Figure 30) pour qu'aucun cycle ne soit détecté. L'insertion d'un time-lag maximal peut conduire à la création de plusieurs cycles, et dans ce cas, la valeur du time-lag doit être itérativement augmentée tant que des cycles sont détectés.

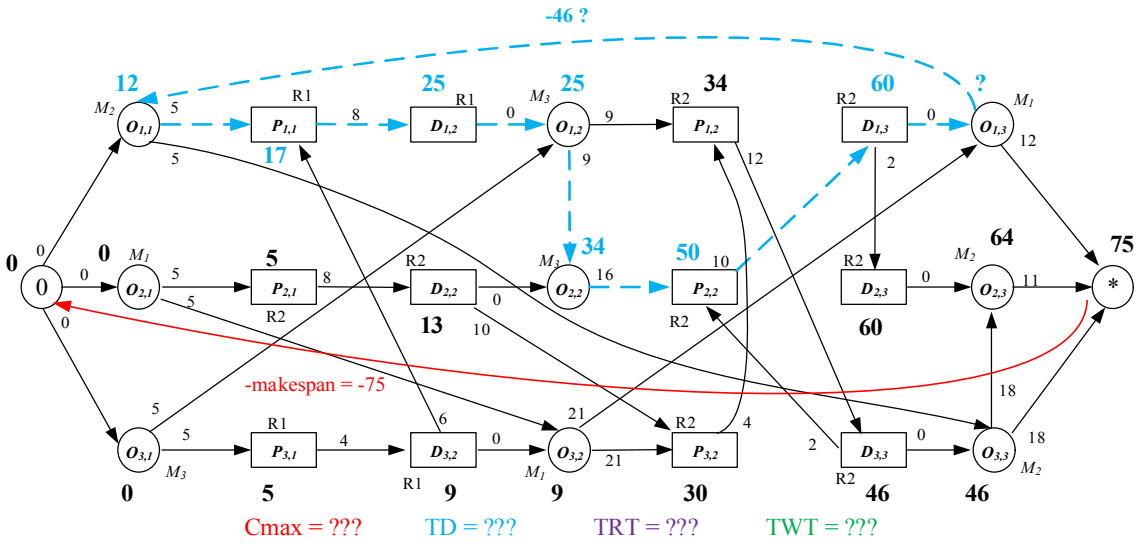


Figure 29 Insertion d'un time-lag menant à un graphe cyclique

La Figure 30 illustre le graphe à la fin de l'étape 2, qui est consacrée à la minimisation du Total Duration (TD) où les time-lags sont insérés itérativement dans l'ordre lexicographique. Le Total Duration est ramené de 209 (Figure 25) à 195 dans la Figure 30 : $TD_{step2} = (58 + 12 - 10) + (64 + 11 - 4) + (46 + 18 - 0) = 195$.

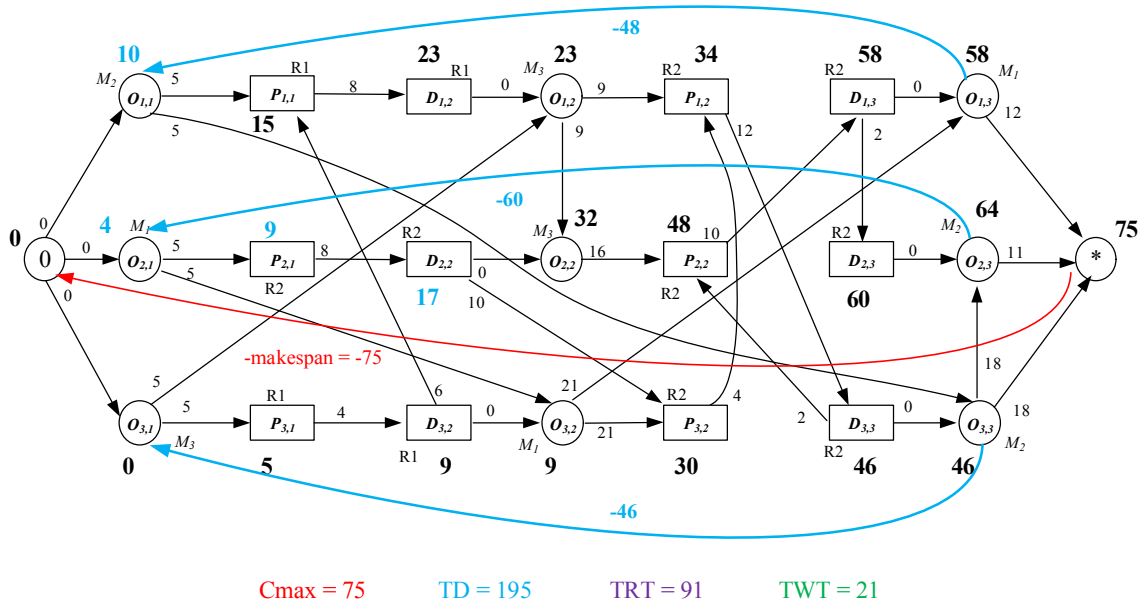


Figure 30 Time-lags après l'étape 2

La troisième étape de l'évaluation TLH concerne la minimisation du Total Riding Time (TRT). Chaque time-lag TL_{ji}, où i représente une opération-machine $O_{x,y}$ et j une opération de chargement $P_{x-1,y}$, est initialisé avec pour valeur le temps de transport de la machine $\mu_{O_{x-1,y}}$ à la machine $\mu_{O_{x,y}}$. Le graphe est évalué et si un cycle est détecté, alors la valeur du time-lag est augmentée (voir Algorithme 3). Par exemple, le plus court chemin entre $P_{1,2}$ et $O_{1,3}$ a pour valeur 24 $(12 + 2 + 10 + 0)$, ce chemin est représenté en violet sur la Figure 31. Cela signifie que le plus petit time-lag maximal qu'il est possible d'insérer avec une

possibilité de ne pas créer de cycle dans le graphe est de valeur 24. Un time-lag avec une valeur plus petite crée obligatoirement un cycle. Toutefois, insérer le time-lag maximal de valeur 24 ne garantit pas que le graphe reste acyclique : il est possible qu'il existe un autre chemin entre $P_{1,2}$ et $O_{1,3}$ de valeur supérieure à 24. En effet, $TL_{O_{1,3},P_{1,2}} = -26$, car une valeur plus petite conduit à un graphe cyclique (voir la Figure 32).

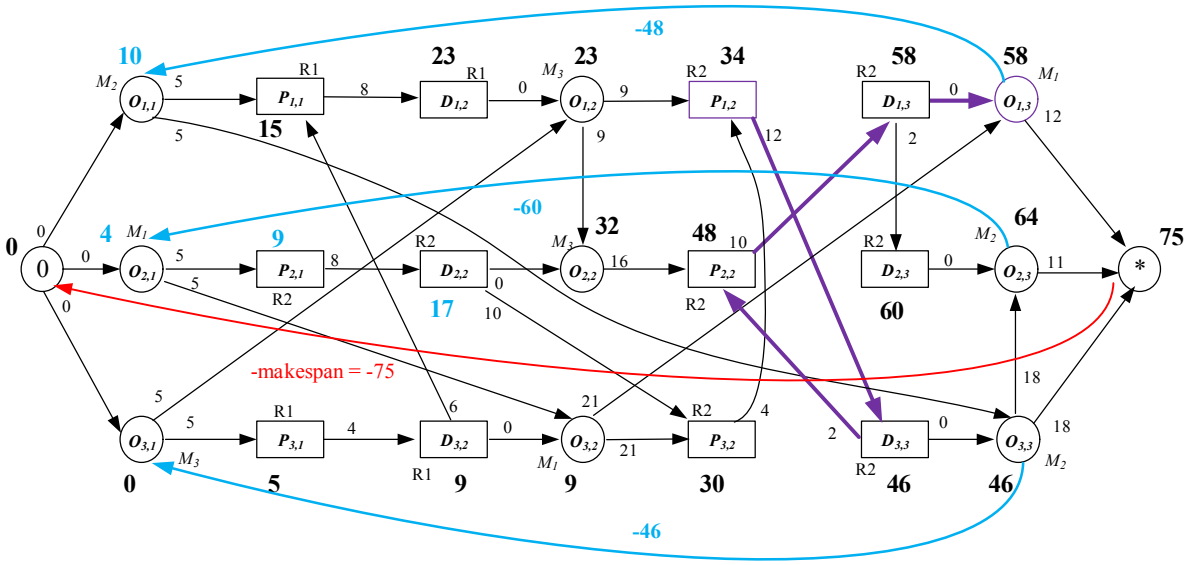


Figure 31 Chemin entre les deux opérations

Sur l'exemple traité, l'insertion d'un time-lag maximal entre $P_{1,2}$ et $O_{1,3}$ de valeur 24 crée un cycle. Le time-lag doit valoir 26 afin de ne pas provoquer l'apparition de cycle. Une fois ce time-lag inséré, un nouveau time-lag est inséré entre deux autres opérations. L'ordre d'insertion des time-lags suit le chemin inverse des tournées des véhicules.

La Figure 32 présente le graphe à la fin de cette étape où les time-lags sont itérativement insérés dans le graphe. Le Total Riding Time est passé de 91 (Figure 30) à 90 (Figure 32).

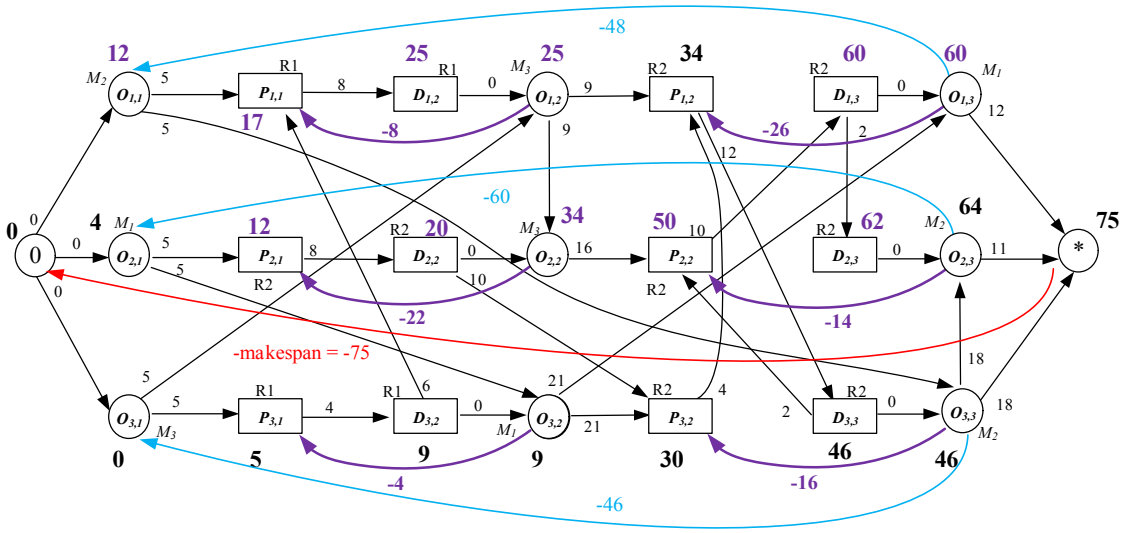


Figure 32 Time-lags après l'étape 3

La Figure 33 illustre le graphe à la fin de la quatrième étape de l'évaluation TLH, cette étape consiste à minimiser le Total Waiting Time (TWT). Les time-lags sont successivement ajoutés dans le graphe en suivant l'ordre inverse des tournées des véhicules. Chaque time-lag TL_{ji}, avec j représentant une opération-machine $O_{x,y}$, est initialisé à 0 (car le temps de déchargement est nul) puis itérativement augmenté si un cycle est détecté. Chaque time-lag TL_{ji} avec j correspondant à une opération de chargement $P_{x,y}$ est initialisé à $Pt_{x,j}$, qui est le temps de traitement de l'opération-machine $O_{x,y}$. La détection de cycle est de nouveau requise et la valeur du time-lag peut être augmentée si nécessaire. La date de début de l'opération de livraison $D_{2,3}$ est décalée de la date 60 (Figure 32) à la date 64 (Figure 33). Les nouveaux time-lag insérés sont en vert.

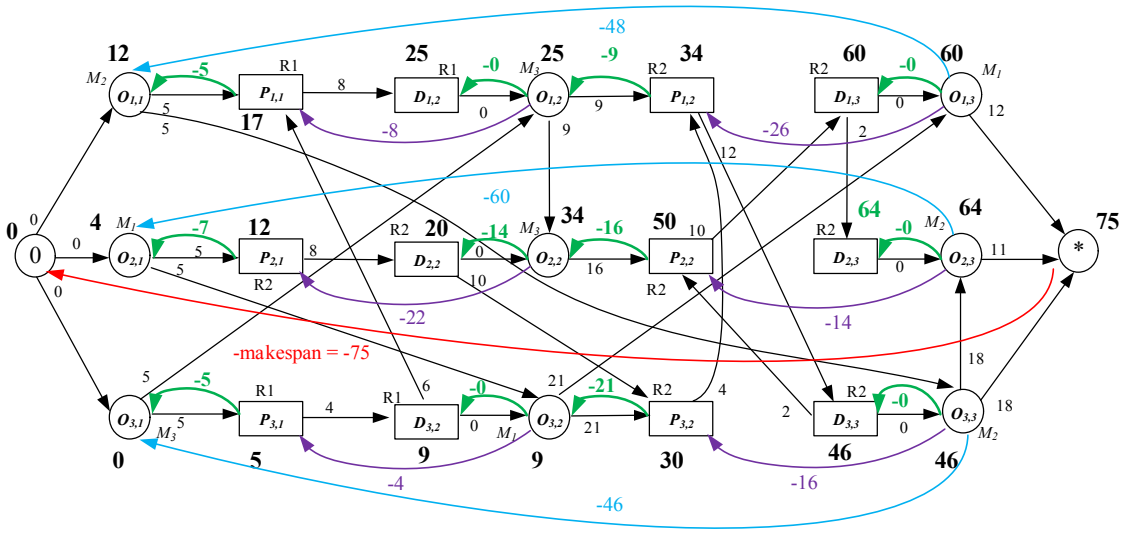


Figure 33 Time-lags après l'étape 4

Le Tableau 10 illustre l'évolution des critères lors de l'évaluation TLH qui minimise séquentiellement le makespan (dont la valeur est 75 dans l'exemple de la Figure 25), le Total Duration à l'étape 2, le Total Riding Time à l'étape 3 (qui passe de 95 à 90) et le Total Waiting Time à l'étape 4, qui passe de 35 à 17. Les trois critères constituant la qualité de service sont minimisés et il est important de noter qu'une fois minimisés, ceux-ci ne sont jamais dégradés durant une étape suivante.

Tableau 10 Évolution des critères au cours des itérations de TLH

	Critère minimisé	$\boldsymbol{Cmax}$	$\boldsymbol{TD}$	$\boldsymbol{TRT}$	$\boldsymbol{TWT}$
Étape 1	$\boldsymbol{Cmax}$	75	209	95	35
Étape 2	$\boldsymbol{TD}$	75	195	91	21
Étape 3	$\boldsymbol{TRT}$	75	195	90	19
Étape 4	$\boldsymbol{TWT}$	75	195	90	17

4.7. GRASP×ELS avec l'évaluation TLH

L'évaluation TLH est une fonction d'évaluation d'un graphe disjonctif et peut être incluse dans une métaheuristique afin d'explorer l'ensemble de l'espace de codage. De nombreux schémas de codage sont proposés pour le Job-shop Scheduling Problem (JSP), et ces schémas peuvent être classés en deux approches : soit une approche directe, soit une approche indirecte (Cheng et al., 1996).

La représentation utilisée par la métaheuristique est la représentation par répétition pour le JSP introduite par (Bierwirth, 1995) et présentée dans la section 2.3. Par conséquent, les points clés de la résolution du Job-shop Scheduling Problem with Routing sont les suivants :

- Une représentation des solutions basée sur un vecteur de (Bierwirth, 1995), plus un vecteur d'affectation des véhicules aux opérations de transport. Ces deux vecteurs forment une représentation indirecte des solutions (voir section 2.3).
- La fonction de décodage TLH, où l'évaluation du graphe disjonctif n'est pas basée sur la recherche du plus long chemin, mais qui est dédiée à la minimisation du makespan et simultanément la maximisation de la qualité de service. Cette fonction ne doit pas rechercher une solution semi-active, mais elle doit définir des dates de débuts pour les opérations maximisant la qualité de service.
- La définition d'une recherche locale alternant entre l'espace de codage et l'espace des solutions.
- Une métaheuristique, le GRASP×ELS (voir chapitre 1), favorisant l'exploration de l'espace de recherche et évitant de rester bloqué dans des minimaux locaux.

Une recherche locale, impliquant une procédure d'intensification du parcours de l'espace de recherche, est communément utilisée pour améliorer la qualité de la solution. (Larabi, 2010) et (Lacomme et al., 2013) définissent des mécanismes d'intensification efficaces pour le Job-shop Scheduling Problem with Transport qui reposent sur l'analyse du chemin critique du graphe. Leur proposition tire profit de la notion de blocs de (Grabowski et al., 1986), initialement introduits pour le job-shop. Un bloc de Grabowski est défini comme étant la séquence maximale d'opérations successives appartenant au chemin critique et exécutées par la même machine. (Van Laarhoven et al., 1992) proposent pour le job-shop, une recherche locale basée sur la permutation de deux opérations successives exécutées sur la même machine et appartenant au chemin critique. (Dell'Amico and Trubian, 1993) présente une des recherches locales les plus efficaces, en échangeant les opérations-machines situées

aux extrémités des blocs de Grabowski. (Larabi, 2010) et (Lacomme et al., 2013) étendent ces recherches locales pour le Job-shop Scheduling Problem with Transport en prenant en compte les opérations de chargement et de livraison : si deux opérations successives de chargement et/ou livraison appartiennent au chemin critique, alors elles sont soit interverties, soit l'une des opérations est affectée à un autre véhicule. Leur proposition peut directement être appliquée au JSPR.

Le principe du GRAS×ELS est donné par l'Algorithme 4 (voir chapitre 1 pour plus de détail). Une solution initiale S est crée aléatoirement (ligne 8). Cette solution est représentée dans l'espace de codage par un vecteur de Bierwirth et un vecteur OA d'affectation des véhicules aux opérations de transport (voir section 2.3) qui permettent d'obtenir un graphe disjonctif orienté. Ce graphe est évalué par la fonction TLH. Une recherche locale définie par (Lacomme et al., 2013) pour le JSPT est utilisée ensuite (ligne 9), elle utilise également la fonction TLH. L'ELS est exécutée (ligne 13) pour générer des voisins à la solution S (voir chapitre 1). Chaque voisin est évalué par la fonction TLH.

Algorithme 4 : GRASP×ELS pour le JSPR

1. **Sortie**
2. S^* : la meilleure solution
3. **Entrée**
4. $Data$: les données du problème
5. **Début**
6. | $S^* = \emptyset$
7. | **Pour** $iterGRASP = 1, \dots, maxGRASP$ **Faire**
8. | | S = Phase de construction //(contient TLH)
9. | | S = Recherche locale //(contient TLH)
10. | | **Si** $f(S) < f(S*)$ **Alors**
11. | | | $S'' = S'$
12. | | **FinSi**
13. | | S = ELS //(contient TLH)
14. | | **Si** $f(S) < f(S*)$ **Alors**
15. | | | $S'' = S'$
16. | | **FinSi**
17. | **FinPour**
18. | **Retourner** $S*$
19. **Fin**

5. Études numériques

5.1. Présentations des instances et des études

5.1.1. Instances pour le Job-shop Scheduling Problem with Routing

À notre connaissance, aucune instance spécifiquement dédiée au Job-shop Scheduling Problem with Routing n'est disponible. Un nouvel ensemble d'instances est introduit et est basé sur les instances de (Bilge and Ulusoy, 1995) initialement établies pour l'ordonnancement simultané d'opérations-machines et d'opérations de transport dans les

FMS. Ces instances sont communément admises par la communauté scientifique comme le jeu de données de référence du JSPT avec deux véhicules de capacité unitaire.

Les instances de (Bilge and Ulusoy, 1995) sont composées de deux ensembles (appelés datasets) $D1$ et $D2$: le premier comprend les instances avec un rapport entre le temps de transport t_{ik} et le temps de traitement p_k supérieur à 0.25 ($t_{ik}/p_k > 0.25$) ce qui signifie que les processing times sont largement supérieurs en durée au temps de transport. Il s'agit d'instances à dominante ordonnancement. Le second ensemble (ou dataset) contient les instances avec $t_{ik}/p_k \leq 0.25$. Les instances du second dataset sont donc à dominante transport. Toutes ces instances sont générées en considérant 10 jobsets et 4 layouts et définissent l'ensemble (ou dataset) $D1$ avec 40 instances et l'ensemble $D2$ avec 42 instances. Les instances de $D1$ sont notées EX_{xy}, où x est le jobset utilisé et y le layout considéré. Pour l'ensemble $D2$, les instances sont nommées EX_{xyz}, où z est un chiffre supplémentaire avec une valeur de 0 si les temps de transport sont divisés par deux et les temps de traitement doublés ou une valeur de 1 si les temps de transport sont divisés par deux et les temps de traitement triplés.

Pour les instances de (Bilge and Ulusoy, 1995), les deux véhicules ont les mêmes temps de transport et ces temps de transport sont indépendants du véhicule et du chargement. Ces instances sont étendues pour des évaluations numériques avec une flotte de véhicules de capacité non unitaire. Dans ce cas, deux véhicules d'une capacité de deux jobs chacun sont considérés, et chaque job représente le même volume. Les gammes opératoires des jobs et les temps de transport sont identiques aux instances originelles.

5.1.2. GRASP×ELS pour le JSPT et le JSPR

L'objectif du JSPR est de minimiser le makespan et de maximiser la qualité de service en définissant $h_{QoS}(y) = 1/h_{cost}(y)$. Les deux critères sont ordonnés par une somme agrégée avec les coefficients (α ; β) modélisant l'importance relative du makespan et de la qualité de service. La fonction objectif est définie comme suit :

$$F(y) = \alpha \times h_{Cmax}(y) + \beta \times h_{cost}(y)$$

Où :

- y : une solution du problème ;
- $h_{Cmax}(y) = Cmax$ de la solution y, c'est-à-dire le makespan de y ;
- $h_{cost}(y) = TD(y) + TRT(y) + TWT(y)$, qui sont respectivement le Total Duration, le Total Riding Time et le Total Waiting Time de la solution y ;
- $h_{QoS}(y) = 1/h_{cost}(y)$ est la qualité de service de y.

Grâce à la métaheuristique utilisée et à la fonction objectif $F(y) = \alpha \times h_{Cmax}(y) + \beta \times h_{cost}(y)$, le GRASP×ELS proposé peut résoudre le Job-shop Scheduling Problem with Transport (JSPT) (minimisation du makespan uniquement) et le Job-shop Scheduling Problem with Routing (JSPR) (minimisation du makespan et maximisation de la qualité de service).

Pour la résolution du JSPT, la fonction objectif est définie avec $\alpha = 1$ et $\beta = 0$ correspondant donc à une évaluation des dates au plus tôt. L'algorithme codé est générique et

dans ce cas précis, il correspond simplement à la première étape de l'évaluation TLH et est donc équivalent à un algorithme de plus court chemin classique.

Deux approches différentes sont expérimentées pour résoudre le JSPR. La première approche est une approche intégrée, tandis que la seconde est une approche séquentielle :

- La fonction objectif de l'approche intégrée (JSPR intégré) est $F = \alpha \times h_{Cmax}(y) + \beta \times h_{cost}(y)$ où $\alpha = 10\,000$ et $\beta = 1$, l'évaluation TLH est utilisée à chaque itération de la métaheuristique.
- Pour l'approche séquentielle (JSPR séquentiel), $\alpha = 1$ et $\beta = 0$ dans la fonction objectif pendant le GRASP×ELS (cette étape est équivalente à la résolution du JSPT). Puis l'évaluation TLH est appliquée à la meilleure solution trouvée durant la résolution du JSPT, pour obtenir une solution du JSPR.

Toutes les instances et les solutions sont téléchargeables sur les pages web suivantes : http://fc.isima.fr/~gondran/researches.html et http://fc.isima.fr/~gondran/JSPT/tlh.html.

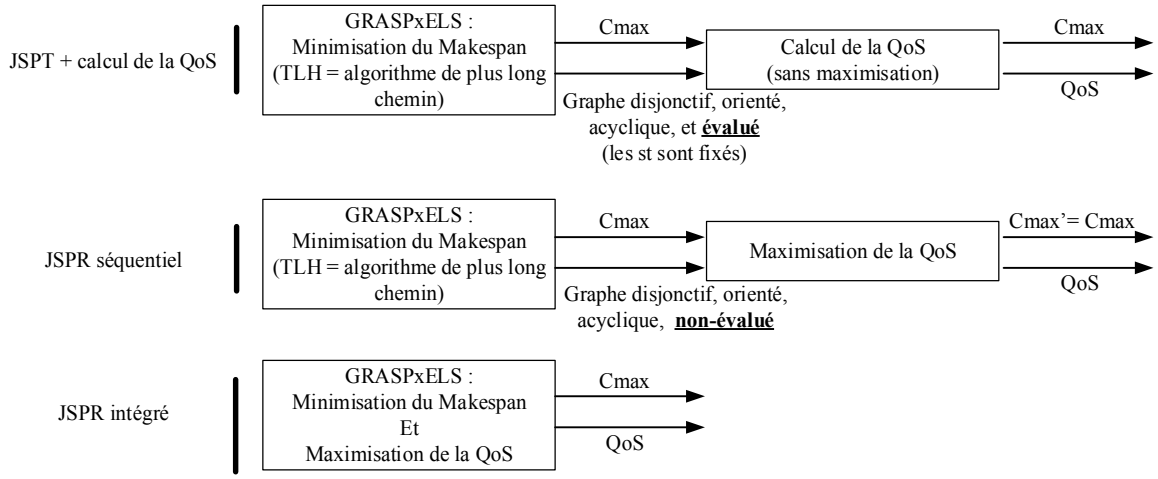


Figure 34 Les différentes approches de résolution

La Figure 34 représente les trois approches utilisées pendant les études numériques. Suite à la résolution du JSPT, la QoS de la meilleure solution trouvée est calculée (mais non maximisée, les dates des opérations sont fixes) afin de comparer ce critère aux différentes approches.

5.1.3. Présentations des quatre études

Les évaluations numériques sont divisées en quatre études.

La première étude (section 5.2) concerne la capacité du GRASP×ELS à fournir des résultats comparables à la littérature pour résoudre le JSPT avec une flotte de véhicules de capacité unitaire. À notre connaissance, il n'existe pas d'article dans une revue proposant de résultats numériques pour le JSPT avec une flotte de véhicules de capacité non unitaire ou proposant d'étude portant sur la QoS.

La seconde étude (section 5.3) compare les résultats obtenus lors de la résolution du JSPT par le GRASP×ELS avec les résultats obtenus lors de la résolution du JSPR intégré, en termes de makespan et de QoS (Gondran et al., 2018a).

La troisième étude (section 5.4) est composée de deux évaluations numériques : la première porte sur l'ordre d'insertion des time-lag maximaux lors de l'évaluation TLH ; et la seconde évaluation numérique a pour objet la capacité de la fonction TLH à trouver rapidement des solutions de bonne qualité (par rapport à l'évaluation exacte du graphe à l'aide de la formulation linéaire et de CPLEX) dans des temps raisonnables (Gondran et al., 2018a).

La quatrième étude (section 5.5) porte sur une comparaison entre le JSPT, le JSPR intégré et le JSPR séquentiel pour des véhicules de capacité unitaire et de capacité non unitaire (Gondran et al., 2019).

Les études sont toutes effectuées dans les mêmes conditions, sous Windows 7 sur un ordinateur Intel Core i7-4790 3.60 GHz avec 16.0 Go de RAM, ce qui équivaut à 2671 MFlops selon Dongarra (http://www.roylongbottom.org.uk/linpackresults.htm).

L'évaluation TLH est incluse dans une métaheuristique GRASP×ELS, les deux sont implémentées en C++. Le nombre d'itérations du GRASP est de 200, le nombre d'itérations de l'ELS est de 60 et le nombre de voisins lors de l'ELS est de 30. Chaque étude est répétée cinq fois avec une séquence de nombres aléatoires différente à chaque réplication pour limiter l'influence de la part d'aléatoire sur les résultats.

5.1.4. Notations

Cette section introduit les notations qui sont utilisées pour présenter les résultats numériques.

$h_k^{i,j}(y)$ La meilleure solution trouvée pour l'instance i, durant la réplication j, pour le critère $k \in \{Cmax\,;\,cost\}$;

$h_k^{i,*}(y)$ $= \min_{j\in\{1,\dots,n\}} \left(h_k^{i,j}(y)\right)$: la meilleure solution trouvée pour l'instance i durant les n réplications, pour le critère $k \in \{Cmax\,;\,cost\}$;

$h_k^{i,n}(y)$ $= \operatorname{avg}_{j\in\{1,\dots,n\}} (h_k^{i,j}(y))$: la moyenne de la meilleure solution trouvée pour l'instance i durant les n réplications, pour le critère $k \in \{Cmax\,;\,cost\}$;

$\overline{h_k^{z,*}(y)}$ $= \operatorname{avg}_{i\in\{z\}}(h_k^{i,*}(y))$: la moyenne des meilleures solutions trouvées pour les n réplications des i instances appartenant au $z \in \{D1\,;\,D2\}$, pour le critère $k \in \{Cmax\,;\,cost\}$;

$\overline{h_k^{z,n}(y)}$ $= \operatorname{avg}_{i\in\{z\}}(h_k^{i,n}(y))$: la moyenne de la meilleure solution trouvée durant les n réplications des i instances appartenant au $z \in \{D1\,;\,D2\}$, pour le critère $k \in \{Cmax\,;\,cost\}$;

$\overline{st^{z,*}(y)}$ Le temps moyen normalisé pour trouver la meilleure solution y, durant la meilleure réplication $*$ parmi les n réplications, pour toutes les instances du dataset $z \in \{D1\,;\,D2\}$;

$\overline{stt^{z,*}(y)}$ Le temps total normalisé de résolution pour la meilleure réplication $*$ parmi les n réplications, pour toutes les instances du dataset $z \in \{D1\,;\,D2\}$;

$\overline{st^{z,n}(y)}$ Le temps total normalisé pour trouver la meilleure solution durant les n réplications, pour toutes les instances du dataset $z \in \{D1\,;\,D2\}$;

$\overline{stt^{z,n}(y)}$ Le temps total normalisé de résolution pour les n réplications, pour toutes les instances du dataset $z \in \{D1\,;\,D2\}$;

$gap^{i,*}(y)$ Le meilleur écart, c'est-à-dire l'écart entre la meilleure solution trouvée

$h_{Cmax}^{i,n}(y)$ (de l'instance i durant les n réplications, et la borne inférieure LB^i donnée par (Ulusoy et al., 1997), $gap^{i,*}(y) = (h_k^{i,*}(y) - LB^i)/LB^i$;

$\overline{gap^{z,*}(y)}$ $= \underset{i \in \{z\}}{avg}(gap^{i,*}(y))$: l'écart moyen de la meilleure réplication de toutes les instances du dataset $z \in \{D1\,;\,D2\}$;

$\overline{\overline{gap^{z,n}(y)}}$ L'écart moyen pour les n réplications pour toutes les instances du dataset $z \in \{D1\,;\,D2\}$;

$\Delta_k^{i,*}$ L'écart entre le JSPT et le JSPR (intégré ou séquentiel) de la meilleure réplication $*$ de l'instance i pour le critère $k \in \{Cmax\,;\,cost\}$;

$\overline{\Delta_k^{i,*}}$ $= \underset{i \in \{z\}}{avg}(\Delta_k^{i,*})$ est l'écart moyen $\Delta_k^{i,*}$ pour toutes les instances du dataset $z \in \{D1\,;\,D2\}$.

5.2. Première étude : comparaison avec la littérature

À notre connaissance, dix-neuf approches avec des résultats numériques et utilisant les instances de (Bilge and Ulusoy, 1995) sont publiées pour le JSPT, pour une flotte de véhicules de capacité unitaire. Le Tableau 11 introduit ces méthodes et indique quels datasets ($D1$ et/ou $D2$) sont utilisés pour tester chaque approche. Aucun article ne propose des résultats numériques pour le JSPT avec plusieurs véhicules de capacité non unitaire.

Tableau 11 Publications avec les instances de (Bilge and Ulusoy, 1995)

	Méthode	**Dataset *D1***	**Dataset *D2***	**Capacité unitaire**	**Capacité non unitaire**
(Bilge and Ulusoy, 1995)	STW	X	X	X	-
(Ulusoy et al., 1997)	LB et UGA	X	X	X	-
(Abdelmaguid et al., 2004)	AGA	X	X	X	-
(Reddy and Rao, 2006)	PGA	X	X	X	-
(Deroussi et al., 2008)	SALS	X		X	-
(Subbaiah et al., 2009)	SFHA	X	X	X	-
(Babu et al., 2010)	DE	X	X	X	-
(Larabi, 2010)	MA	X		X	-
(Kumar et al., 2011)	PDE	X	X	X	-
(Erol et al., 2012)	MAS	X	X	X	-
(Zhang et al., 2012)	GATS	X		X	-
(Lacomme et al., 2013)	MEMA		X	X	-
(Zhang et al., 2014)	MSB	X		X	-
(Zheng et al., 2014)	TS-SPMA	X	X	X	-
(Sahin et al., 2015)	FMAS	X	X	X	-
(Umar et al., 2015)	WA	Seulement layout : 1 et 2		X	-
(Baruwa and Piera, 2016)	ALS	X	X	X	-
(Nouri et al., 2016b)	GATS+HM	X		X	-
(Zheng et al., 2016)	HRA	X	X	X	-

L'étude se focalise sur les cinq dernières publications du JSPT avec une flotte de véhicules de capacité unitaire. Ces cinq méthodes proposent de nombreuses nouvelles meilleures solutions et sont testées sur l'ensemble des instances des deux datasets, excepté pour l'approche proposée par (Umar et al., 2015) qui n'utilise que deux layouts et un dataset. Cette approche n'est donc pas prise en compte dans cette étude, car leurs résultats ne

concernent pas toutes les instances. Les résultats numériques des 19 méthodes sont disponibles sur la page web : http://fc.isima.fr/~gondran/JSPT/state_of_the_art.html.

Rappelons que (voir section 1.2), la méthode TS_SPMA est introduite par (Zheng et al., 2014) et consiste en un algorithme de type Tabu Search. (Sahin et al., 2015) décrivent une approche multiagents : le FMAS pour un ordonnancement dynamique avec des capacités de traitements souples, ils résolvent également une version statique du JSPT. La méthode ALS est proposée par (Baruwa and Piera, 2016) : un Anytime Layered Search basé sur un réseau de Petri coloré hybridé avec une heuristique. Le GATS+HM est introduit par (Nouri et al., 2016c) et est une approche basée sur un Clustered Holonic Multiagent Model. Enfin, HRA est une Hormone Regulation based Approach pour des ordonnancements dynamiques, cette méthode est proposée par (Zheng et al., 2016) et est également testée sur les instances en mode statique. Pour chaque méthode, les résultats en termes de solutions et temps de calcul fournis dans les articles sont résumés dans le Tableau 12.

Tableau 12 Résultats fournis par les méthodes récemment publiées

	Méthode	**Dataset *D*1**		**Dataset *D*2**	
		Best	Time	Best	Time
(Zheng et al., 2014)	TS_SPMA	×	/	×	/
(Sahin et al., 2015)	FMAS	×	/	×	/
(Baruwa and Piera, 2016)	ALS	×	×	/	/
(Nouri et al., 2016b)	GATS+HM	×	×	/	/
(Zheng et al., 2016)	HRA	×	/	×	/
Légende	×	L'information est fournie			
	/	Pas d'information donnée			

Afin d'assurer une étude comparative équitable des différentes méthodes publiées, le facteur de vitesse est calculé et est tiré d'articles de recherche antérieurs, y compris, mais non limité à Dongarra, au site web dédié au benchmark Linpack (http://www.roylongbottom.org.uk/linpackresults.htm) et au site : http://asteroidsathome.net/boinc/cpu_list.php

Les facteurs de vitesse sont présentés dans le Tableau 13, ce dernier livre également les informations disponibles sur l'ordinateur, puisque les performances MFlops ne sont pas les seules influences sur le temps CPU. Le système d'exploitation et le langage utilisé des articles publiés précédemment, sont également rapportés dans cette table. Les temps de calcul normalisés des méthodes sont calculés à partir des temps de calcul présentés dans la publication originale multipliés par le facteur de vitesse.

Tableau 13 Performances relatives des ordinateurs

	(Zheng et al., 2014)	(Sahin et al., 2015)	(Baruwa and Piera, 2016)	(Nouri et al., 2016c)	(Zheng et al., 2016)	GRASP×ELS
Méthode	**TS_SPMA**	**FMAS**	**ALS**	**GATS+HM**	**HRA**	**TLH**
Nombre de répliques	5	/		10	/	5
Ordinateur	Pentium IV 2.60 GHz	/	2.60 GHz AMD Opteron 4 GB RAM	Intel Core 2 Duo 2.10 GHz	Intel Core 2 Duo 2.0 GHz	Intel Core i7-4790 3.60 GHz
OS	Windows XP	/		/	Windows 7	Windows 7
Langage	C	JACKTM et Java	C++	Java	Java	C++
MFlops	3066	/	2400	2400	2400	2671
Facteur de vitesse	1.15	/	0.90	0.90	0.90	1

L'évaluation TLH proposée est générique et permet la résolution du Job-shop Scheduling Problem with Transport, qui est défini par une fonction objectif uniquement basée sur le makespan sans aucune considération pour la qualité de service. Dans ce cas la fonction objectif est définie par $\alpha = 1$ et $\beta = 0$ dans l'évaluation TLH. Celle-ci correspond à un algorithme de plus long chemin de type Dijkstra.

Le Tableau 14 et le Tableau 15 présentent les résultats obtenus par le GRASP×ELS résolvant le JSPT avec une flotte de véhicules de capacité unitaire pour les deux datasets. L'écart moyen, $\overline{gap_k^{z,*}(y)}$, de la meilleure exécution est 26.8% pour le dataset $D1$ (Tableau 14) et 6.5% pour le dataset $D2$ (Tableau 15).

Pour le dataset $D1$ (Tableau 14), le GRASP×ELS concurrence les meilleures méthodes publiées, avec un écart par rapport à la borne inférieure de $\overline{gap_k^{z,*}(y)} = 26.8\ \%$, ce qui est meilleur que HRA et FMAS. Le GATS+HM fournit un écart d'environ 20.6 % ce qui est légèrement inférieur à l'écart du GRASP×ELS avec un temps de calcul d'environ 12.08 secondes, soit quatre fois plus que le temps de calcul du GRASP×ELS, qui est d'environ 2.83 secondes. Similairement, l'ALS fournit un écart d'environ 25.5 %, mais un temps de calcul de l'échelle de 503.1 secondes en moyenne. Le TS_SPMA obtient des résultats d'environ 25.5 %, mais aucune étude comparative équitable sur le temps de calcul n'est possible.

Tableau 14 Performances du GRASP×ELS pour le JSPT, dataset $D1$

	Dataset $D1 : t_{ik}/p_k > 0.25$					
	$\overline{gap_k^{z,*}(y)}$ %	$\overline{st^{z,*}(y)}$ (second)	$\overline{stt^{z,*}(y)}$ (second)	$\overline{gap_k^{z,n}(y)}$ %	$\overline{st^{z,n}(y)}$ (second)	$\overline{stt^{z,n}(y)}$ (second)
TS_SPMA	25.5 %	/	/	/	/	/
FMAS	44.9 %	/	/	/	/	/
ALS	25.5 %	503.1	3600	/	/	/
GATS+HM	20.6 %	12.08	/	/	/	/
HRA	70.4 %	/	/	/	/	/
GRASP×ELS	**26.8 %**	**2.83**	**10.50**	**27.4 %**	**2.10**	**10.51**

La question typique qui se pose est de savoir si l'écart de solutions entre le GRASP×ELS et le GATS+HM pourrait être réduit en augmentant le nombre d'itérations. Plusieurs évaluations numériques ont été réalisées pour donner au GRASP×ELS le même temps de fonctionnement que l'algorithme GATS+HM, mais aucune amélioration

significative n'a été obtenue. Il y a donc probablement une convergence très et trop rapide du GRASP×ELS et les mécanismes de diversifications n'ont pas permis l'exploration d'une autre région de l'espace.

Pour le dataset $D2$ (Tableau 15), le GRASP×ELS fournit des résultats de bonne qualité avec une déviation moyenne de 6.5%, ce qui le place entre la méthode TS_SPMA (6.0%) et la méthode FMAS (7.2%). Mais aucune étude comparative n'est possible sur les temps de calcul, car aucune des méthodes ne donne leurs temps d'exécution.

Tableau 15 Performances du GRASP×ELS pour le JSPT, dataset $D2$

	Dataset $D2 : t_{ik}/p_k \leq 0.25$					
	$\overline{gap_k^{z,*}(y)}$ %	$\overline{st^{z,*}(y)}$ (second)	$\overline{stt^{z,*}(y)}$ (second)	$\overline{gap_k^{z,n}(y)}$ %	$\overline{st^{z,n}(y)}$ (second)	$\overline{stt^{z,n}(y)}$ (second)
TS_SPMA	6.0 %	/	/	/	/	/
FMAS	7.2 %	/	/	/	/	/
ALS	/	/	/	/	/	/
GATS+HM	/	/	/	/	/	/
HRA	11.3 %	/	/	/	/	/
GRASP×ELS	**6.5 %**	**0.15**	**8.01**	**7.0 %**	**1.60**	**8.02**

Cette première étude numérique nous amène à considérer que GRASP×ELS est adapté et bien paramétré pour résoudre le JSPT avec deux véhicules de capacité unitaire.

5.3. Deuxième étude : JSPT et JSPR intégré

La seconde étude est une comparaison des critères de makespan et de QoS entre le JSPT et le JSPR intégré. La résolution du JSPT ne prend pas en compte l'évaluation de la QoS, et correspond à une évaluation de type Dijkstra. Afin de comparer les méthodes de résolution, la QoS du JSPT est calculée sur la solution finale retournée par le GRASP×ELS. Il est important de noter que dans le cas du JSPT, la QoS est uniquement évaluée sans essayer de la maximiser : les dates de début des opérations sont des contraintes et ne peuvent pas être modifiées.

Cette étude est découpée en deux sections : une section (5.3.1) pour le cas d'une flotte de véhicules de capacité unitaire, et une section (5.3.2) pour le cas d'une flotte de véhicules de capacité non unitaire.

5.3.1. Capacité unitaire

Le Tableau 16 et le Tableau 17 présentent les résultats obtenus pour le JSPT (avec en plus une évaluation – sans modifier les dates des opérations – de la QoS) et le JSPR. La première colonne est le nom de l'instance, la seconde est la borne inférieure proposée par (Ulusoy et al., 1997). Les colonnes 3, 4, 5 et 6 concernent la résolution du JSPT par le GRASP×ELS avec pour fonction objectif : $F = h_{Cmax}(y)$ où $h_{Cmax}(y)$ est le makespan. Les colonnes 7 à 12 se rapportent à la résolution du JSPR intégré (la fonction objectif est : $F(x) = 10\,000 \times h_{Cmax}(x) + 1 \times h_{cost}(x)$).

Les colonnes 3 et 7 donnent la valeur du makespan de la meilleure solution trouvée, durant les cinq réplications, respectivement par le JSPT et le JSPR intégré. Par exemple, pour

l'instance EX11 la meilleure solution a un makespan de 96 pour le JSPT et le JSPR. Les colonnes 4 et 9 sont la valeur du coût (*cost*) de la meilleure solution trouvée, durant les cinq réplications, respectivement par le JSPT et le JSPR intégré. Par exemple, pour l'instance EX12, le coût du JSPT est de 799 et le coût du JSPR est de 529. Les colonnes 5 et 11 sont les temps normalisés mis par le GRASP×ELS pour obtenir la meilleure solution, lors de la meilleure réplique, respectivement lors de la résolution du JSPT et du JSPR intégré. Les colonnes 6 et 12 sont les temps totaux normalisés de résolution par les deux méthodes.

Une valeur de $h_{Cmax}^{i,*}(y)$ (colonne 3) égale à $h_{Cmax}^{i,*}(x)$ (colonne 7) signifie que la meilleure solution obtenue pour le JSPT et la meilleure solution obtenue pour le JSPR intégré ont un makespan équivalent. Par exemple pour l'instance EX41 (Tableau 16), $h_{Cmax}^{i,*}(y) = h_{Cmax}^{i,*}(x) = 112$, il en résulte un écart $\Delta_{Cmax}^{i,*}$ nul. $\Delta_{Cmax}^{i,*}$ est l'écart, en pourcentage, entre $h_{Cmax}^{i,*}(x)$ et $h_{Cmax}^{i,*}(\mathrm{y})$ avec $\Delta_{Cmax}^{i,*} = (h_{Cmax}^{i,*}(y) - h_{Cmax}^{i,*}(x))/h_{Cmax}^{i,*}(y)$. Pour l'instance EX101 (Tableau 16), $\Delta_{Cmax}^{i,*} = -0.68\%$ signifie que le JSPT a obtenu un meilleur makespan ($h_{Cmax}^{i,*}(y) = 148$) que le JSPR intégré ($h_{Cmax}^{i,*}(x) = 149$). L'inverse se produit également, avec un $\Delta_{Cmax}^{i,*} = 3.88\%$ pour l'instance EX31 (Tableau 16) : le JSPR intégré a trouvé un meilleur makespan que le JSPT. Ceci est dû à l'impact du critère de QoS concernant le choix de la meilleure solution lors de la sélection du meilleur voisin au cours du GRASP×ELS, impliquant un parcours de l'espace de recherche différent.

La valeur moyenne des $\Delta_{Cmax}^{i,*}$ est $\overline{\Delta_{Cmax}^{i,*}} = 0.16\%$ pour le dataset $D1$ (Tableau 16) et $\overline{\Delta_{Cmax}^{i,*}} = 0.00\%$ pour le dataset $D2$ (Tableau 17). Un écart de 0.16% signifie que le makespan moyen trouvé durant la résolution du JSPR intégré est meilleur que le makespan moyen trouvé lors de la résolution du JSPT. Cet écart résulte d'une meilleure diversification durant le GRASP×ELS. Un écart de 0% signifie que les deux méthodes ont des solutions avec un makespan équivalent.

$st^{i,*}$ est le temps mis par le GRASP×ELS pour trouver la meilleure solution tandis que $stt^{i,*}$ est le temps total d'exécution du GRASP×ELS. Par exemple, pour l'instance EX21 (Tableau 16), le temps pour trouver la meilleure solution pour le JSPT est de 1.94 seconde, et le temps total de résolution du JSPT est de 8.72 secondes. Le temps que requiert le GRASP×ELS pour fournir la meilleure solution est de 11.40 secondes (soit quasiment 5 fois plus long que le JSPT) et le temps total est de 19.93 secondes (soit 2 fois plus long que le JSPT).

La différence des temps d'exécution s'explique par le fait que le nombre d'évaluations du graphe disjonctif est beaucoup plus grand pour le JSPR (puisqu'une évaluation est réalisée à chaque insertion). Par ailleurs, pour le JSPT, le temps $st^{i,*}$ correspond à la première solution obtenue avec le makespan minimal, tandis que pour le JSPR, la solution avec le makespan minimal et la QoS minimale peut nécessiter beaucoup plus de temps.

L'écart entre la QoS de la meilleure solution trouvée durant le JSPT et la meilleure solution trouvée durant le JSPR intégré est défini par $\Delta_{cost}^{i,*} = (h_{cost}^{i,*}(y) - h_{cost}^{i,*}(x))/h_{cost}^{i,*}(y)$ qui représente donc l'amélioration de la qualité de service du JSPR par rapport à celle du JSPT. Un $\Delta_{cost}^{i,*} > 0$ signifie que la solution du JSPR a une meilleure QoS que celle du JSPT. C'est le cas, par exemple pour l'instance EX11 (Tableau 16) avec $\Delta_{cost}^{i,*} = 25.33\%$: la solution du JSPT a pour critère $h_{cost}^{i,*}(y) = 683$ et la solution du JSPR intégré a pour valeur de $h_{cost}^{i,*}(x) = 510$. $\overline{\Delta_{cost}^{i,*}}$ est l'écart moyen des $\Delta_{cost}^{i,*}$, $\overline{\Delta_{cost}^{i,*}} = 35.78\%$ pour le dataset $D1$

(Tableau 16) et $\overline{\Delta_{cost}^{i,*}} = 46.37\%$ pour le dataset $D2$ (Tableau 17). Ces écarts mettent en évidence l'importance de considérer la qualité de service pendant le schéma de résolution.

Tableau 16 JSPT et JSPR intégré pour $D1$: véhicules de capacité unitaire

	Ulusoy LB	**Résolution du JSPT : $F = h_{cmax}(y)$**				**Résolution du JSPR intégré : $F = 10\,000 \times h_{cmax}(x) + h_{cost}(x)$**					
		$h_{Cmax}^{i,*}(y)$	$h_{cost}^{i,*}(y)$	$st^{i,*}$	$stt^{i,*}$	$h_{Cmax}^{i,*}(x)$	$\Delta_{Cmax}^{i,*}$	$h_{cost}^{i,*}(x)$	$\Delta_{cost}^{i,*}$	$st^{i,*}$	$stt^{i,*}$
EX11	72	96	683	0.01	6.80	96	0.00	510	25.33	5.05	15.92
EX21	86	100	799	1.94	8.72	100	0.00	529	33.79	11.40	19.93
EX31	81	103	858	5.21	9.09	99	3.88	543	36.71	4.28	20.37
EX41	76	112	831	3.52	11.37	112	0.00	673	19.01	4.80	23.67
EX51	60	87	641	0.02	6.75	87	0.00	441	31.20	4.09	16.11
EX61	96	118	901	1.73	12.73	118	0.00	582	35.41	0.08	24.84
EX71	76	116	1387	4.46	13.50	115	0.86	882	36.41	10.59	28.41
EX81	146	161	1338	0.03	13.77	161	0.00	597	55.38	22.35	30.72
EX91	93	116	807	1.43	10.52	116	0.00	589	27.01	3.87	21.79
EX101	124	148	1144	3.93	14.97	149	-0.68	788	31.12	9.98	28.99
EX12	68	82	520	0.08	5.78	82	0.00	370	28.85	0.12	15.28
EX22	76	76	553	1.93	7.78	76	0.00	373	32.55	3.95	18.95
EX32	75	85	663	0.00	7.84	85	0.00	407	38.61	8.84	19.54
EX42	64	87	646	3.61	10.82	87	0.00	532	17.65	14.02	22.97
EX52	59	69	460	0.01	6.18	69	0.00	338	26.52	5.78	15.59
EX62	86	98	725	0.15	11.70	98	0.00	424	41.52	4.96	23.93
EX72	74	84	894	0.93	13.00	84	0.00	532	40.49	26.31	27.71
EX82	140	151	1206	0.16	12.75	151	0.00	408	66.17	5.72	38.19
EX92	91	102	657	1.81	10.21	102	0.00	496	24.51	16.89	21.40
EX102	114	135	1049	0.16	14.58	136	-0.74	695	33.75	5.78	28.35
EX13	66	84	583	0.02	5.93	84	0.00	412	29.33	0.02	15.77
EX23	82	86	687	0.07	8.16	86	0.00	388	43.52	14.06	19.80
EX33	77	86	675	0.05	7.65	86	0.00	401	40.59	5.94	19.89
EX43	66	89	648	4.07	10.63	89	0.00	503	22.38	0.57	23.03
EX53	57	74	500	1.07	6.47	74	0.00	299	40.20	4.82	16.20
EX63	88	103	777	8.77	11.50	103	0.00	430	44.66	18.91	24.11
EX73	76	89	957	10.93	13.06	88	1.12	513	46.39	25.81	28.26
EX83	142	153	1232	0.05	12.65	153	0.00	422	65.75	20.80	36.85
EX93	93	105	677	1.09	10.09	105	0.00	528	22.01	5.06	21.83
EX103	116	140	1151	14.37	14.49	139	0.71	679	41.01	1.98	28.43
EX14	68	103	744	0.16	6.52	103	0.00	528	29.03	0.88	16.12
EX24	84	108	871	3.84	8.89	108	0.00	511	41.33	1.48	19.62
EX34	84	112	964	6.42	9.19	111	0.89	597	38.07	12.15	20.08
EX44	76	125	980	9.92	11.36	124	0.80	715	27.04	22.00	23.39
EX54	56	97	746	0.19	6.97	97	0.00	519	30.43	15.19	16.21
EX64	90	121	998	10.44	12.94	120	0.83	584	41.48	12.27	24.78
EX74	76	135	1566	5.42	13.69	134	0.74	859	45.15	13.35	27.99
EX84	148	163	1351	1.14	14.39	163	0.00	697	48.41	11.16	29.48
EX94	91	120	829	0.40	10.70	120	0.00	584	29.55	4.48	21.76
EX104	120	159	1253	3.55	15.67	162	-1.89	965	22.98	15.62	29.24
AVG		**109.45**	**873.78**	**2.83**	**10.50**	**109.30**	**0.16**	**546.08**	**35.78**	**9.38**	**23.14**

Pour le dataset $D1$ (Tableau 16), la durée moyenne nécessaire pour trouver la meilleure solution du JSPT est 2.83 secondes et la durée totale est de 10.50 secondes. La durée moyenne du JSPR est de 9.38 secondes pour trouver la meilleure solution et 23.14 secondes pour la résolution complète. Les différences entre ces durées s'expliquent par le nombre d'évaluations du graphe qui est beaucoup plus important pour le JSPR. Les mêmes remarques restent vraies pour le dataset $D2$ (Tableau 17).

Tableau 17 JSPT et JSPR intégré pour D2 : véhicules de capacité unitaire

	Ulusoy LB	Résolution du JSPT : $F = h_{cmax}(y)$				Résolution du JSPR intégré : $F = 10\,000 \times h_{cmax}(x) + h_{cost}(x)$					
		$h^{i,*}_{Cmax}(y)$	$h^{i,*}_{cost}(y)$	$st^{i,*}$	$stt^{i,*}$	$h^{i,*}_{Cmax}(x)$	$\Delta^{i,*}_{Cmax}$	$h^{i,*}_{cost}(x)$	$\Delta^{i,*}_{cost}$	$st^{i,*}$	$stt^{i,*}$
EX110	126	126	729	0.00	4.33	126	0.00	516	29.22	0.09	18.24
EX210	148	148	1062	0.02	6.47	148	0.00	538	49.34	6.65	25.60
EX310	138	150	1076	0.01	6.20	150	0.00	642	40.33	0.96	27.05
EX410	112	119	774	0.21	7.74	119	0.00	632	18.35	0.72	23.62
EX510	102	102	642	0.00	4.38	102	0.00	360	43.93	0.80	18.69
EX610	163	186	1387	0.01	9.73	186	0.00	676	51.26	1.68	30.06
EX710	137	146	1661	0.01	9.95	146	0.00	536	67.73	38.76	40.25
EX810	271	292	2395	0.01	11.43	292	0.00	684	71.44	2.99	53.61
EX910	150	176	988	0.19	8.44	176	0.00	678	31.38	1.49	25.41
EX1010	218	238	1637	1.21	11.96	238	0.00	942	42.46	17.17	33.92
EX120	123	123	715	0.00	4.11	123	0.00	500	30.07	0.02	21.16
EX220	143	143	1072	0.01	6.39	143	0.00	527	50.84	9.02	27.51
EX320	135	145	1057	0.01	6.09	145	0.00	606	42.67	4.53	29.53
EX420	111	114	736	0.11	7.62	114	0.00	614	16.58	3.26	24.76
EX520	99	100	597	0.00	4.08	100	0.00	338	43.38	2.25	22.67
EX620	160	181	1392	0.07	9.64	181	0.00	624	55.17	4.21	31.16
EX720	136	143	1640	0.09	9.64	143	0.00	484	70.49	33.54	42.10
EX820	268	287	2440	0.00	11.28	287	0.00	616	74.75	2.44	54.33
EX920	150	173	1015	0.15	8.44	173	0.00	650	35.96	4.11	25.62
EX1020	216	236	1638	1.56	11.98	236	0.00	902	44.93	10.96	34.38
EX130	122	122	694	0.00	4.17	122	0.00	498	28.24	0.25	19.07
EX230	146	146	1171	0.00	6.21	146	0.00	488	58.33	23.22	27.48
EX330	136	146	1082	0.00	6.00	146	0.00	610	43.62	0.12	29.13
EX430	110	114	732	0.09	7.40	114	0.00	594	18.85	1.70	24.09
EX530	98	99	607	0.00	4.19	99	0.00	338	44.32	0.19	19.04
EX630	161	182	1350	0.06	9.57	182	0.00	636	52.89	6.27	30.05
EX730	137	144	1690	0.06	9.75	144	0.00	500	70.41	31.50	42.05
EX830	269	288	2553	0.01	11.44	288	0.00	628	75.40	7.43	54.89
EX930	151	174	998	0.07	8.37	174	0.00	660	33.87	0.78	26.15
EX1030	217	237	1573	1.32	11.91	237	0.00	924	41.26	33.13	34.27
EX140	124	124	734	0.00	4.55	124	0.00	534	27.25	0.08	16.69
EX241	217	217	1713	0.00	6.42	217	0.00	731	57.33	18.17	25.71
EX340	138	151	1064	0.00	6.34	151	0.00	632	40.60	1.21	25.30
EX341	203	221	1668	0.01	6.09	221	0.00	911	45.38	0.70	28.01
EX441	166	172	1101	0.16	7.75	172	0.00	889	19.26	0.31	24.06
EX541	148	148	911	0.00	4.16	148	0.00	497	45.44	0.54	20.61
EX640	161	184	1369	0.13	9.89	184	0.00	654	52.23	12.78	28.66
EX740	137	145	1546	0.13	10.39	145	0.00	528	65.85	26.30	34.64
EX741	203	214	2243	0.16	9.56	214	0.00	726	67.63	26.69	40.25
EX840	272	293	2589	0.00	11.61	293	0.00	688	73.43	17.00	50.71
EX940	149	175	1009	0.10	8.65	175	0.00	668	33.80	0.42	24.97
EX1040	219	240	1625	0.43	12.21	240	0.00	944	41.91	3.26	32.48
AVG		**172.95**	**1301.79**	**0.15**	**8.01**	**172.95**	**0.00**	**627.21**	**46.37**	**8.52**	**30.19**

5.3.2. Capacité non unitaire

Similairement à la section précédente, le Tableau 18 et le Tableau 19 introduisent les meilleures réplications parmi les cinq réplications du GRASP×ELS pour le JSPT et le JSPR intégré. Dans cette section, les véhicules ont des capacités non unitaires, chacun pouvant transporter éventuellement deux jobs simultanément.

Pour le dataset *D*1 (Tableau 18), l'écart moyen des makespans, $\overline{\Delta^{i,*}_{Cmax}} = 0.06\%$, signifie que le JSPR intégré est en moyenne très légèrement meilleur que le JSPT concernant le critère du makespan. C'est notamment le cas pour l'instance EX104 où le JSPT obtient un makespan de 144 et le JSPR de 141. Pour d'autres instances, le cas inverse se produit, par

exemple pour l'instance EX102, le JSPT a un makespan de 129 tandis que le JSPR intégré à un makespan de 131.

Concernant le critère de QoS, $\overline{\Delta_{cost}^{i,*}} = 26.00\%$, signifie que le JSPR intégré trouve des solutions en moyenne 26% meilleures que le JSPT pour le critère de la QoS. Les temps de calcul moyen du JSPT sont $\overline{st^{i,*}} = 3.61$ secondes et $\overline{stt^{i,*}} = 18.66$ secondes, et pour le JSPR $\overline{st^{i,*}} = 14.51$ secondes et $\overline{stt^{i,*}} = 32.19$ secondes. La résolution du JSPR requiert à peu près deux fois plus de temps que la résolution du JSPT, mais elle permet d'obtenir des solutions de qualité très nettement supérieure en termes de QoS, sans dégrader le critère du makespan.

Tableau 18 JSPT et JSPR intégré pour $D1$: véhicules de capacité non unitaire

	Ulusoy LB	**Résolution du JSPT : $F = h_{cmax}(y)$**				**Résolution du JSPR intégré : $F = 10\,000 \times h_{cmax}(x) + h_{cost}(x)$**					
		$h_{Cmax}^{i,*}(y)$	$h_{cost}^{i,*}(y)$	$st^{i,*}$	$stt^{i,*}$	$h_{Cmax}^{i,*}(x)$	$\Delta_{Cmax}^{i,*}$	$h_{cost}^{i,*}(x)$	$\Delta_{cost}^{i,*}$	$st^{i,*}$	$stt^{i,*}$
EX11	/	82	547	0.40	12.41	82	0.00	494	9.69	0.26	22.54
EX21	/	87	668	2.38	15.82	87	0.00	532	20.36	1.60	27.02
EX31	/	90	719	2.26	17.44	90	0.00	593	17.52	24.77	28.91
EX41	/	95	729	6.69	26.32	95	0.00	652	10.56	8.81	39.28
EX51	/	71	531	0.14	13.35	71	0.00	449	15.44	8.65	22.82
EX61	/	108	819	1.20	21.23	108	0.00	645	21.25	22.84	33.63
EX71	/	86	1005	2.43	26.69	86	0.00	887	11.74	3.64	40.22
EX81	/	161	1260	0.01	20.44	161	0.00	588	53.33	37.00	45.17
EX91	/	106	683	1.67	19.41	106	0.00	608	10.98	12.07	30.53
EX101	/	138	965	15.05	26.33	138	0.00	801	16.99	19.91	40.64
EX12	/	76	465	0.01	10.17	76	0.00	380	18.28	18.66	21.43
EX22	/	76	592	0.08	12.30	76	0.00	373	36.99	0.51	24.81
EX32	/	80	577	0.02	12.87	80	0.00	399	30.85	2.13	26.13
EX42	/	74	489	3.64	23.31	74	0.00	447	8.59	4.42	35.49
EX52	/	64	432	0.47	11.25	64	0.00	321	25.69	12.40	21.84
EX62	/	98	741	0.81	18.09	98	0.00	425	42.65	19.47	31.43
EX72	/	79	826	2.15	21.85	79	0.00	476	42.37	10.89	36.80
EX82	/	151	1250	0.00	18.15	151	0.00	434	65.28	18.37	48.02
EX92	/	98	622	1.19	16.61	98	0.00	477	23.31	20.42	28.55
EX102	/	129	892	14.62	23.38	131	-1.55	677	24.10	24.58	38.74
EX13	/	74	432	0.01	10.35	74	0.00	378	12.50	4.02	21.00
EX23	/	82	681	0.06	13.50	82	0.00	375	44.93	22.65	25.68
EX33	/	82	567	0.07	12.99	82	0.00	407	28.22	0.38	25.99
EX43	/	74	524	4.51	22.68	74	0.00	429	18.13	27.61	34.77
EX53	/	63	405	0.07	11.76	63	0.00	311	23.21	11.28	21.76
EX63	/	100	759	0.50	18.70	100	0.00	468	38.34	27.95	31.56
EX73	/	81	911	1.60	21.94	81	0.00	525	42.37	25.29	36.97
EX83	/	153	1221	0.01	19.28	153	0.00	450	63.14	5.47	47.19
EX93	/	100	653	1.32	17.35	100	0.00	497	23.89	0.94	28.41
EX103	/	133	1002	4.20	23.76	133	0.00	716	28.54	32.30	38.82
EX14	/	84	571	0.26	12.88	84	0.00	486	14.89	1.63	22.35
EX24	/	88	662	13.22	16.07	88	0.00	531	19.79	26.72	27.36
EX34	/	92	755	1.13	17.50	92	0.00	571	24.37	19.18	28.91
EX44	/	98	724	25.55	27.04	98	0.00	690	4.70	3.55	38.89
EX54	/	73	502	0.01	13.73	73	0.00	421	16.14	5.95	23.28
EX64	/	106	811	2.48	22.35	104	1.89	608	25.03	28.09	34.17
EX74	/	90	1036	6.27	26.43	90	0.00	820	20.85	26.41	40.61
EX84	/	163	1336	0.03	21.60	163	0.00	647	51.57	26.50	42.81
EX94	/	105	678	4.94	20.06	105	0.00	578	14.75	5.96	30.88
EX104	/	144	1112	23.19	29.07	141	2.08	905	18.62	7.17	42.32
AVG	/	**98.35**	**753.85**	**3.61**	**18.66**	**98.28**	**0.06**	**536.78**	**26.00**	**14.51**	**32.19**

Pour le dataset $D2$ (Tableau 19), $\overline{\Delta_{Cmax}^{i,*}} = 0.00\%$ signifie que le JSPT et le JSPR intégré obtiennent des solutions avec un makespan équivalent. Des remarques similaires aux résultats numériques obtenus avec le dataset $D1$ peuvent être établies pour les résultats obtenus à partir du dataset $D2$. L'écart moyen de la QoS, pour le dataset $D2$, est $\overline{\Delta_{cost}^{i,*}} = 45.60\%$, et reflète

l'amélioration apportée par le JSPR intégré pour le critère de la QoS comparé à une résolution du JSPT. Les temps de calcul du JSPR intégré sont néanmoins beaucoup plus grands que pour le JSPT, par exemple pour l'instance EX110, le $h^{i,*}_{Cmax}(y)$ obtenu par résolution du JSPT est 126, et le $h^{i,*}_{cost}(y)$ est 742 avec un temps total de calcul de $stt^{i,*} = 6.79$ secondes, la résolution du JSPR permet d'obtenir une solution de $h^{i,*}_{Cmax}(x) = 126$, soit un gap entre les deux approches de $\Delta^{i,*}_{Cmax} = 0.00\%$, le $h^{i,*}_{cost}(x) = 516$ ce qui est $\Delta^{i,*}_{cost} = 30.46\%$ meilleur que celui du JSPT. Néanmoins le temps de résolution total est de $stt^{i,*} = 28.93$ secondes.

Tableau 19 JSPT et JSPR intégré pour D2 : véhicules de capacité non unitaire

	Ulusoy LB	**Résolution du JSPT : $F = h_{cmax}(y)$**				**Résolution du JSPR intégré : $F = 10\,000 \times h_{cmax}(x) + h_{cost}(x)$**					
		$h^{i,*}_{Cmax}(y)$	$h^{i,*}_{cost}(y)$	$st^{i,*}$	$stt^{i,*}$	$h^{i,*}_{Cmax}(x)$	$\Delta^{i,*}_{Cmax}$	$h^{i,*}_{cost}(x)$	$\Delta^{i,*}_{cost}$	$st^{i,*}$	$stt^{i,*}$
EX110	/	126	742	0.00	6.79	126	0.00	516	30.46	0.26	28.93
EX210	/	148	1072	0.00	9.17	148	0.00	538	49.81	5.93	32.74
EX310	/	150	1085	0.03	9.67	150	0.00	642	40.83	17.12	37.10
EX410	/	116	726	0.03	14.31	116	0.00	654	9.92	0.13	34.46
EX510	/	102	625	0.00	6.55	102	0.00	360	42.40	0.52	28.14
EX610	/	186	1487	0.02	13.84	186	0.00	676	54.54	3.26	37.00
EX710	/	146	1483	0.06	13.64	146	0.00	548	63.05	17.59	48.92
EX810	/	292	2540	0.00	14.49	292	0.00	684	73.07	5.26	57.71
EX910	/	174	1055	1.58	13.16	174	0.00	710	32.70	0.04	32.23
EX1010	/	238	1664	1.25	18.08	238	0.00	942	43.39	8.05	43.34
EX120	/	123	664	0.00	6.28	123	0.00	500	24.70	0.23	26.33
EX220	/	143	1030	0.00	8.78	143	0.00	522	49.32	8.56	30.28
EX320	/	145	1068	0.00	8.92	145	0.00	606	43.26	4.76	34.18
EX420	/	114	720	0.01	13.27	114	0.00	604	16.11	0.80	33.90
EX520	/	100	620	0.00	6.25	100	0.00	338	45.48	0.85	28.70
EX620	/	181	1319	0.43	13.08	181	0.00	624	52.69	22.18	34.17
EX720	/	143	1633	0.03	12.71	143	0.00	501	69.32	26.55	45.82
EX820	/	287	2373	0.00	14.22	287	0.00	616	74.04	24.21	56.05
EX920	/	173	1070	0.08	12.18	173	0.00	650	39.25	3.79	30.58
EX1020	/	236	1624	1.08	17.09	236	0.00	896	44.83	2.11	40.47
EX130	/	122	706	0.00	6.60	122	0.00	498	29.46	0.18	25.18
EX230	/	146	1082	0.00	8.90	146	0.00	490	54.71	21.32	29.87
EX330	/	146	1062	0.04	8.94	146	0.00	610	42.56	5.24	33.13
EX430	/	113	773	0.09	12.94	113	0.00	606	21.60	0.31	32.33
EX530	/	99	643	0.02	6.64	99	0.00	336	47.74	0.87	26.63
EX630	/	182	1399	0.14	13.26	182	0.00	636	54.54	7.89	33.65
EX730	/	144	1500	0.01	12.99	144	0.00	516	65.60	35.45	45.50
EX830	/	288	2464	0.01	14.19	288	0.00	628	74.51	21.93	55.28
EX930	/	174	944	0.34	12.17	174	0.00	660	30.08	2.66	30.08
EX1030	/	237	1663	2.35	17.45	237	0.00	908	45.40	2.24	39.52
EX140	/	124	679	0.00	7.30	124	0.00	524	22.83	0.35	28.00
EX241	/	217	1486	0.00	9.18	217	0.00	739	50.27	14.67	32.08
EX340	/	148	1042	0.07	9.97	148	0.00	664	36.28	2.94	33.66
EX341	/	218	1658	0.08	9.56	218	0.00	963	41.92	0.56	36.10
EX441	/	169	1046	0.00	13.87	169	0.00	895	14.44	0.70	36.29
EX541	/	148	943	0.00	6.35	148	0.00	497	47.30	0.12	30.39
EX640	/	184	1366	0.33	14.49	184	0.00	654	52.12	18.37	37.02
EX740	/	145	1621	0.08	14.64	145	0.00	526	67.55	23.61	45.34
EX741	/	214	2410	0.04	13.32	214	0.00	726	69.88	19.69	48.55
EX840	/	293	2717	0.02	15.09	293	0.00	688	74.68	15.72	57.81
EX940	/	173	1005	0.09	13.47	173	0.00	680	32.34	0.47	31.89
EX1040	/	237	1580	1.89	18.95	237	0.00	942	40.38	20.70	43.26
AVG		**172.48**	**1294.98**	**0.24**	**11.73**	**172.48**	**0.00**	**631.26**	**45.60**	**8.77**	**36.97**

5.3.3. Conclusion de la seconde étude

La seconde étude met en évidence l'importance de la qualité de service lors du schéma d'optimisation et l'impact de la qualité de service pour une solution. Cette QoS peut être grandement améliorée par la résolution d'un JSPR intégré sans perte de qualité en termes de makespan. Néanmoins, le JSPR intégré nécessite un temps de calcul plus important. Ces remarques restent vraies quel que soit le dataset considéré et quelle que soit la capacité des véhicules (unitaire ou non unitaire).

5.4. Troisième étude : paramétrage et optimalité de l'évaluation TLH

La troisième étude porte sur le paramétrage et l'optimalité de la fonction d'évaluation TLH. La fonction TLH étant une heuristique qui donne une situation de compromis entre les trois critères de la QoS, il est important de comparer les résultats obtenus avec une évaluation optimale grâce à un solveur de type PLNE. Cette analyse est détaillée dans la section 5.4.1.

Par ailleurs, l'ordre d'insertion des time-lags maximaux durant l'évaluation TLH est arbitraire et peut influencer la valeur de la QoS. L'ordre d'insertion des time-lags maximaux fait l'objet de l'étude de la section 5.4.2.

5.4.1. Optimalité de l'évaluation TLH

Afin d'évaluer la qualité de l'évaluation TLH, une évaluation exacte est fournie pour la meilleure solution trouvée par le JSPR. L'évaluation exacte est réalisée par CPLEX ayant comme objectif la maximisation de la qualité de service. En entrée, CPLEX reçoit un graphe disjonctif orienté acyclique : les disjonctions de machines sont des contraintes (ce qui signifie que l'ordre des jobs sur les machines est défini et non une variable) et le makespan est limité par celui de la solution fournie par l'évaluation TLH.

Afin de permettre une comparaison équitable, le Total Duration, le Total Riding Time et le Total Waiting Time sont ordonnés similairement à l'évaluation TLH. La fonction objectif de CPLEX est donc $F(x) = 10\,000 \times TD(x) + 100 \times TRT(x) + TWT(x)$, signifiant que le TD est minimisé en premier, le TRT en second et le TWT en troisième. Cette fonction objectif permet « d'imiter » le principe de l'évaluation TLH.

L'évaluation exacte avec CPLEX pourrait être appliquée à la meilleure solution trouvée pour le JSPR comme un processus de post-optimisation. Son efficacité est évaluée dans le Tableau 20 et le Tableau 21, respectivement dans le cas de plusieurs véhicules de capacité unitaire et dans le cas de véhicules de capacité non unitaire, et ceci pour les deux datasets.

$\Delta_{cost}^{i,*}$ est l'écart entre $h_{cost}^{i,*}(x)$ et $h_{cost}^{i,*}(y)$, où x est une solution obtenue par l'évaluation TLH et y est la solution obtenue en post-optimisation par CPLEX (uniquement maximisation de la qualité de service et où le makespan est borné). L'écart, pour la QoS, entre la solution CPLEX et la solution obtenue par la TLH est $\Delta_{cost}^{i,*} = (h_{cost}^{i,*}(x) - h_{cost}^{i,*}(y))/h_{cost}^{i,*}(y)$ et la valeur moyenne pour l'ensemble des instances d'un dataset est : $\overline{\Delta_{cost}^{i,*}}$.

Comme le montre le Tableau 20, la post-optimisation réalisée par CPLEX ne parvient pas à améliorer significativement la qualité de service de la solution trouvée par l'évaluation TLH. Pour le dataset $D1$, avec une flotte de véhicules de capacité unitaire (Tableau 20), les valeurs moyennes du makespan sont identiques ($\overline{h_{Cmax}^{Z,*}(x)} = \overline{h_{Cmax}^{Z,*}(y)} = 109.30$), ce qui est normal, car l'évaluation TLH est optimale concernant le critère du makespan pour une disjonction donnée des opérations. L'amélioration moyenne du coût est seulement de $\overline{\Delta_{cost}^{I,*}} = 0.08\%$ pour le dataset $D1$ (Tableau 20), et pour le dataset $D2$ (véhicules de capacité unitaire, Tableau 20), $\overline{\Delta_{cost}^{I,*}} = 0.02\%$. L'écart maximal est $\Delta_{cost}^{I,*} = 1.04\%$ pour le dataset $D1$, et 0.40% pour le dataset $D2$. Ces écarts sont très petits et démontrent la difficulté de trouver une solution meilleure que celle déjà fournie par l'évaluation TLH. Cette évaluation est proche d'être optimale.

Concernant le cas avec plusieurs véhicules de capacité non unitaire (Tableau 21), $\overline{\Delta_{cost}^{I,*}} = 0.05\%$ et $\overline{\Delta_{cost}^{I,*}} = 0.03\%$ respectivement pour le dataset $D1$ et le dataset $D2$. Les écarts maximaux sont de 1.01% pour le dataset $D1$ et de 0.60% pour le dataset $D2$. Encore une fois, ces écarts très petits démontrent la capacité à l'évaluation TLH d'obtenir des solutions très proches de celles optimales.

Tableau 20 Post-optimisation avec CPLEX, véhicules de capacité unitaire

	Résolution du JSPR : $F = 10\,000 \times h_{cmax}(x) + h_{cost}(x)$		**Post-optimisation avec CPLEX : $F = h_{cost}(y)$**			
	$\overline{h_{cmax}^{Z,*}(x)}$	$\overline{h_{cost}^{Z,*}(x)}$	$\overline{h_{cmax}^{Z,*}(y)}$	$\overline{h_{cost}^{Z,*}(y)}$	$\overline{\Delta_{cost}^{I,*}}$	**Écart maximal**
$D1$	109.30	546.08	109.30	545.63	0.08%	1.04%
$D2$	172.95	627.21	172.95	627.12	0.02%	0.40%

Tableau 21 Post-optimisation avec CPLEX, véhicules de capacité non unitaire

	Résolution du JSPR : $F = 10\,000 \times h_{cmax}(x) + h_{cost}(x)$		**Post-optimisation avec CPLEX : $F = h_{cost}(y)$**			
	$\overline{h_{cmax}^{Z,*}(x)}$	$\overline{h_{cost}^{Z,*}(x)}$	$\overline{h_{cmax}^{Z,*}(y)}$	$\overline{h_{cost}^{Z,*}(y)}$	$\overline{\Delta_{cost}^{I,*}}$	**Écart maximal**
$D1$	98.28	536.78	98.28	536.55	0.05	1.01%
$D2$	172.48	631.26	172.48	631.10	0.03	0.60%

5.4.2. Influence de l'ordre d'insertion des time-lags

L'évaluation TLH est une fonction heuristique dépendante de l'ordre d'insertion des time-lags maximaux et elle est basée sur un processus glouton qui minimise de façon itérative chaque time-lag maximal en considérant les time-lags déjà insérés dans le graphe sans les remettre en question : ces deux aspects définissent une approche heuristique. Par exemple, lors de la deuxième étape de l'évaluation TLH – étape concernant la minimisation du Total Duration (TD) – l'approche proposée insère les time-lags maximaux suivant l'ordre lexicographique (voir section 4.6), mais l'ajout de ceux-ci par une autre séquence d'insertion pourrait conduire à une autre solution, éventuellement meilleure.

Afin d'évaluer l'impact de l'ordre d'insertion des time-lags maximaux, cette étude comparative se focalise sur la seconde étape de l'évaluation TLH (concernant la minimisation du Total Duration). Soit un graphe $G = (V, E)$ obtenu à partir de l'instance EX64 avec deux véhicules de capacité unitaire. Le graphe G est soumis, d'une part, à une évaluation optimale

de sa QoS avec le solveur CPLEX et, d'autre part, à 10 évaluations TLH dont la séquence d'insertions des time-lags maximaux est aléatoire lors de la seconde étape. Les disjonctions des opérations-machines, ainsi que les affectations des véhicules aux opérations de transport et les disjonctions des opérations de transport sont fixées.

Afin de fournir une étude comparative équitable, le Total Duration, le Total Riding Time et le Total Waiting Time sont ordonnés dans l'évaluation exacte par CPLEX de la même façon que dans l'évaluation TLH. La fonction à minimiser de CPLEX est donc : $F(x) = 10000 \times TD(x) + 100 \times TRT(x) + TWT(x)$, avec x une solution, $TD(x)$ le Total Duration de x, $TRT(x)$ le Total Riding Time de x et $TWT(x)$ le Total Waiting Time de x.

La première ligne du Tableau 22 est l'évaluation du graphe de l'instance EX64 par CPLEX : le makespan (colonne $h_{Cmax}(x)$) est de 120 et le coût (colonne $h_{cost}(x)$) est de 578. L'évaluation TLH obtient pour chacune des 10 évaluations un makespan $h_{Cmax}(x) = 120$. De plus, l'évaluation TLH est capable de trouver $h_{cost}(x) = 578$ pour les évaluations 1, 2, 5, 7 et 10. L'écart type sur le coût est limité et vaut 3.13%. L'évaluation TLH est donc capable de trouver la solution optimale et reste dans le pire cas très proche de celle-ci.

Tableau 22 Impact de l'ordre d'insertion des TL max durant la minimisation du TD pour l'instance EX64

	$h_{Cmax}(x)$	$h_{cost}(x)$	$TD(x)$	$TRT(x)$	$TWT(x)$
Évaluation CPLEX	120	578	405	104	69
Évaluation n°1	120	578	405	104	69
Évaluation n°2	120	578	405	104	69
Évaluation n°3	120	584	408	104	72
Évaluation n°4	120	584	408	104	72
Évaluation n°5	120	578	405	104	69
Évaluation n°6	120	584	408	104	72
Évaluation n°7	120	578	405	104	69
Évaluation n°8	120	584	408	104	72
Évaluation n°9	120	584	408	104	72
Évaluation n°10	120	578	405	104	69
Moyenne :	**120.00**	**580.73**	**406.36**	**104.00**	**70.36**
Écart type :	**0.00%**	**3.13%**	**1.57%**	**0.00%**	**1.57%**

Plusieurs évaluations numériques similaires ont été menées concernant l'ordre d'insertion des time-lags maximaux durant les étapes de minimisation du Total Riding Time (TRT) et de minimisation du Total Waiting Time (TWT). Des évaluations numériques ont également été conduites dans le cas de véhicules à capacité non unitaire. Les résultats obtenus sont similaires à ceux présentés et débouchent sur les mêmes synthèses.

En conclusion, l'ordre d'insertion des time-lags maximaux au cours des différentes étapes de la fonction d'évaluation, avec une flotte de véhicules de capacité unitaire ou avec plusieurs véhicules de capacité non unitaire, n'est pas une caractéristique clé de l'algorithme et a une influence très limitée sur les résultats obtenus. Toutefois, l'ordre d'insertion peut être considéré comme un paramètre qui pourrait être ajusté en fonction des instances.

5.4.3. Conclusion de la troisième étude

Ces deux évaluations numériques démontrent que, d'une part, l'évaluation TLH est capable de trouver des solutions presque optimales concernant le critère de qualité de service ; et d'autre part, l'évaluation TLH est peu sensible à l'ordre d'insertion des time-lags maximaux au cours des différentes étapes de minimisation. Une étape de post-optimisation avec CPLEX n'a pas un impact important sur l'amélioration de la solution et les résultats prouvent qu'il est possible d'obtenir des solutions avec, simultanément, un makespan faible et une qualité de service élevée.

5.5. Quatrième étude : approche séquentielle et approche intégrée pour le JSPR

La quatrième étude porte sur les avantages et les inconvénients d'une approche intégrée par rapport à une approche séquentielle (Gondran et al., 2019). Lors d'une approche intégrée, la fonction objectif prend en compte la minimisation du makespan et la maximisation de la QoS, tandis que l'approche séquentielle ne prend en compte que la minimisation du makespan durant le déroulement l'heuristique, puis la QoS de la meilleure solution finale est maximisée (voir la Figure 34).

5.5.1. Capacité unitaire

La première étude concerne une comparaison entre la résolution du JSPT et la résolution du JSPR avec l'approche séquentielle pour une flotte de véhicules de capacité unitaire. Pendant l'approche séquentielle, la valeur des coefficients de la fonction objectif ($F(y) = \alpha \times h_{Cmax}(y) + \beta \times h_{cost}(y)$) sont $\alpha = 1$ et $\beta = 0$. L'évaluation TLH est dans ce cas équivalent à un algorithme de plus long chemin de type Bellmand-Ford (Ford and Lester, 1956; Bellman, 1958; Moore, 1959) ou Dijkstra (Dijkstra, 1959). À la fin du GRASP×ELS, l'évaluation TLH est appliquée à la meilleure solution trouvée avec $\alpha = 10\,000$ et $\beta = 1$, correspondant à la maximisation de la QoS. L'approche intégrée minimise le makespan et maximise la QoS à chaque itération, car les coefficients de la fonction objectif sont les suivants : $\alpha = 10\,000$ et $\beta = 1$.

Le Tableau 23 présente le makespan moyen $\overline{h_{Cmax}^{z,*}(y)}$, le coût moyen $\overline{h_{cost}^{z,*}(y)}$, le temps normalisé moyen $\overline{st^{z,*}(y)}$ pour obtenir la meilleure solution et le temps total normalisé de résolution $\overline{stt^{z,*}(y)}$, pour les deux datasets (respectivement $D1$ et $D2$) pour le JSPT et le JSPR séquentiel. Le makespan moyen est $\overline{h_{Cmax}^{z,*}(y)} = 109.45$ pour le JSPT et le JSPR séquentiel, la valeur est identique dans les deux cas, car la fonction objectif lors de la résolution du JSPT ou du JSPR séquentiel ne prend en compte que ce critère. L'écart moyen $\overline{\Delta_{Cmax}^{\iota,*}}$ entre les deux résolutions est donc nul. Les temps d'exécution $\overline{st^{z,*}(y)}$ et $\overline{stt^{z,*}(y)}$ sont également identiques : la durée d'une unique évaluation TLH n'est pas assez grande pour être significative sur les temps de calcul. $\overline{\Delta_{cost}^{\iota,*}}$ est l'écart moyen du critère du coût entre le JSPT et le JSPR séquentiel. $\overline{\Delta_{cost}^{\iota,*}} = 21.43\%$ pour le dataset $D1$ et $\overline{\Delta_{cost}^{\iota,*}} = 22.94\%$ pour le dataset $D2$. Ces écarts signifient que la qualité de service est bien meilleure pour une solution du JSPR séquentiel que pour une solution du JSPT et démontrent l'importance de l'utilisation de l'évaluation TLH pour obtenir des solutions de bonne qualité.

Tableau 23 JSPT et JSPR séquentiel pour les véhicules de capacité unitaire

	JSPT				**JSPR séquentiel**					
	$\overline{h_{cmax}^{z,*}(y)}$	$\overline{h_{cost}^{z,*}(y)}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$	$\overline{h_{cmax}^{z,*}(y)}$	$\overline{\Delta_{cmax}^{l,*}}$	$\overline{h_{cost}^{z,*}(y)}$	$\overline{\Delta_{cost}^{l,*}}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$
$D1$	109.45	873.95	2.83	10.50	109.45	0.00	688.00	21.43	2.83	10.50
$D2$	172.95	1301.79	0.15	8.01	172.95	0.00	995.40	22.94	0.15	8.01

La même étude est effectuée entre le JSPR séquentiel et le JSPR intégré et est présentée dans le Tableau 24. Pour le JSPR intégré, la fonction objectif utilisée durant le GRASP×ELS prend en compte la QoS et la fonction d'évaluation TLH est employée pour évaluer tous les graphes disjonctifs créés. L'écart moyen, $\overline{\Delta_{Cmax}^{l,*}}$, entre le makespan obtenu lors du JSPR séquentiel et celui du JSPR intégré est $\overline{\Delta_{max}^{l,*}} = 0.16\%$, pour le dataset $D1$, signifiant que le JSPR intégré est capable de trouver des solutions légèrement meilleures que celles du JSPR séquentiel (voir section 6.3.1). $\overline{\Delta_{Cmax}^{l,*}} = 0.00\%$ pour le dataset $D2$: les solutions du JSPR séquentiel et du JSPR intégré ont le même makespan moyen.

L'écart moyen du critère de coût $\overline{\Delta_{cost}^{l,*}} = 18.14\%$ pour le dataset $D1$, et $\overline{\Delta_{cost}^{l,*}} = 30.71\%$ pour le dataset $D2$. Le JSPR intégré est capable de trouver des solutions avec une qualité de service nettement meilleure que le JSPR séquentiel. Toutefois, le temps total d'exécution du JSPR intégré est $\overline{stt^{z,*}(y)} = 23.14$ secondes pour le dataset $D1$ et de $\overline{stt^{z,*}(y)} = 30.19$ secondes pour le dataset $D2$, contre $\overline{stt^{z,*}(y)} = 10.50$ et $\overline{stt^{z,*}(y)} = 8.01$ respectivement pour le dataset $D1$ et le dataset $D2$. Le temps moyen $\overline{st^{z,*}(y)}$ pour obtenir la meilleure solution est beaucoup plus petit pour le JSPR séquentiel, car ce temps est celui nécessaire pour obtenir la première solution trouvée ayant le meilleur makespan, tandis que pour le JSPR intégré, ce temps est celui nécessaire pour obtenir la solution avec le meilleur makespan et la meilleure QoS. Une solution avec le meilleur makespan a peut-être été trouvée plus tôt, mais elle ne possède pas forcément la meilleure QoS.

Tableau 24 JSPR séquentiel et JSPR intégré pour les véhicules de capacité unitaire

	JSPR séquentiel				**JSPR intégré**					
	$\overline{h_{cmax}^{z,*}(y)}$	$\overline{h_{cost}^{z,*}(y)}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$	$\overline{h_{cmax}^{z,*}(y)}$	$\overline{\Delta_{cmax}^{l,*}}$	$\overline{h_{cost}^{z,*}(y)}$	$\overline{\Delta_{cost}^{l,*}}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$
$D1$	109.45	688.00	2.83	10.50	109.30	0.16	546.08	18.14	9.38	23.14
$D2$	172.95	995.40	0.15	8.01	172.95	0.00	627.21	30.71	8.52	30.19

Cette étude met en évidence, encore une fois, l'importance de la maximisation du critère de la qualité de service. Le JSPR séquentiel permet d'améliorer le coût de la solution par rapport à une solution du JSPT, sans perte de temps. Le JSPR intégré obtient des solutions de meilleure QoS, mais nécessite un temps d'exécution plus long.

5.5.2. Capacité non unitaire

Cette section présente une étude similaire à la section précédente, mais dans le cas d'une flotte de véhicules de capacité non unitaire.

Le Tableau 25 présente une comparaison entre le JSPT et le JSPR séquentiel pour les deux datasets tandis que le Tableau 26 présente une comparaison entre le JSPR séquentiel et le JSPR intégré.

Le JSPT et le JSPR séquentiel ont un écart du makespan moyen $\overline{\Delta_{Cmax}^{l,*}} = 0.00\%$ pour les deux datasets et des temps de calcul identiques (Tableau 25). Néanmoins, le JSPR séquentiel et le JSPT ont un écart $\overline{\Delta_{cost}^{l,*}} = 25.82\%$ pour le dataset $D1$ et $\overline{\Delta_{cost}^{l,*}} = 19.65\%$ pour le dataset $D2$. Le JSPR séquentiel est capable, dans le cas d'une flotte de véhicules de capacité non unitaire, de trouver des solutions meilleures en termes de QoS que le JSPT, avec un makespan équivalent.

Tableau 25 JSPT et JSPR séquentiel pour les véhicules de capacité non unitaire

	JSPT				**JSPR séquentiel**					
	$\overline{h_{cmax}^{z,*}(y)}$	$\overline{h_{cost}^{z,*}(y)}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$	$\overline{h_{cmax}^{z,*}(y)}$	$\overline{\Delta_{cmax}^{l,*}}$	$\overline{h_{cost}^{z,*}(y)}$	$\overline{\Delta_{cost}^{l,*}}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$
$D1$	98.35	753.85	3.61	18.66	98.35	0.00	642.43	25.82	3.61	18.66
$D2$	172.48	1294.98	0.24	11.73	172.48	0.00	1030.69	19.65	0.24	11.73

L'écart moyen, entre le JSPR séquentiel et le JSPR intégré, en termes de makespan est très faible, respectivement $\overline{\Delta_{Cmax}^{l,*}} = 0.06\%$ et $\overline{\Delta_{Cmax}^{l,*}} = 0.00\%$, pour les datasets $D1$ et $D2$. Toutefois, l'écart en termes de QoS est significatif $\overline{\Delta_{cost}^{l,*}} = 14.38\%$ pour le dataset $D1$ et $\overline{\Delta_{cost}^{l,*}} = 33.31\%$ pour le dataset $D2$. Le temps de calcul pour le JSPR intégré est supérieur à celui du JSPR séquentiel : $\overline{stt^{z,*}(y)} = 32.19$ secondes pour le dataset $D1$ et $\overline{stt^{z,*}(y)} = 36.97$ secondes pour le dataset $D2$ contre $\overline{stt^{z,*}(y)} = 18.66$ secondes et $\overline{stt^{z,*}(y)} = 11.73$ secondes, respectivement pour les dataset $D1$ et $D2$ avec le JSPR séquentiel.

Tableau 26 JSPR séquentiel et JSPR intégré pour les véhicules de capacité non unitaire

	JSPR séquentiel				**JSPR intégré**					
	$\overline{h_{cmax}^{z,*}(y)}$	$\overline{h_{cost}^{z,*}(y)}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$	$\overline{h_{cmax}^{z,*}(y)}$	$\overline{\Delta_{cmax}^{l,*}}$	$\overline{h_{cost}^{z,*}(y)}$	$\overline{\Delta_{cost}^{l,*}}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$
$D1$	98.35	753.85	3.61	18.66	98.28	0.06	536.78	14.38	14.51	32.19
$D2$	172.48	1294.98	0.24	11.73	172.48	0.00	631.26	33.31	8.77	36.97

Similairement à la section 5.5.1, ces résultats numériques mettent en lumière l'apport du JSPR séquentiel par rapport à une résolution classique du JSPT : un gain de qualité de service non négligeable, sans détériorer le critère du makespan et sans augmenter le temps de calcul. Le JSPR intégré permet d'obtenir des solutions de meilleure qualité en termes de QoS, sans dégrader le critère du makespan, mais nécessite un temps de calcul plus long. Le JSPR séquentiel peut être un bon compromis entre le JSPT et le JSPR pour trouver des solutions de bonne qualité en un temps raisonnable.

6. Conclusion

Ce chapitre propose de prendre en compte un critère de qualité dans un problème d'ordonnancement avec routing. La grande majorité des problèmes combinant ordonnancement et transport ne prennent pas de tels critères en considération. Le problème de transport est souvent vu comme un « sous-problème » ou « une contrainte » du problème d'ordonnancement et alors une solution semi-active est suffisante. La prise en considération d'un critère de qualité de service impose d'obtenir des solutions qui ne sont plus semi-actives. La recherche d'une solution d'un problème d'ordonnancement avec routing place le transport sur un pied d'égalité avec l'ordonnancement. La qualité de service du routing est traitée avec des algorithmes issus de la communauté de tournées de véhicules, et notamment le célèbre algorithme de (Cordeau and Laporte, 2003) qui est adapté pour le Job-shop Scheduling Problem with Routing.

Ce chapitre présente une fonction d'évaluation d'un graphe disjonctif, nommée TLH (Time-Lag Heuristic), pour le Job-shop Scheduling Problem with Routing, qui peut être utilisée dans le cas d'une flotte de véhicules de capacité unitaire et dans le cas d'une flotte de capacité non unitaire.

La procédure TLH est une fonction d'évaluation d'un graphe disjonctif, et définit une solution qui n'est pas left-shifted, c'est-à-dire où les opérations ne commencent pas au plus tôt. La solution obtenue par TLH minimise le makespan et maximise la qualité de service. La fonction TLH est optimale pour le critère du makespan. Une définition de la qualité de service pour le JSPR est proposée dans ce chapitre, elle englobe le Total Duration, le Total Riding Time et le Total Waiting Time. L'évaluation TLH minimise itérativement ces trois critères.

L'évaluation TLH est incluse dans une métaheuristique de type GRASP×ELS et est testée numériquement sur les instances de la littérature proposées par (Bilge and Ulusoy, 1995). Ces instances sont étendues pour prendre en compte le cas d'une flotte de véhicules de capacité non unitaire.

Deux schémas algorithmiques utilisant l'évaluation TLH sont expérimentés : le premier schéma, nommé JSPR séquentiel, minimise – au cours du GRASP×ELS – uniquement le makespan, puis l'évaluation TLH est appliquée à la meilleure solution pour maximiser la qualité de service de celle-ci. Le second schéma, nommé JSPR intégré, utilise à chaque évaluation d'un graphe la fonction TLH, et le GRASP×ELS a pour fonction objectif la minimisation du makespan et la maximisation de la qualité de service.

Les différentes études montrent que l'utilisation de l'évaluation TLH, pour le JSPR, dans un GRASP×ELS permet de trouver des solutions comparables à la littérature du JSPT en termes de makespan pour une flotte de véhicules de capacité unitaire. Pour le cas d'une flotte de véhicules de capacité non unitaire, le JSPR séquentiel et le JSPR intégré (utilisant l'évaluation TLH) trouvent des solutions aussi bonnes en termes de makespan que le JSPT. Toutefois, un gain important et non négligeable est réalisé lors de l'utilisation de la procédure TLH pour le critère de la qualité de service. Le gain le plus important est pour le JSPR intégré, mais celui-ci est plus long en temps de calcul que le JSPR séquentiel. Cette dernière méthode a un temps de calcul quasiment identique au JSPT, tout en ayant des solutions avec une QoS meilleure que le JSPT.

Une autre série d'études établit, d'une part que l'évaluation TLH est peu sensible à l'ordre d'insertion des time-lags maximaux durant les différentes étapes de cette procédure ; et d'autre part, que cette évaluation est capable de fournir des résultats très proches des solutions optimales.

7. Bibliographie

Abdelmaguid, T.F., 2007. Integrated scheduling of manufacturing operations and material handling tasks in the job shops: A network model, in: Proceedings of the 37th International Conference on Computers and Industrial Engineering. Alexandria, Egypt. pp. 1951–1960.

Abdelmaguid, T.F., Nassef, A.O., 2010. A constructive heuristic for the integrated scheduling of machines and multiple-load material handling equipment in job shops. The

International Journal of Advanced Manufacturing Technology 46, 1239–1251. https://doi.org/10.1007/s00170-009-2176-7

Abdelmaguid, T.F., Nassef, A.O., Kamal, B.A., Hassan, M.F., Lacomme, P., 2004. A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research 42, 267–281. https://doi.org/10.1080/0020754032000123579

Adnen El, A., Elhafsi, M., 2016. An efficient new heuristic for the hoist scheduling problem. Computers & Operations Research 67, 184–192. https://doi.org/10.1016/j.cor.2015.10.006

Ahmadi-Javid, A., Hooshangi-Tabrizi, P., 2017. Integrating employee timetabling with scheduling of machines and transporters in a job-shop environment: A mathematical formulation and an Anarchic Society Optimization algorithm. Computers & Operations Research 84, 73–91. https://doi.org/10.1016/j.cor.2016.11.017

Artigues, C., 2017. On the strength of time-indexed formulations for the resource-constrained project scheduling problem. Operations Research Letters 45, 154–159. https://doi.org/10.1016/j.orl.2017.02.001

Artigues, C., Huguet, M.-J., Lopez, P., 2011. Generalized disjunctive constraint propagation for solving the job shop problem with time lags. Engineering Applications of Artificial Intelligence 24, 220–231. https://doi.org/10.1016/j.engappai.2010.07.008

Artigues, C., Michelon, P., Reusser, S., 2003. Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research 149, 249–267. https://doi.org/10.1016/S0377-2217(02)00758-0

Babu, A.G., Jerald, J., Noorul Haq, A., Muthu Luxmi, V., Vigneswaralu, T.P., 2010. Scheduling of machines and automated guided vehicles in FMS using differential evolution. International Journal of Production Research 48, 4683–4699. https://doi.org/10.1080/00207540903049407

Baruwa, O.T., Piera, M.A., 2016. A coloured Petri net-based hybrid heuristic search approach to simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research 54, 4773–4792. https://doi.org/10.1080/00207543.2015.1087656

Bellman, R., 1958. On a routing problem. Quarterly of applied mathematics 16, 87–90.

Bierwirth, C., 1995. A generalized permutation approach to job shop scheduling with genetic algorithms. Operations-Research-Spektrum 17, 87–92.

Bilge, Ü., Ulusoy, G., 1995. A Time window approach to simultaneous scheduling of machines and material handling system in an FMS. Operations research 43, 1058–1070.

Çaliş, B., Bulkan, S., 2015. A research survey: review of AI solution strategies of job shop scheduling problem. Journal of Intelligent Manufacturing 26, 961–973.

Carlier, J., Moukrim, A., Xu, H., 2009. The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm. Discrete Applied Mathematics 157, 3631–3642. https://doi.org/10.1016/j.dam.2009.02.012

Caumond, A., 2006. Le problème de jobshop avec contraintes : modélisation et optimisation (Thèse de doctorat). Université Blaise Pascal-Clermont-Ferrand II, Clermont-Ferrand.

Caumond, A., Lacomme, P., Moukrim, A., Tchernev, N., 2009. An MILP for scheduling problems in an FMS with one vehicle. European Journal of Operational Research 199, 706–722. https://doi.org/10.1016/j.ejor.2008.03.051

Caumond, A., Lacomme, P., Tchernev, N., 2008. A memetic algorithm for the job-shop with time-lags. Computers & Operations Research 35, 2331–2356. https://doi.org/10.1016/j.cor.2006.11.007

Cheng, R., Gen, M., Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation. Computers & industrial engineering 30, 983–997.

Chvátal, V., Cook, W., Dantzig, G.B., Fulkerson, D.R., Johnson, S.M., 2010. Solution of a Large-Scale Traveling-Salesman Problem, in: 50 Years of Integer Programming 1958-2008. Springer, Berlin, Heidelberg, pp. 7–28. https://doi.org/10.1007/978-3-540-68279-0_1

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Annals of Operations Research 153, 29–46. https://doi.org/10.1007/s10479-007-0170-8

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579–594. https://doi.org/10.1016/S0191-2615(02)00045-0

Dantzig, G.B., Fulkerson, R., Johnson, S., 1954. Solution of a Large-Scale Traveling-Salesman Problem. Journal of the Operations Research Society of America 2, 393–410. https://doi.org/10.1287/opre.2.4.393

Dell'Amico, M., Trubian, M., 1993. Applying tabu search to the job-shop scheduling problem. Annals of Operations research 41, 231–252.

Deroussi, L., Gourgand, M., Tchernev, N., 2008. A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research 46, 2143–2164. https://doi.org/10.1080/00207540600818286

Deroussi, L., Norre, S., 2010. Simultaneous scheduling of machines and vehicles for the flexible job shop problem, in: International Conference on Metaheuristics and Nature Inspired Computing. pp. 1–2.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische mathematik 1, 269–271.

El Khoukhi, F., Lamoudan, T., Boukachour, J., El Hilali Alaoui, A., 2011. Ant Colony Algorithm for Just-in-Time Job Shop Scheduling with Transportation Times and Multirobots. ISRN Applied Mathematics 2011, 1–19. https://doi.org/10.5402/2011/165620

Erol, R., Sahin, C., Baykasoglu, A., Kaplanoglu, V., 2012. A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems. Applied Soft Computing 12, 1720–1732. https://doi.org/10.1016/j.asoc.2012.02.001

Firat, M., Woeginger, G.J., 2011. Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. Operations Research Letters 39, 32–35. https://doi.org/10.1016/j.orl.2010.11.004

Ford, J., Lester, R., 1956. Network flow theory. Rand Corp Santa Monica Ca.

Gondran, M., Huguet, M.-J., Lacomme, P., Quilliot, A., Tchernev, N., 2018a. A Dial-a-Ride evaluation for solving the job-shop with routing considerations. Engineering Applications of Artificial Intelligence 74, 70–89. https://doi.org/10.1016/j.engappai.2018.05.010

Gondran, M., Huguet, M.-J., Lacomme, P., Tchernev, N., 2019. Comparison between two approaches to solve the Job-shop Scheduling Problem with Routing, in: 9th Manufacturing Modelling, Management and Control (MIM). Berlin, Germany.

Gondran, M., Lacomme, P., Tchernev, N., 2018b. A Dial-a-Ride evaluation for solving the Job-shop Problem with Transport. Presented at the 19ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Lorient, France,.

Gondran, M., Lacomme, P., Tchernev, N., 2017. Resolution of a Job-shop with routing considering a quality of service for customer: JSSP with routing, in: 7th International Conference on Industrial Engineering and Systems Management (IESM). Saarbrücken, Germany.

Grabowski, J., Nowiki, E., Zdrzałka, S., 1986. A block approach for single-machine scheduling with release dates and due dates. European Journal of Operational Research 26, 278–285.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey, in: Annals of Discrete Mathematics. Elsevier, pp. 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X

Honglin, Q., Sujing, W., Qiang, X., 2017. Simultaneous 2D hoist scheduling and production line design for multi-recipe and multi-stage material handling processes. Chemical Engineering Science 167, 251–264. https://doi.org/10.1016/j.ces.2017.04.022

Hurink, J., Knust, S., 2005. Tabu search algorithms for job-shop problems with a single transport robot. European journal of operational research 162, 99–111.

Hurink, J., Knust, S., 2002. A tabu search algorithm for scheduling a single robot in a job-shop environment. Discrete Applied Mathematics 119, 181–203.

Jain, A.S., Meeran, S., 1999. Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research 113, 390–434.

Kesen, S.E., Baykoç, Ö.F., 2007. Simulation of automated guided vehicle (AGV) systems based on just-in-time (JIT) philosophy in a job-shop environment. Simulation Modelling Practice and Theory 15, 272–284. https://doi.org/10.1016/j.simpat.2006.11.002

Khayat, G.E., Langevin, A., Riopel, D., 2006. Integrated production and material handling scheduling using mathematical programming and constraint programming. European Journal of Operational Research 175, 1818–1832. https://doi.org/10.1016/j.ejor.2005.02.077

Knust, S., 1999. Shop-scheduling problems with transportation (PhD thesis). Fachbereich Mathematik/Informatik Universitat Osnabruck.

Kumar, M.V.S., Janardhana, R., Rao, C.S.P., 2011. Simultaneous scheduling of machines and vehicles in an FMS environment with alternative routing. The International Journal of Advanced Manufacturing Technology 53, 339–351. https://doi.org/10.1007/s00170-010-2820-2

Lacomme, P., Larabi, M., Tchernev, N., 2013. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Economics 143, 24–34. https://doi.org/10.1016/j.ijpe.2010.07.012

Lacomme, P., Larabi, M., Tchernev, N., 2007. A disjunctive graph for the job-shop with several robots, in: MISTA Conference. pp. 285–292.

Larabi, M., 2010. Le problème de job-shop avec transport: modélisation et optimisation (Thèse de doctorat). Université Blaise Pascal-Clermont-Ferrand II.

Lenstra, J.K., Kan, A.H.G.R., 1981. Complexity of vehicle routing and scheduling problems. Networks 11, 221–227. https://doi.org/10.1002/net.3230110211

Lenstra, J.K., Kan, A.R., 1979. Computational complexity of discrete optimization problems. Annals of Discrete Mathematics 4, 121–140.

Lenstra, J.K., Kan, A.R., Brucker, P., 1977. Complexity of Machine Scheduling Problems, in: Annals of Discrete Mathematics. Elsevier, pp. 343–362. https://doi.org/10.1016/S0167-5060(08)70743-X

Manne, A.S., 1960. On the Job-Shop Scheduling Problem. Operations research 8, 219–223.

Moore, E.F., 1959. The shortest path through a maze, in: Proc. Int. Symp. Switching Theory 1959. pp. 285–292.

Morihiro, Y., Miyamoto, T., Kumagai, S., 2006. An initial task assignment method for autonomous distributed vehicle systems with finite buffer capacity, in: Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference On. IEEE, pp. 805–812.

Nageswararao, M., Narayanarao, K., Ranagajanardhana, G., 2014. Simultaneous Scheduling of Machines and AGVs in Flexible Manufacturing System with Minimization of Tardiness Criterion. Procedia Materials Science 5, 1492–1501. https://doi.org/10.1016/j.mspro.2014.07.336

Nouri, H.E., Belkahla Driss, O., Ghédira, K., 2016a. Simultaneous Scheduling of Machines and a Single Moving Robot in a Job Shop Environment by Metaheuristics based Clustered Holonic Multiagent Model: SCITEPRESS - Science and and Technology Publications, pp. 51–62. https://doi.org/10.5220/0005694300510062

Nouri, H.E., Belkahla Driss, O., Ghédira, K., 2016b. Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model. Computers & Industrial Engineering 102, 488–501. https://doi.org/10.1016/j.cie.2016.02.024

Nouri, H.E., Driss, O.B., Ghédira, K., 2016c. A Classification Schema for the Job Shop Scheduling Problem with Transportation Resources: State-of-the-Art Review. Artificial Intelligence Perspectives in Intelligent Systems 464, 1–11. https://doi.org/10.1007/978-3-319-33625-1_1

Nouri, H.E., Driss, O.B., Ghédira, K., 2016d. Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment. Applied Intelligence 45, 808–828. https://doi.org/10.1007/s10489-016-0786-y

Quilliot, A., Toussaint, H., 2012. Resource Constrained Project Scheduling with Transportation Delays. IFAC Proceedings Volumes 45, 1481–1486. https://doi.org/10.3182/20120523-3-RO-2023.00061

Reddy, B.S.P., Rao, C.S.P., 2006. A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS. The International Journal of Advanced Manufacturing Technology 31, 602–613. https://doi.org/10.1007/s00170-005-0223-6

Roy, B., Sussmann, B., 1964. Les problèmes d'ordonnancement avec contraintes disjonctives. Note Ds 9.

Sahin, C., Demirtas, M., Erol, R., Baykasoğlu, A., Kaplanoğlu, V., 2015. A multi-agent based approach to dynamic scheduling with flexible processing capabilities. Journal of Intelligent Manufacturing 28, 1827–1845. https://doi.org/10.1007/s10845-015-1069-x

Stein, D.M., 1978. Scheduling Dial-a-Ride Transportation Systems. Transportation Science 12, 232–249. https://doi.org/10.1287/trsc.12.3.232

Subbaiah, K.V., Rao, M.N., Rao, K.N., 2009. Scheduling of AGVs and machines in FMS with makespan criteria using sheep flock heredity algorithm. International Journal of Physical Sciences 4, 139–148.

Ulusoy, G., Sivrikaya-Şerifoğlu, F., Bilge, Ü., 1997. A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. Computers & Operations Research 24, 335–351.

Umar, U.A., Ariffin, M.K.A., Ismail, N., Tang, S.H., 2015. Hybrid multiobjective genetic algorithms for integrated dynamic scheduling and routing of jobs and automated-guided vehicle (AGV) in flexible manufacturing systems (FMS) environment. The International Journal of Advanced Manufacturing Technology 81, 2123–2141. https://doi.org/10.1007/s00170-015-7329-2

Van Laarhoven, P.J., Aarts, E.H., Lenstra, J.K., 1992. Job shop scheduling by simulated annealing. Operations research 40, 113–125.

Xie, C., Allen, T.T., 2015. Simulation and experimental design methods for job shop scheduling with material handling: a survey. The International Journal of Advanced Manufacturing Technology 80, 233–243. https://doi.org/10.1007/s00170-015-6981-x

Zeng, C., Tang, J., Yan, C., 2015. Job-shop cell-scheduling problem with inter-cell moves and automated guided vehicles. Journal of Intelligent Manufacturing 26, 845–859. https://doi.org/10.1007/s10845-014-0875-x

Zeng, C., Tang, J., Yan, C., 2014. Scheduling of no buffer job shop cells with blocking constraints and automated guided vehicles. Applied Soft Computing 24, 1033–1046. https://doi.org/10.1016/j.asoc.2014.08.028

Zhang, Q., Manier, H., Manier, M.-A., 2014. A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. International Journal of Production Research 52, 985–1002. https://doi.org/10.1080/00207543.2013.828164

Zhang, Q., Manier, H., Manier, M.-A., 2012. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Computers & Operations Research 39, 1713–1723. https://doi.org/10.1016/j.cor.2011.10.007

Zheng, K., Tang, D., Giret, A., Salido, M.A., Sang, Z., 2016. A hormone regulation-based approach for distributed and on-line scheduling of machines and automated guided vehicles. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 232, 99–113. https://doi.org/10.1177/0954405416662078

Zheng, Y., Xiao, Y., Seo, Y., 2014. A tabu search algorithm for simultaneous machine/AGV scheduling problem. International Journal of Production Research 52, 5748–5763. https://doi.org/10.1080/00207543.2014.910628

Chapitre III : Workforce Scheduling and Routing Problem

Ce chapitre aborde le Workforce Scheduling and Routing Problem (WSRP) qui est un problème de production et de transport où des visites (ou opérations) sont à effectuer en différents lieux géographiques. Le WSRP modélise notamment les problèmes de services à domicile et de maintenance. L'objectif du WSRP est d'affecter des employés à des visites chez des clients, puis de planifier les tournées de chaque employé. Le WSRP diffère des problèmes classiques de production et de transport par sa fonction objectif qui prend en compte à la fois un coût opérationnel et des critères de qualité de service pour les employés et les clients (Garaix et al., 2018a, 2018b, 2018c).

Une difficulté majeure du WSRP réside dans le critère de qualité de service de l'employé qui dépend de la date de début des visites. Une solution du WSRP est donc un ensemble de tournées dont les dates de visites ne sont pas nécessairement planifiées au plus tôt, afin de maximiser la qualité de service pour les clients et les employés.

Ce chapitre présente un modèle PLNE et un modèle PPC, une fonction d'évaluation d'une tournée maximisant la qualité de service de l'employé qui l'effectue, et une génération de colonnes avec le sous-problème résolu soit par PLNE soit par programmation dynamique.

1. Introduction

Il a été montré l'intérêt, lors des chapitres précédents, de la prise en considération de critères de qualité de service du point de vue du client lors de la recherche de solutions pour des problèmes intégrés d'ordonnancement et de transport. L'objet de ce chapitre est de s'intéresser à un problème d'ordonnancement et de transport qui intègre des considérations de qualité de service à la fois des clients mais aussi des employés.

Les problèmes de référence dans ce domaine sont les problèmes de workforce scheduling car la planification de la main-d'œuvre est un point crucial dans les secteurs industriels comme le soulignent (Mundschenk and Drexl, 2007; Saadat et al., 2013). Une gestion efficace des ressources humaines et des compétences est l'une des fonctions les plus critiques des entreprises et elle a un impact direct sur les performances de celles-ci (Wirojanagud et al., 2007; Attia et al., 2014).

La grande majorité des publications concernant la planification d'employés suppose que les travailleurs sont tous identiques (Wirojanagud et al., 2007). Toutefois, la prise en compte des individualités peut influencer les décisions de planification et de gestion des effectifs à un niveau stratégique (Maenhout and Vanhoucke, 2009; Wongmongkolrit and Rassameethes, 2010; Attia et al., 2014). Les différences entre les employés peuvent concerner les compétences, les coûts opérationnels, les horaires de travail, des préférences parmi les clients, etc. La polyvalence des employés en termes de compétences est importante pour les entreprises et est suggérée comme stratégie pour faire face à une pénurie de main-d'œuvre à long terme, en utilisant plus efficacement les travailleurs existants (Haas et al., 2001). En prenant ces considérations en compte, la planification de la main-d'œuvre dans une entreprise a pour

objectif d'optimiser la capacité collective des employés afin de répondre à tous les besoins de cette entreprise, et d'assurer un travail avec une certaine qualité de service auprès de la clientèle et la satisfaction des employés.

Les problèmes de planification de la main-d'œuvre avec tournées (Workforce with Scheduling and Routing) sont des problèmes d'ordonnancement où les employés doivent réaliser des opérations (ou services) à différentes localisations géographiques (Castillo-Salazar et al., 2016). Ces problèmes doivent prendre en considération le transport en termes économiques (une distance à minimiser) mais aussi en termes de qualité perçue par les clients (ou usagers) et les employés qui assurent les tournées. Ces problèmes représentent un défi important pour les entreprises impliquées, notamment dans les secteurs de production et/ou de maintenance (Technician Routing and Scheduling Problem – TRSP), portuaires et aéroportuaires (de Armas et al., 2015; Fink et al., 2019), et les services de soins à domicile (Home Health Care problem – HHC). Ces problèmes peuvent être vus comme des problèmes de tournées de véhicules (Vehicle Routing Problem – VRP) avec des contraintes spécifiques (Bachouch et al., 2011; Mathlouthi et al., 2018). Par ailleurs, de nombreux problèmes prennent en compte des contraintes de coopération ou de coordination entre les employés. Cet aspect n'est pas abordé dans cette partie du manuscrit et fait l'objet du chapitre 4.

Le problème traité dans ce chapitre concerne le Workforce Scheduling and Routing Problem (WSRP) introduit par (Castillo-Salazar et al., 2014). Le WSRP est un problème d'ordonnancement d'employés avec tournées de véhicules, sans contrainte de coordination, où les employés sont non-identiques et où des critères de qualité de services sont pris en compte (Figure 1).

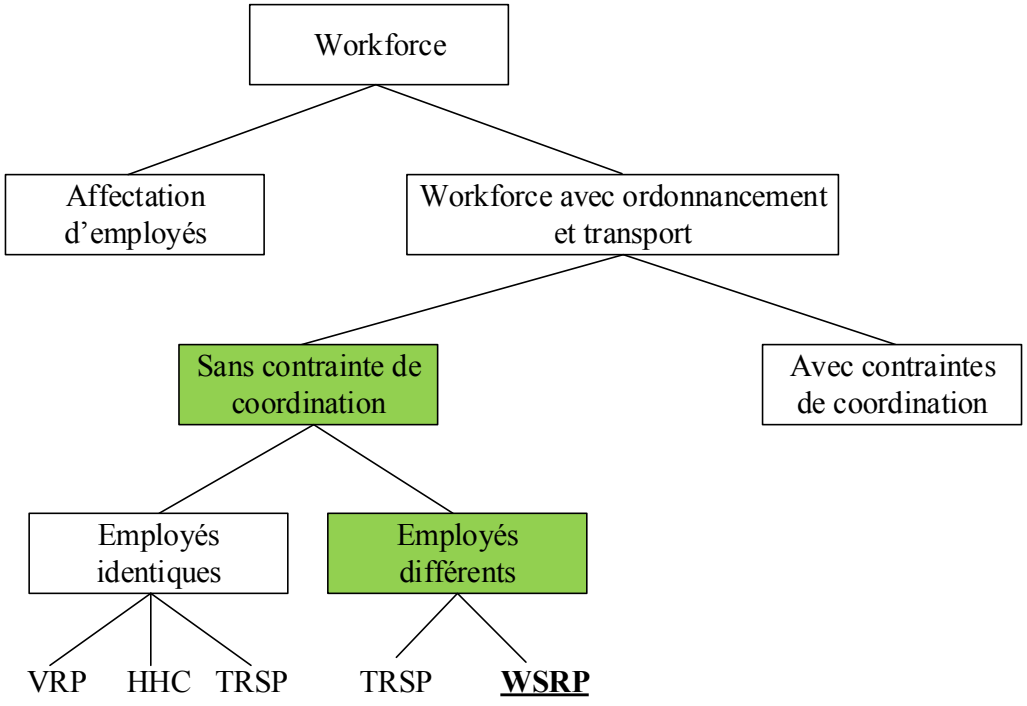


Figure 1 Classification des problèmes de planification et de tournées

Un des points importants du WSRP concerne la fonction objectif qui prend en compte un critère de coûts opérationnels, un critère de qualité de service pour les clients et un critère de qualité de service pour les employés. Ce dernier critère dépend des dates de début des visites et impose donc la recherche d'une solution qui ne soit pas forcément semi-active.

Ce chapitre introduit : 1) une formalisation du WSRP avec un modèle PLNE et un modèle de Programmation Par Contraintes (PPC) ; 2) une procédure calculant le coût minimal d'une tournée afin d'obtenir une solution qui n'est pas semi-active ; 3) une génération de colonnes pour résoudre le WSRP dont le problème esclave est une extension de l'Elementary Shortest Path Problem with Resource Constraints (ESPPRC) – où la solution optimale n'est pas semi-active – et est résolu par programmation dynamique.

Ce chapitre contient six parties : la première est une introduction du WSRP et comprend également un état de l'art. La seconde partie concerne la définition formelle du WSRP. La troisième partie a pour objet une fonction d'évaluation d'une tournée de WSRP permettant de maximiser la qualité de service de l'employé (cette dernière est dépendante des dates de début des visites). La quatrième partie présente deux modèles : l'un en PLNE et l'autre en PPC. La cinquième partie est une présentation d'une génération de colonnes avec notamment la formulation en PLNE du sous-problème et une nouvelle procédure de programmation dynamique pour résoudre le sous-problème. La sixième – et dernière – partie est une évaluation numérique des performances de la génération de colonnes.

1.1. Le Workforce Scheduling and Routing Problem

Le Workforce Scheduling and Routing Problem (WSRP) est décrit comme étant l'affectation d'employés (aussi appelés travailleurs) à des visites demandées par des clients (ou des patients dans le cas de soins à domicile) qui sont localisés sur différents sites géographiques, et comme la planification de ces visites. Un client peut nécessiter plusieurs visites. Le WSRP est un problème conjoint d'affectation de personnel à des visites et de tournées de véhicules pour chaque personnel. Les tournées définissent un ordonnancement des visites ce qui comprend la détermination des dates de passage. Le problème de tournées de véhicules et le problème d'ordonnancement consistent à générer une tournée pour chaque employé – en fonction des visites qui lui sont affectées – tout en respectant des contraintes sur les horaires de travail et les horaires des visites. Chaque visite est caractérisée par l'employé qui l'effectue, la date d'arrivée de l'employé, la date de début de la visite, la date de fin de visite et la date de départ de l'employé.

Les visites sont regroupées en région. Les employés ont une préférence sur les lieux géographiques (ou régions) où ils veulent (ou non) travailler. Une visite dispose d'une fenêtre de temps définissant la période de disponibilité du client durant laquelle la visite doit être effectuée. La fenêtre de temps (ou fenêtre de travail) d'un employé définit la période de disponibilité du travailleur durant laquelle il souhaite effectuer des visites. De plus, un critère de préférence est défini entre chaque client et chaque employé, ainsi qu'un coût entre chaque visite et chaque travailleur.

D'après (Castillo-Salazar et al., 2016), le Workforce Scheduling and Routing Problem (WSRP) fournit une modélisation efficace pour : le problème de planification des soins de santé à domicile (Home Health Care problem – HHC), le Technician and Task Scheduling Problem (TTSP), le Security Personnel Routing and Rostering Problem (SPRRP), le Manpower Allocation Problem (MAP) et le Vehicle Routing Problem with Time Windows (VRPTW). (Paraskevopoulos et al., 2017) fournissent un état de l'art portant sur les problèmes de tournées de véhicules et d'ordonnancement sous contrainte de ressources. Ils mettent en évidence l'intérêt majeur des problèmes d'ordonnancement de personnel pour de nombreuses applications, notamment le Home Health Care problem (Fikar and Hirsch, 2017) ; les problèmes d'installation, de maintenance et de réparations (Cordeau et al., 2010) ; les

problèmes de gestion forestière (Karlsson et al., 2004). D'après (Paraskevopoulos et al., 2017), le WSRP fait partie des problèmes classiques et typiques de tournées de véhicules et d'ordonnancement.

L'objectif du WSRP est, en premier lieu, de minimiser le nombre de visites non réalisées ; en second lieu, de maximiser la qualité de service des employés (ce qui inclut le respect de la fenêtre de temps de travail des employées et le respect des régions géographiques de travail) ; en troisième lieu de maximiser la satisfaction des préférences entre clients et employés ; et enfin de minimiser les coûts opérationnels qui incluent le transport et le coût de l'employé pour effectuer une visite.

1.2. État de l'art du WSRP

(Goel and Meisel, 2013) proposent un WSRP appliqué à un problème de maintenance de lignes électriques. Chaque opération de maintenance est constituée de plusieurs tâches qui sont situées en différents emplacements, et la ligne électrique doit être déconnectée du réseau. Leur objectif est de minimiser le temps d'arrêt des lignes et les déplacements des employés. Ils utilisent un Large Neighborhood Search (LNS) et leurs propres instances (GM).

(Castillo-Salazar et al., 2014) font partie des premiers à présenter le problème du WSRP, tel qu'il est étudié dans ce chapitre, avec une formalisation linéaire. Ils introduisent notamment les instances du WSRP qui sont utilisées dans la littérature. Leur objectif est d'obtenir une solution (mais pas optimale) de leurs instances avec un solveur linéaire.

(Laesanklang et al., 2015b) proposent une résolution du WSRP, en décomposant le problème et où chaque sous-problème représente une zone géographique spécifique, nommée Geographical Decomposition with Conflict Avoidance (GDCA). Les sous-problèmes sont résolus itérativement et sont dépendants les uns des autres. Un sous-problème est résolu par PLNE, et possède comme paramètre une liste d'employés potentiellement utilisables dans cette région. Cette liste est mise à jour à la fin de chaque résolution. Un travailleur appartient à cette liste s'il ne réalise pas déjà une visite dans une autre région (il ne doit pas faire partie de la solution d'un sous-problème déjà résolu). La qualité de la solution dépend alors de l'ordre dans lequel les sous-problèmes sont résolus, et les auteurs présentent donc plusieurs stratégies d'ordre de résolution des sous-problèmes. Ils développent également un algorithme génétique. Cet article est la version étendue de (Laesanklang et al., 2015a).

(Laesanklang et al., 2016a) proposent une heuristique de décomposition similaire à celle de (Laesanklang et al., 2015b), appelée Repeated Decomposition with Conflict Repair (RDCR) pour le WSRP. Cette heuristique consiste à appliquer de façon répétée une phase comprenant la décomposition du problème en sous-problèmes et leur résolution, suivie d'une phase dédiée à la réparation des conflits. Dans cette méthode de décomposition, les sous-problèmes sont indépendants les uns des autres. Un conflit survient lorsqu'un employé est affecté à plusieurs tournées situées dans des régions différentes. Cet article est la version étendue de (Laesanklang et al., 2016b).

(Laesanklang and Landa-Silva, 2017) utilisent une autre méthode de décomposition pour résoudre le WSRP : le Geographical Decomposition with Conflict Repair (GDCR), qu'ils comparent avec le RDCR (Laesanklang et al., 2016a). Cette nouvelle méthode (GDCR) n'est pas dépendante de l'ordre de résolution des sous-problèmes. Dans le GDCR, un sous-problème est composé de toutes les visites appartenant à la même région et l'ensemble des employés est

complet (tous les employés sont disponibles). Une solution du problème est l'assemblage des solutions des sous-problèmes. Puisque tous les employés sont disponibles pour chaque sous-problème, un même employé peut appartenir à la solution de deux sous-problèmes distincts et donc deux tournées sont affectées au même employé dans la solution finale. Dans ce cas, la solution est réparée heuristiquement pour n'obtenir qu'une tournée par employé.

(Algethami and Landa-Silva, 2015) introduisent un Steady State Genetic Algortihm (SSGA), et leur étude se focalise sur des opérateurs génétiques dédiés au WSRP, pour réaliser le crossover et la mutation. Ils utilisent une représentation directe des solutions. Une représentation peut donc ne mener à aucune solution et dans ce cas, ils utilisent une heuristique pour essayer de corriger cette représentation.

(Algethami et al., 2016) proposent un algorithme génétique (GA), pour le WSRP, avec une représentation indirecte des solutions qui permet de parcourir uniquement les états qui correspondent à des solutions. Ils utilisent huit opérateurs de crossovers et cinq opérateurs de mutation qui sont sélectionnés avec des probabilités. (Algethami and Landa-Silva, 2017) étendent ces travaux avec un Diversity-Based Adaptive Genetic Algorithm (DBAGA) où la taille de la population varie en fonction d'une mesure basée sur la diversité de la population. Les mutations et les crossovers, utilisés à chaque itération, sont sélectionnés en fonction de cette mesure.

(Algethami et al., 2017) comparent le Steady State Genetic Algorithm (SSGA) avec le Genetic Algorithme (GA) de (Algethami et al., 2016). Ils utilisent deux opérateurs pour réparer les solutions : le time conflict repair, et le worker suitability repair. (Algethami et al., 2018) est une version étendue de cet article.

(Pinheiro et al., 2016) présentent un Variable Neighbourhood Search (VNS) pour le WSRP faisant appel à deux heuristiques dédiées au WSRP : une pour créer une liste de travailleurs prioritaires pour chaque visite afin de réduire l'espace de recherche ; et la seconde pour réduire les violations des fenêtres de temps des employés et de leurs régions de préférence.

Le Tableau 1 présente une synthèse des différentes publications et des méthodes utilisées pour résoudre le WSRP. Le Tableau 1 met en évidence que la majorité des méthodes et approches proposées dans la littérature sont des heuristiques, et qu'elles sont testées sur les instances WSRP de (Castillo-Salazar et al., 2014).

Tableau 1 Résumé des publications concernant le WSRP

Articles	Méthodes	Instances
(Goel and Meisel, 2013)	LNS	GM
(Castillo-Salazar et al., 2014)	PLNE	WSRP
(Algethami and Landa-Silva, 2015)	Steady State Genetic Algorithm	WSRP
(Algethami et al., 2016)	Genetic Algorithm	WSRP
(Algethami and Landa-Silva, 2017)	Diversity-Based Adaptive Genetic Algorithm	WSRP
(Algethami et al., 2017)	Generational Genetic Algorithm	WSRP
(Algethami et al., 2018)	Adaptive Multiple Crossover Genetic Algorithm	WSRP
(Laesanklang et al., 2015b)	Geographical Decomposition with Conflict Avoidance	WSRP
(Laesanklang et al., 2015a)	Geographical Decomposition with Conflict Avoidance	WSRP
(Laesanklang et al., 2016a)	Repeated Decomposition with Conflict Repair	WSRP
(Laesanklang et al., 2016b)	Repeated Decomposition with Conflict Repair	WSRP
(Laesanklang and Landa-Silva, 2017)	Geographical Decomposition with Conflict Repair	WSRP
(Pinheiro et al., 2016)	Variable Neighborhood Search	WSRP

(Castillo-Salazar et al., 2016) fournissent une définition du WSRP et une comparaison de ce problème avec cinq autres problèmes qui peuvent être considérés comme des cas particuliers du WSRP. Ils présentent le Home Health Care problem (HHC), le Technician and Task Scheduling Problem (TTSP), le Security Personnel Routing and Rostering Problem (SPRRP), le Manpower Allocation Problem (MAP) et le Vehicle Routing Problem with Time Windows (VRPTW). Le Tableau 2 présente un résumé des principales caractéristiques de ces six problèmes (WSRP, HHC, TTSP, SPRRP, MAP et VRPTW) d'après (Castillo-Salazar et al., 2016).

Les problèmes se différencient par leur fonction objectif : le WSRP possède une fonction agrégée minimisant simultanément la distance, le coût opérationnel et maximisant la qualité de service pour les employés et les clients, et le nombre de visites réalisées ; le VRPTW a pour objectif la minimisation de la distance ; le HHC se focalise sur la minimisation de la distance ou la maximisation des préférences ; le MAP (Manpower Allocation Problem) a pour objectif la maximisation du nombre de visites réalisées ; les fonctions objectifs du TTS (Technician and Task Scheduling Problem) et du SSRP (Security Personnel Routing and Rostering Problem) ne sont pas données. Le WSRP est le seul problème à prendre en compte des préférences à la fois chez les clients et chez les employés ainsi qu'un coût pour réaliser la visite dépendant de l'employé qui la réalise. Pour le WSRP, le HHC et le SPRRP les employés peuvent soit débuter leur tournée depuis leur propre domicile, soit les commencer à partir d'un dépôt, il en est de même pour le retour : ils peuvent soit directement rentrer à leur domicile soit ils doivent retourner à un dépôt d'arrivée. Les problèmes de VRPTW, de TTSP et de MAP imposent que les tournées débutent à un dépôt. Pour plus d'informations sur les problèmes de type HHC, (Fikar and Hirsch, 2017) proposent un état de l'art récent de ces derniers. De nombreux problèmes de HHC et de VRPTW prennent en compte des contraintes de coordination des visites, lesquelles sont l'objet du chapitre 4 et ne sont pas considérées dans ce chapitre.

Tableau 2 Caractéristiques du WSRP d'après (Castillo-Salazar et al., 2016)

	WSRP	**VRPTW**	**HHC**	**TTSP**	**SPRRP**	**MAP**
Objectif	Distance, coût opérationnel, préférence, nb visites	Distance	Distance ou préférence			Nb visites
Visites non réalisées	Autorisé	Interdit	Interdit	Interdit	Interdit	Autorisé
Caractéristiques des clients						
Nombre de visites chez le même client	≥ 1	1				
Fenêtre de temps	X	X	X		X	
Préférences	X		X			
Durées de service	X		X	X	X	
Coût	X					
Caractéristiques des employés						
Positions départ - arrivée	Dépôt / domicile	Dépôt	Dépôt / domicile	Dépôt	Dépôt / domicile	Dépôt
Fenêtre de temps	X		X			
Compétences	X		X	X	X	
Préférences	X					
Temps de transport	X	X	X	X	X	X
Régions	X			X	X	
Pauses						X
Temps de travail maximal			X			X
Travail en équipe				X		
Shift			X			

La principale différence entre le WSRP et les autres problèmes réside dans la possibilité de ne pas respecter les fenêtres de temps des employés afin de réaliser plus de visites. Ces violations entraînent toutefois une pénalité dans la fonction objectif. Ces pénalités modifient profondément l'espace de recherche des solutions, et en ayant pour objectif la minimisation du nombre de ces pénalités, la solution recherchée n'est pas nécessairement semi-active. En effet, le nombre de pénalités dépend des dates de début des visites et retarder une ou plusieurs dates de début peut permettre de respecter les fenêtres de travail des employés, et donc d'obtenir une solution de coût plus faible. Ainsi l'ensemble des solutions semi-actives ne contient pas forcément la solution optimale.

2. Définition du Workforce Scheduling and Routing Problem

2.1. Modélisation sous forme de graphe

Soit un graphe $G = (\mathcal{V}, \mathcal{E})$, où $\mathcal{V}$ est l'ensemble des nœuds et $\mathcal{E}$ l'ensemble des arrêtes. Soit $W = \{w_1, w_i, \dots w_{|W|}\}$ l'ensemble des employés. $\mathcal{V} = V \cup I \cup L$ où V est l'ensemble des visites à réaliser, $I = \{I_1, \dots, I_{w_{|W|}}\}$ est l'ensemble des dépôts de départ des employés et $L = \{L_1, \dots, L_{w_{|W|}}\}$ est l'ensemble des dépôts d'arrivée des employés.

(Misir et al., 2010) donnent une définition précise des principales caractéristiques des employés et des visites dans le cadre du Home Health Care Problem. Ces définitions sont tout à fait en adéquation avec le problème du WSRP, et certaines sont utilisées pour définir les visites et les employés.

Une visite i est caractérisée par :

- Une fenêtre de temps $[E_i\ ;\ L_i]$ durant laquelle la visite doit être effectuée.
- Une durée de service Pt_i, pour le WSRP cette durée est déterministe, mais elle pourrait dépendre de l'employé réalisant la visite ou stochastique.
- Une localisation (ou zone géographique).
- Un coût PC_i^w dépendant de l'employé w effectuant la visite.
- Un critère de qualité de service ρ_i^w qui est fonction de w.
- Un nombre R_i d'employés nécessaires pour effectuer la visite. Si $R_i > 1$ alors la visite est dupliquée en R_i visites $i'^1, \dots, i'^{R_i}$ identiques. Si plusieurs employés sont requis, il est généralement important voire, essentiel qu'ils se coordonnent. Cet aspect de synchronisation et de coordination est un défi majeur des problèmes de WSRP, VRPTW et HHC, et est l'objet intégral de l'étude du chapitre 4. Ces contraintes ne sont pas présentes pour le WSRP introduit dans ce chapitre.
- Le temps de transport $T_{i,j}$ de la visite i à la visite j est supposé indépendant des employés.

Un employé w est caractérisé par :

- Une fenêtre de temps $[TW_{inf}^w\ ;\ TW_{sup}^w]$ (aussi appelée fenêtre de travail). La définition de la fenêtre de temps de travail peut varier en différents scénarii : le temps de déplacement peut être considéré comme du temps de travail ; un employé peut être contraint de violer sa fenêtre de temps de travail si cela est nécessaire pour exécuter une visite. Pour le WSRP, le temps de déplacement n'est pas considéré comme du temps de travail et un travailleur est autorisé à travailler au-delà de sa fenêtre horaire, mais ces dépassements d'horaires sont pénalisés dans la fonction objectif.
- Un contrat de travail $CoTr_i^w$ autorisant ($CoTr_i^w = 1$) – ou non ($CoTr_i^w = 0$) – w à réaliser la visite i. Le contrat peut être vu comme équivalent à la définition des compétences proposées par (Cordeau et al., 2010), où chaque visite réclame une ou plusieurs compétences particulières dont une au moins doit être possédée par l'employé réalisant cette visite.
- Un dépôt de départ I_w.
- Un dépôt d'arrivée L_w.
- Une liste de compétence.
- Une valeur binaire PA_i^w qui vaut 0 si la visite i n'est pas dans une région souhaitée par l'employé, ou qui vaut 1 si elle ne l'est pas.

Les dépôts de départ I_w et les dépôts d'arrivée peuvent être, pour un employé donné, les mêmes ($I_w = L_w$) ou différents ($I_w \neq L_w$). Similairement, pour l'ensemble des employés, les dépôts peuvent être tous identiques ($\forall w, k \in W, I_w = I_k$ (Eveborn et al., 2006)) ou tous différents ($\forall w, k \in W, I_w \neq I_k$ (Castillo-Salazar et al., 2016)). Cette représentation des différents dépôts permet de prendre en compte tous les scénarii possibles suivant les politiques de chaque entreprise les employés partent directement de chez eux ou ils débutent leur tournée à partir d'un dépôt central, puis ils retournent directement chez eux ou doivent repasser par le dépôt central avant de rentrer chez eux.

Le critère de service ρ_i^w, établi par (Castillo-Salazar et al., 2016), englobe trois termes et est défini par $\rho_i^w = (PG_i^w + PW_i^w + PS_i^w)$ tel que $\rho_i^w \in [0\ ;\ 3]$, $\rho_i^w \in \mathbb{R}$, avec :

- $PG_i^w \in [0\,;\,1]$, la préférence de l'employé w à travailler dans la zone géographique dans laquelle se trouve la visite i. $PG_i^w = 0$ signifie que l'employé w ne souhaite pas effectuer la visite à cause de sa zone géographique. Un PG_i^w proche de 1 signifie que l'employé est d'accord pour réaliser la visite.
- $PS_i^w \in [0\,;\,1]$, la préférence du client demandant la visite i par rapport à l'employé w. $PS_i^w = 0$ signifie que l'employé w n'est pas du tout apprécié pour réaliser la visite. Une valeur proche de 1 signifie, au contraire, que l'employé est très vivement approuvé.
- $PW_i^w \in [0\,;\,1]$, la préférence du client requêtant la visite i par rapport aux qualifications de l'employé w. Une valeur proche de 0 exprime un désaccord, tandis qu'une valeur proche de 1 traduit une satisfaction.

La fonction objectif comprend quatre critères hiérarchisés (Castillo-Salazar et al., 2016; Garaix et al., 2018b). Le premier critère regroupe les coûts opérationnels, c'est-à-dire, la somme des temps de transport $T_{i,j}$ plus la somme des coûts PC_i^w. Il combine donc des coûts liés aux tournées et des coûts liés à la planification. Le second critère concerne la Qualité de Service (QoS) ρ_i^w des clients. Le troisième critère englobe les pénalités des employés violant leur fenêtre de temps et violant leur zone géographique (il correspond à la qualité de service des employés). Le quatrième critère est le nombre de visites non effectuées. La fonction objectif est la somme de ces quatre critères avec des pondérations différentes : le critère le plus important est le quatrième, ensuite le troisième, puis le second et enfin le premier. La fonction objectif f peut se résumer par :

$$f = \lambda_1(\text{coûts opérationnels}) + \lambda_2(\text{QoS des clients})$$

$$+\lambda_3(\text{QoS des employés}) + \lambda_4(\text{nombre de visites non effectuées})$$

Avec λ_1, λ_2, λ_3 et λ_4 les pondérations où $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$.

Le Tableau 3 présente les objectifs, les contraintes et les préférences du WSRP d'après la description du problème par (Castillo-Salazar et al., 2016) et (Garaix et al., 2018b). La première colonne du Tableau 3 concerne les quatre critères de la fonction objectif. La deuxième colonne concerne les contraintes qui doivent être vérifiées à tout moment : le respect des fenêtres de temps des visites et le respect des contrats entre employés et les clients. Un contrat permet de définir la liste des clients qu'un employé peut visiter. La troisième colonne a pour objet les préférences qui sont traduites par des pénalités dans la fonction objectif (second et troisième critère de celle-ci) et qui définissent la qualité de service des employés et des clients.

Tableau 3 Objectifs, contraintes et préférences dans le WSRP

Critères	Contraintes	Préférences
Minimisation des temps de transport et des coûts opérationnels : $T_{i,j} + PC_i^w$	Contraintes légales liées aux contrats de travail : $CoTr_i^w$	Qualité de service des clients : ρ_i^w
Maximisation de la qualité de service : ρ_i^w	Fenêtre de temps des visites : $[E_i\ ;\ L_i]$	Régions préférées des employés : PA_i^w
Minimisation du nombre de visites réalisées en dehors de la fenêtre de temps de l'employé et du nombre d'employés travaillant dans une zone géographique non souhaitée ($PA_i^w = 0$)		Horaires de travail des employés : $[TW_{inf}^w\ ;\ TW_{sup}^w]$
Minimisation du nombre de visites non effectuées		

2.2. Représentation d'une solution

Une solution du WSRP doit bien évidemment respecter les contraintes du Tableau 3 et est définie par :

- L'affectation d'un employé à chaque visite.
- L'ordre de passage, pour chaque employé, sur les visites qui lui sont affectées.
- La date A_i d'arrivée de l'employé sur la visite i.
- La date st_i de début de service de l'employé sur la visite i.
- La date C_i de fin de service de l'employé sur la visite i.
- La date D_i de départ de l'employé de la visite i.

Le Tableau 4 propose une instance simplifiée du WSRP où les matrices PC_i^w, ρ_i^w, PA_i^w et $CoTr_i^w$ sont simplifiées en étant unitaires (ce qui n'impacte donc pas la solution). Cette instance comporte six visites ($V1, \dots, V6$) et deux employés ($W1$ et $W2$). Les fenêtres de temps des visites et celles des employés sont indiquées dans la seconde colonne du Tableau 4. La troisième colonne présente, le processing time des visites, le dépôt de départ et d'arrivée de chacun des employés.

Tableau 4 Exemple de WSRP

Visite	Fenêtre de temps	Processing time
$V1$	$[10\ ;30]$	$Pt = 5$
$V2$	$[5\ ;33]$	$Pt = 8$
$V3$	$[68\ ;94]$	$Pt = 11$
$V4$	$[49\ ;90]$	$Pt = 25$
$V5$	$[29\ ;54]$	$Pt = 11$
$V6$	$[54\ ;95]$	$Pt = 20$
Employé	**Fenêtre de temps**	**Dépôts (départ – arrivée)**
$W1$	$[35\ ;100]$	$I1$ et $L1$
$W2$	$[9\ ;85]$	$I2$ et $L2$

La Figure 2 est un exemple de solution de l'instance du Tableau 4, modélisée par un graphe. Les dépôts des employés et les visites sont des nœuds du graphe et sont respectivement représentés par des carrés et des ronds. Sont rappelés, pour chaque visite, sa fenêtre de temps (TW), son processing time Pt, la date A d'arrivée de l'employé, la date st de début de service, la date C de fin de service et la date D de départ de l'employé. Par exemple, pour la visite $V2$, l'employé arrive à la date $A = 12$, il débute son service à la date $st = 12$, il finit à la date $C = 20$, puis attend avant de partir à la date $D = 29$. Pour la visite $V4$, l'employé arrive à la date $A = 42$, mais attend la date $st = 49$ pour débuter le service.

Chaque tournée est un chemin constitué d'arcs, débutant d'un dépôt de départ et finissant à un dépôt d'arrivée. La tournée de l'employé $W1$ (en pointillé bleu sur la Figure 2) débute au dépôt $I1$ puis, $W1$ effectue la visite $V1$, ensuite la visite $V5$, puis $V6$ et enfin il retourne à son dépôt d'arrivée $L1$. La tournée de l'employé $W2$ est $I2 - V2 - V4 - V3 - L2$ et est dessinée en traits pleins marron.

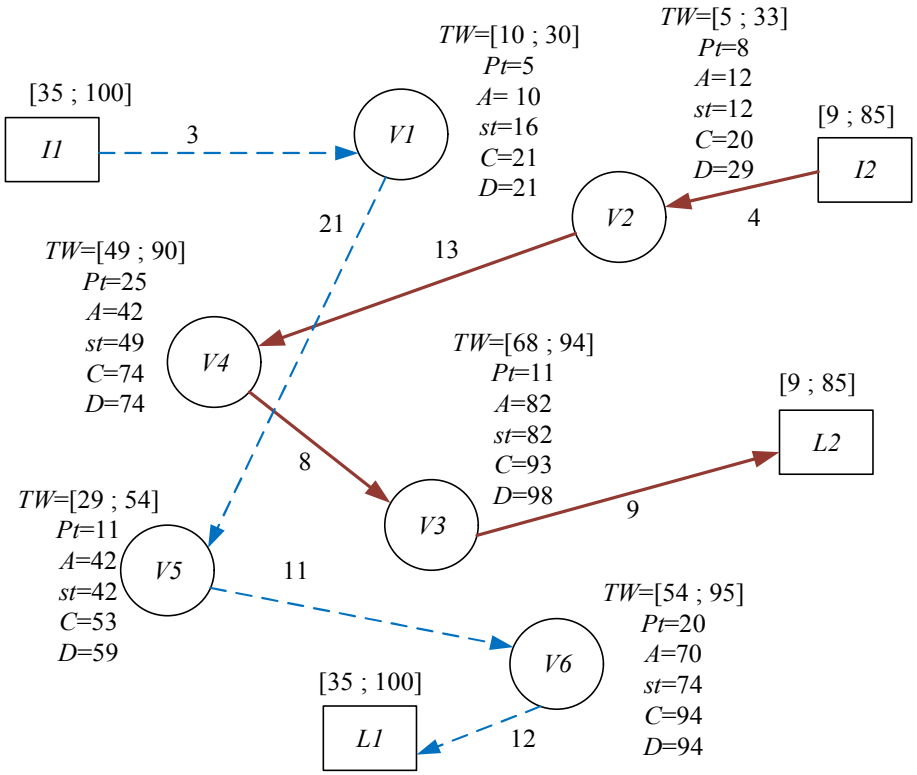


Figure 2 Exemple de solution du WSRP

La Figure 3 présente les diagrammes de Gantt de la solution : le premier diagramme est « axé » sur les employés (partie haute de la figure) tandis que le second est « axé » sur les visites (partie basse de la figure). Le premier diagramme représente la tournée de chaque employé sur un axe temporel. Le second diagramme représente la réalisation de chaque visite par un employé, sur un axe temporel. Les rectangles les plus larges représentent les fenêtres de temps des employés (sur le premier diagramme) et des visites (sur le second diagramme). Pour chaque employé, la fenêtre de temps est donnée. L'employé $W1$ commence sa tournée par la visite $V1$ en dehors de sa fenêtre de temps. Une pénalité est donc comptabilisée (cet aspect est repris en détail dans la section 4.1.2). L'employé $W2$ commence ses visites dans sa fenêtre de temps, mais la visite $V3$ finit à la date $C = 93$ qui est une date supérieure à la fenêtre de temps de $W2$ ($[9 ; 85]$), une pénalité est également comptée.

La seconde partie du diagramme de Gantt met en évidence les dates de début des visites dans leur fenêtre de temps, puisque c'est une contrainte du WSRP. Une flèche entre deux visites représente le déplacement d'un employé entre ces deux visites. Ce diagramme montre les temps d'attente des employés : par exemple après la visite $V5$ (qui termine à la date $C = 53$), l'employé ne part pas immédiatement, il part à la date $D = 59$, il attend donc 6 unités de temps. De manière similaire, les employés ne débutent pas forcément les visites immédiatement après leur arrivée : par exemple l'employé $W1$ arrive à la visite $V6$ à la date $A = 70$, mais débute seulement à la date $st = 74$. L'employé $W2$ arrive sur la visite $V4$ à la date $A = 49$, mais doit attendre l'ouverture de la fenêtre de temps de la visite avant de pouvoir commencer le service.

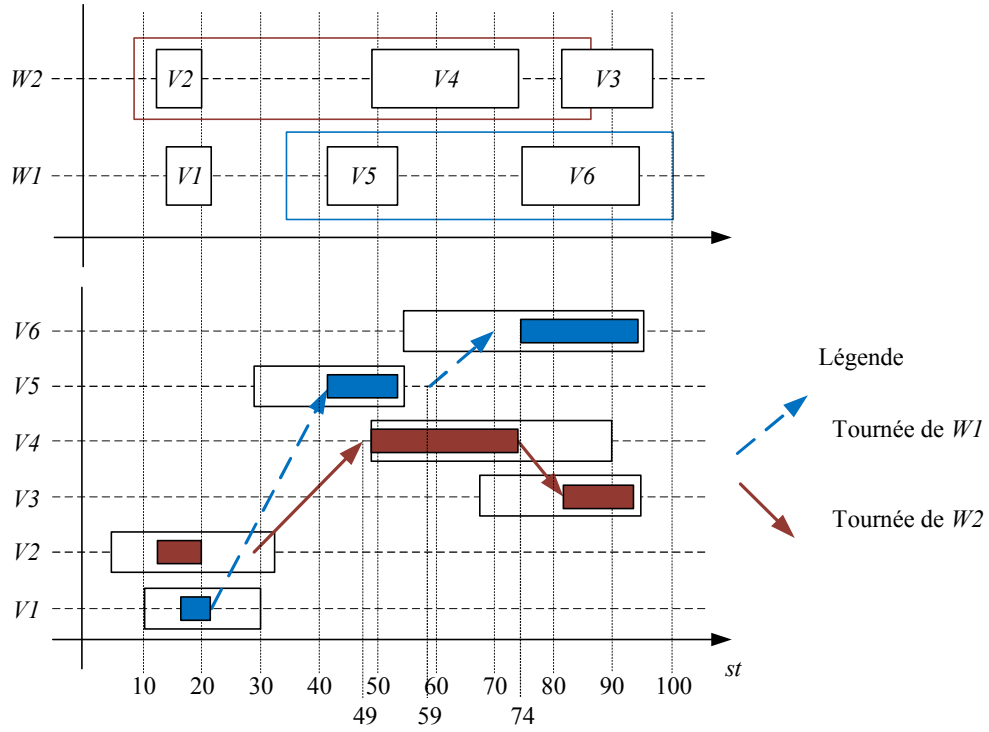


Figure 3 Diagrammes de Gantt de la solution

La solution présentée (Figure 3) n'est donc ni au plus tôt, ni au plus tard, mais une solution intermédiaire. Puisque la définition du WSRP autorise la violation des fenêtres de temps des employés (sachant que ces violations sont pénalisées dans la fonction objectif), il est important que la solution ne soit une solution ni au plus tôt, ni au plus tard. Cette pénalité empêche de réduire l'ensemble des solutions à l'ensemble des solutions semi-actives, car la solution optimale n'est pas obligatoirement semi-active. Cette propriété du WSRP est très différente de celle des problèmes d'ordonnancement classiques, notamment le Job-shop Scheduling Problem pour le critère du makespan où l'espace des solutions semi-actives contient au moins une solution optimale (voir chapitre 1 et chapitre 2).

3. Évaluation d'une tournée de coût minimal

Une solution du WSRP est un ensemble de tournées indépendantes les unes des autres, qui doivent être évaluées par une fonction d'évaluation ne donnant pas une solution semi-active. La difficulté de la fonction d'évaluation réside dans le choix de la date de début des

visites, car celles-ci doivent maximiser la qualité de service (QoS) des employés et une solution semi-active ne possède pas un QoS maximal.

Pour une tournée donnée (la séquence des visites est connue), un algorithme de plus long chemin classique, de type Bellman-Ford, ne permet pas de connaître les dates de début des visites maximisant la QoS car cet algorithme retourne une solution semi-active. Cette section introduit une fonction d'évaluation d'une tournée pour obtenir les dates de début des visites afin de maximiser la qualité de service. Cette fonction d'évaluation permet également de se familiariser avec le problème et est étendue pour être incluse dans la génération de colonnes (voir la section 5).

3.1. Données du WSRP

Les données sont les mêmes que celles introduites pour le WSRP à la section 2.1. Elles sont néanmoins rappelées de manière concice dans cette section afin d'aider le lecteur. Soient i une visite et w un employé.

- V : l'ensemble des visites, numérotées de 1 à $|V|$.
- W : l'ensemble des employés, numérotés de 1 à $|W|$.
- I : l'ensemble des dépôts de départ, numérotés de $I^d = |V| + 1$ à $I^f = |V| + 2|W|$.
- I_w : le dépôt de départ de l'employé w.
- L : l'ensemble des dépôts d'arrivée, numérotés de $L^d = |V| + |W| + 1$ à $L^f = |V| + 2|W|$.
- L_w : le dépôt d'arrivée de l'employé w.
- Nts : l'ensemble de tous les nœuds du graphe, $Nts = V \cup I \cup L$.
- Pt_i : durée de service de la visite i (ou processing time).
- $[E_i\ ;\ L_i]$: la fenêtre de temps de la visite i.
- PC_i^w : le coût de réalisation de la visite i par l'employé w.
- ρ_i^w : la qualité de service de la visite i pour l'employé w.
- $CoTr_i^w$: le contrat de travail autorisant ($CoTr_i^w = 1$) – ou non ($CoTr_i^w = 0$) – l'employé w à réaliser la visite i.
- $T_{i,j}$: le temps de transport de la visite i à la visite j.
- $[TW_{inf}^w\ ;\ TW_{sup}^w]$: la fenêtre de temps de l'employé w.
- PA_i^w: donnée binaire valant 1 si w apprécie la région où est située i, et valant 0 sinon.

3.2. Une solution non semi-active

Une solution du WSRP n'est pas une solution semi-active où les dates de début des visites sont au plus tôt, ni une solution où les dates de début sont au plus tard, mais il s'agit d'un compromis entre les deux, car les employés peuvent violer leur fenêtre de travail. Une solution peut donc accepter que des employés exécutent des visites en dehors de leur fenêtre de travail (la visite commence avant la fenêtre de travail de l'employé ou la visite finie après la fenêtre de travail de l'employé). Toutefois, une pénalité est comptabilisée pour chaque violation.

Pour le WSRP, les tournées sont indépendantes les unes des autres, et une solution optimale est composée des tournées de coût optimal, indépendantes les unes des autres. Pour obtenir la solution optimale au WSRP, la fonction d'évaluation évalue les tournées itérativement en minimisant le coût (et maximisant la qualité de service des employés) de chaque tournée, sans tenir compte des autres tournées. Il suffit donc d'avoir une fonction

d'évaluation qui prend une tournée en entrée et trouve les dates de début des visites de cette tournée afin de fournir le coût minimal.

Soit une tournée fixée Γ, pour l'employé w, débutant au dépôt de départ I_w, finissant au dépôt d'arrivée L_w et qui est constituée d'une succession de visites i. Le successeur de la visite i dans la tournée est noté $\Gamma(i+1)$. Le coût d'une tournée (Laesanklang et al., 2015b; Garaix et al., 2018b) est donné par CT_Γ :

$$CT_\Gamma = \lambda_1 \sum_{i \in \Gamma \setminus L_w} \left(T_{i,\Gamma(i+1)} + p_i^w\right) + \lambda_2 \sum_{i \in \Gamma \setminus \{I_w; L_w\}} (3 - \rho_i^w) + \lambda_3 \sum_{i \in \Gamma \setminus \{I_w; L_w\}} (\psi_i^w + \theta_i^w)$$

Ce coût peut se réécrire :

$$CT_\Gamma = \gamma_\Gamma + \Theta_\Gamma(st_i)$$

Avec

$$\gamma_\Gamma = \lambda_1 \sum_{i \in \Gamma \setminus L_w} \left(T_{i,\Gamma(i+1)} + p_i^w\right) + \lambda_2 \sum_{i \in \Gamma \setminus \{I_w; L_w\}} (3 - \rho_i^w) + \lambda_3 \sum_{i \in \Gamma \setminus \{I_w; L_w\}} (\psi_i^w)$$

Et

$$\Theta_\Gamma(st_i) = \lambda_3 \sum_{i \in \Gamma \setminus \{I_w; L_w\}} (\theta_i^w)$$

Il est important de noter que γ_Γ est indépendant des dates de début des visites, mais que Θ_Γ est dépendant des dates de début st_i ($i \in \Gamma$), car θ_i^w est une pénalité qui dépend également des dates de début.

Les pénalités concernant les violations des fenêtres de travail des employés sont une particularité du WSRP par rapport aux problèmes classiques de tournées de véhicules. La Figure 4 illustre le calcul des pénalités concernant la violation de la fenêtre de travail de l'employé (donc les variables θ_i^w). La fenêtre de travail de l'employé $w \in W$ est représentée par le rectangle orange. Les visites i, j, m et n sont associées à une pénalité. En effet, soit les visites débutent avant la fenêtre de travail de w (pour les visites i et j) ; soit les visites se terminent après la fenêtre de travail de w (visites m et n). Par conséquent, $\theta_i^w = \theta_j^w = \theta_m^w = \theta_n^w = 1$ et $\theta_k^w = \theta_l^w = 0$, car les deux dernières visites citées débutent et finissent durant la fenêtre de temps de w. Il est important de noter qu'il y a donc autant de pénalités que de visites (et non pas une pénalité par tournée, comme il peut être courant dans certains problèmes de tournées de véhicules).

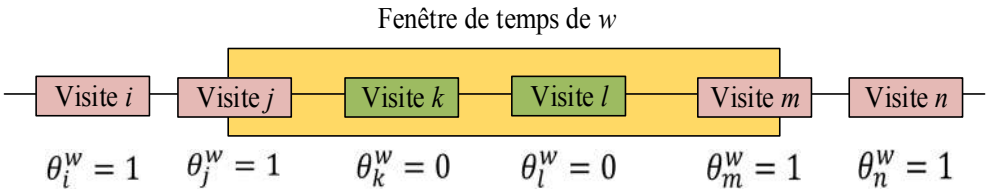


Figure 4 Pénalités concernant les fenêtres de temps

Le coût de la tournée est donc dépendant des dates des visites, et trouver les dates minimisant le coût est une difficulté. Lorsque l'employé arrive sur une visite avant sa fenêtre de temps (également dite fenêtre de travail), deux choix sont possibles :

- Soit l'employé commence la visite immédiatement, mais il risque de payer une pénalité dans la fonction objectif.
- Soit l'employé attend le début de sa fenêtre de travail pour ne pas payer de pénalité. Mais il se peut qu'il ne puisse plus réaliser la tournée à cause des fenêtres de temps des visites, ou qu'il paye plus de pénalité car plusieurs visites seront alors en dehors de sa fenêtre de temps.

Sur la Figure 5, la date d'arrivée de l'employé w sur la visite i est symbolisée par un rectangle rouge et est située avant la fenêtre de temps de l'employé. Celui-ci a deux possibilités : soit commencer immédiatement la visite (partie haute de la figure), soit attendre (partie basse de la figure) pour débuter la visite dans sa fenêtre de travail (aussi appelée fenêtre de temps). Dans le premier cas, la visite i est réalisée en-dehors de la fenêtre de temps de w, et une pénalité est comptée. Dans le second cas, il n'y a aucune pénalité. Il est évident que dans ce cas présent, l'employé w a intérêt à attendre l'ouverture de sa fenêtre de travail pour ne pas payer de pénalité.

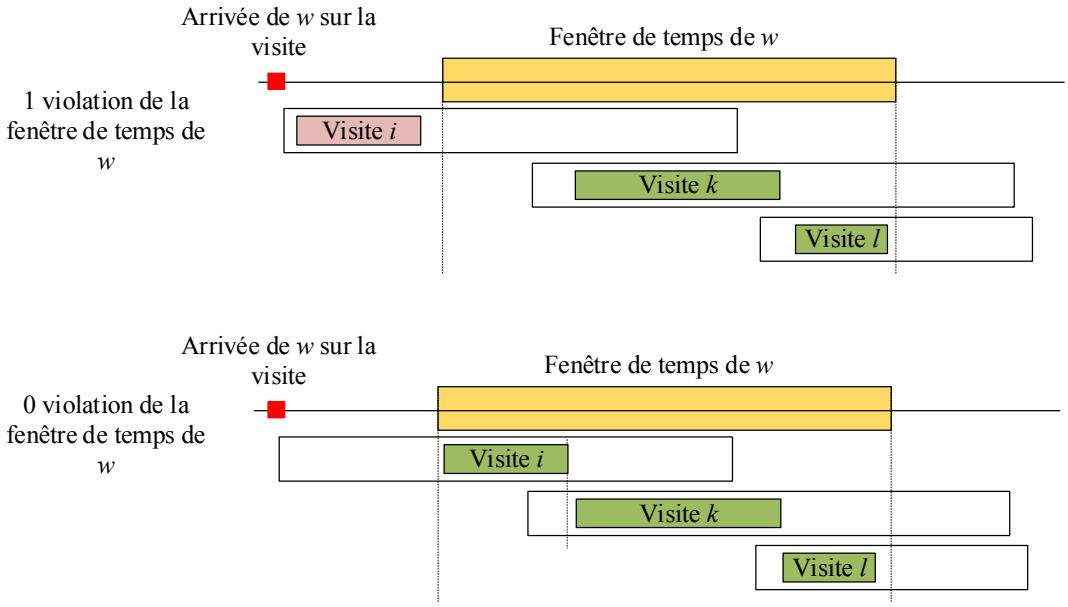


Figure 5 Début de la visite « trop tôt » entraînant une pénalité

Sur la Figure 6, la date d'arrivée de l'employé w sur la visite i est toujours représentée par un rectangle rouge. La partie haute de la figure illustre le cas où l'employé commence la visite avant sa fenêtre de travail. Une pénalité est alors comptabilisée. Si l'employé fait le choix d'attendre, alors les deux autres visites k et l débutent plus tard et finissent après la fenêtre de travail de w, impliquant deux pénalités. Le cas haut de la Figure 6 est le plus intéressant des

deux en termes de pénalité. Contrairement au cas précédent (Figure 5), ici, il vaut mieux que l'employé commence la visite dès son arrivée.

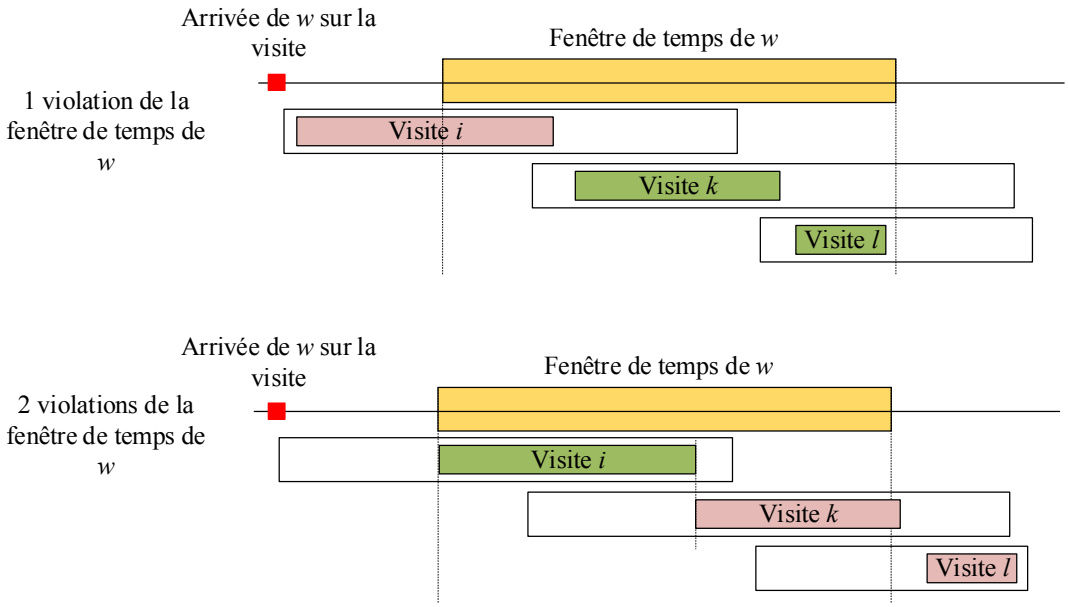


Figure 6 Retardement de la visite i impliquant un coût plus élevé

Sur la partie haute de la Figure 7, l'employé débute la visite *i* avant sa fenêtre de travail, et une pénalité est comptée. Sur la partie basse de la figure, l'employé attend avant de commencer la visite afin de ne pas payer de pénalité, mais la tournée devient irréalisable car la visite *k* ne tient plus dans sa fenêtre de temps. Or il est interdit pour une visite d'être en dehors de sa fenêtre de temps (contrainte inviolable du WSRP).

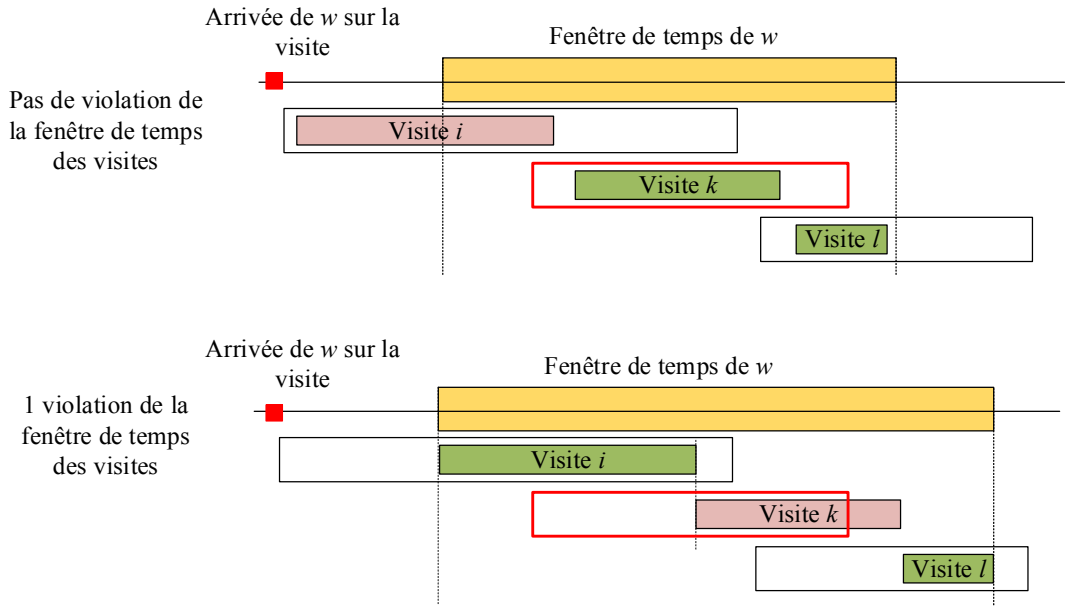


Figure 7 Retardement de i provoquant une violation de la fenêtre de temps de k

Ces trois illustrations montrent la difficulté de savoir si un employé doit attendre ou commencer la visite tout de suite. En effet, lors d'un algorithme de plus long chemin, lorsque

l'employé w arrive sur la visite, il est impossible de prédire quelle est la date de début permettant d'obtenir le coût minimal sans évaluer le reste de la tournée.

3.3. Date de début d'une visite

Bien que la date de début de la visite st_i^w soit sujette à discussion, parmi toutes les valeurs possibles, deux valeurs sont importantes et doivent être prises en considération : la valeur de la date quand l'employé commence la visite au plus tôt (immédiatement quand il arrive, ou quand la fenêtre de temps de la visite s'ouvre) ; et la valeur de la date quand l'employé commence la visite immédiatement lorsque sa fenêtre de travail s'ouvre. Les autres cas ne nécessitent pas d'être pris en compte comme le montre les explications par la suite.

Une solution (partielle) sur une visite est dite dominante par rapport à une autre solution (partielle) sur la même visite si les deux solutions ont le même coût, et que la première solution a une date de début plus petite.

Soient i et j deux visites avec, pour chacune et sans perte de généralité, une fenêtre de temps très large (de manière à ce que les fenêtres de temps des visites ne soient pas contraignantes pour cet exemple), et soit un employé w avec une fenêtre de travail $[TW_{inf}^w ; TW_{sup}^w]$. Soient st_j^w la date de début de la visite j par l'employé w ; a_j^w la date d'arrivée de l'employé w sur la visite j ; d_j^w la date de départ de l'employé w de la visite j ; Pt_i la durée de la visite i ; et $T_{j,i}$ le temps de trajet de la visite j à la visite i.

Dans les cas A, A', B et B' (illustrés par la Figure 8 et la Figure 9), l'employé w arrive sur la visite i avant le début de sa fenêtre de temps de travail (TW_{inf}^w). Il peut alors soit débuter la visite i immédiatement (il est supposé, sans perte de généralité, que la fenêtre de temps de la visite i n'est pas contraignante) ; soit attendre et commencer la visite plus tard. Le premier cas est représenté par le cas A, et les deux autres cas sont représentés par les cas A', B et B'.

Pour les cas A et A' (Figure 8) l'employé arrive sur la visite avant sa fenêtre de temps ($a_i^w < TW_{inf}^w$) : soit il débute sa visite immédiatement (cas A), soit il attend avant de commencer (cas A'). Dans les deux cas, $st_i^w < TW_{inf}^w$ et, quelle que soit la date de début de la visite, il y a une pénalité. Les deux cas ont le même coût puisque ce dernier ne prend en compte ni les temps d'attente ni leurs positions. Il est donc plus intéressant de commencer la visite au plus tôt ($st_i^w = a_i^w$) : au mieux l'employé peut faire plus de visites dans sa journée, au pire, il finit sa journée le plus tôt possible. Le cas A' est donc dominé par le cas A, et il n'y a alors pas de nécessité de le prendre en compte.

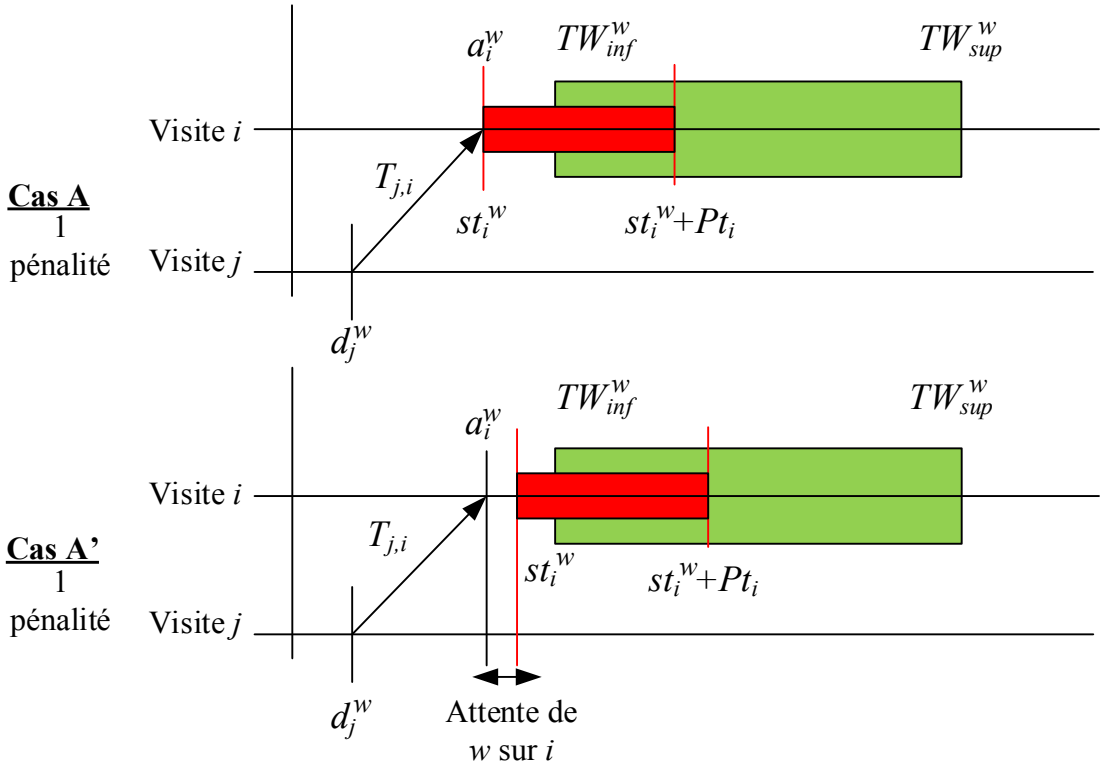


Figure 8 L'employé arrive avant l'ouverture de sa fenêtre de travail

Dans les cas B et B' (Figure 9), l'employé w arrive avant sa fenêtre de temps sur la visite et débute celle-ci après le début de sa fenêtre de travail ($st_i^w \geq TW_{inf}^w$), donc il n'y a pas de pénalité dans la fonction objectif. La différence entre ces deux cas et les deux précédents réside dans la durée du temps d'attente. L'employé attend sur le nœud de la visite i : $st_i^w \neq a_i^w$. Les cas B et B' ont un coût inférieur à la solution du cas A (du fait de l'absence de pénalité), mais dans les cas B et B' l'employé débute la visite plus tard. Les solutions induites par les cas B et B' ne sont pas comparables avec la solution induite par le cas A.

Le cas B' (Figure 9) représente tous les cas où $st_i^w > TW_{inf}^w$ et $st_i^w + Pt_i \leq TW_{sup}^w$. Dans ces cas, les solutions n'ont pas de pénalité de violation de la fenêtre de travail de l'employé, mais ces solutions sont dominées par le cas B : les solutions de B ont le même coût que B', mais les solutions de type B' ont une date de début plus grande et sont donc moins intéressantes.

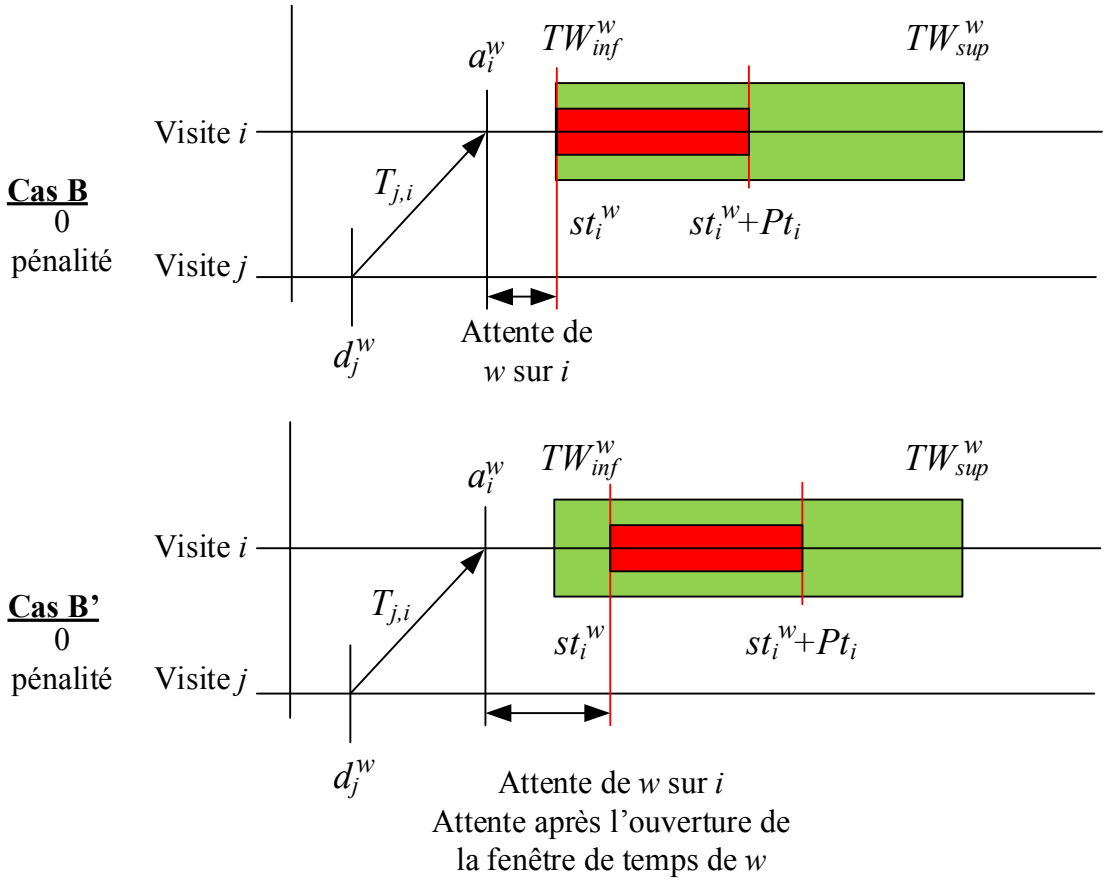


Figure 9 L'employé débute le travail durant sa fenêtre de temps

Le cas C (Figure 10) illustre un cas où l'employé attend avant de débuter la visite i ($a_i^w < st_i^w$), de telle sorte qu'il finit la visite après la fin de sa fenêtre de travail ($st_i^w + Pt_i \leq TW_{sup}^w$). Dans ce cas, une pénalité de violation est comptée, et cette solution est dominée par les cas A et B. En effet, dans le cas A, la solution a le même coût mais une date de début plus petite, et dans le cas B, la solution a un coût inférieur (sans pénalité) ainsi qu'une date de début plus petite.

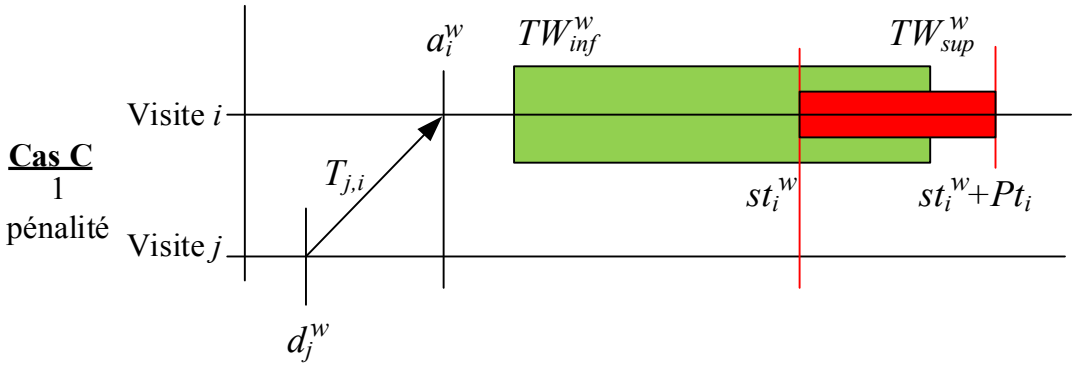


Figure 10 L'employé finit après sa fenêtre de travail

Dans le cas D (Figure 11), l'employé a fini la visite j au temps $st_j^w + Pt_j$ mais quitte celle-ci à la date $d_j^w > st_j^w + Pt_j$. Le cas F est équivalent au cas B, car le temps d'attente peut être sans distinction sur la visite i ou sur la visite j. Il n'est pas nécessaire de conserver ce cas.

En effet, il n'est pas intéressant de faire attendre l'employé sur le nœud de la visite précédente : il vaut mieux le faire partir au plus tôt, car l'attente n'a pas de coût, quelles que soient sa durée ou sa position.

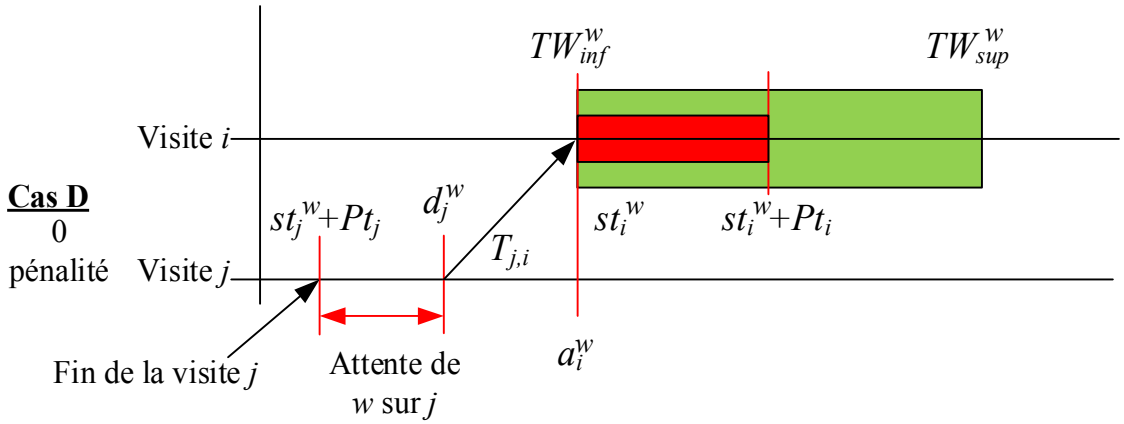


Figure 11 L'employé attend sur la visite précédente

Les cas E, F et G (illustrés sur la Figure 12) concernent les dates de début de la visite i en fonction de la borne supérieure (TW_{sup}^{w}) de la fenêtre de travail de l'employé w. L'employé arrive dans sa fenêtre de temps sur le nœud de la visite i. Il peut commencer la visite tout de suite (cas E), ou attendre (cas F et cas G). Dans le cas E, il n'y a pas de pénalité comptée, car la visite finit avant la fin de la fenêtre de travail de l'employé et débute après l'ouverture de celle-ci.

Dans le cas F (Figure 12), l'employé attend avant de commencer la visite i ($st_i^w > a_i^w$), mais la date de fin de la visite ($st_i^w + Pt_i$) ne dépasse pas la borne supérieure de la fenêtre de temps de l'employé ($st_i^w + Pt_i \leq TW_{sup}^w$). La solution a le même coût que dans le cas E, mais une date de début supérieure, donc elle est dominée par la solution du cas E.

Dans le cas G, l'employé attend avant de commencer la visite ($st_i^w > a_i^w$), et la date de fin de la visite ($st_i^w + Pt_i$) est supérieure à la borne supérieure de la fenêtre de temps de l'employé ($st_i^w + Pt_i > TW_{sup}^w$) : une pénalité est comptée. Le coût de cette solution est supérieur au coût de la solution du cas E, et la date de début est plus tard. Cette solution est donc dominée par le cas E et ne nécessite pas d'être prise en compte. En suivant le même raisonnement, tout autre cas avec une date st_i^w supérieure à celle du cas E est donc également dominé.

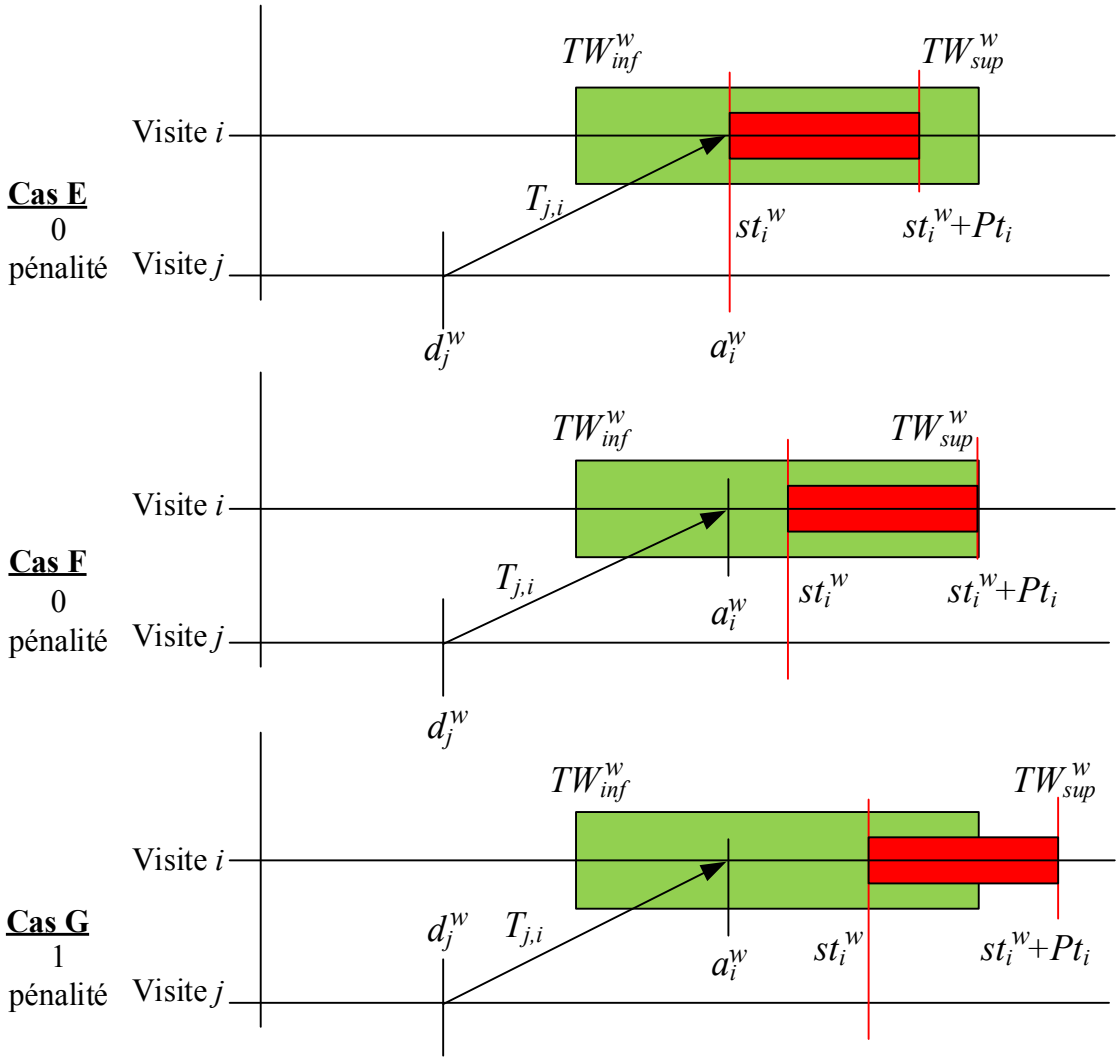


Figure 12 Date de début avec attente

Pour conclure cette section, la date de départ de l'employé de la visite et la date de début de la visite doivent respecter les déductions suivantes :

1) L'employé doit partir au plus tôt de la visite précédente, l'attente de celui-ci se fait sur la visite courante.
2) Soit l'employé réalise la visite en violant sa propre fenêtre de temps, soit il ne la viole pas. Mais dans les deux cas, il doit réaliser la visite au plus tôt (soit $st_i^w = a_i^w$, soit $st_i^w = TW_{inf}^w$).
3) Si l'employé peut commencer la visite sans violer la date de fin de sa fenêtre de travail (donc sans pénalité), il n'est pas nécessaire de prendre en compte des cas où l'employé viole sa fenêtre de travail.
4) Tous les autres cas ne nécessitent pas d'être pris en compte.

3.4. Fonction d'évaluation basée sur une procédure dynamique

Afin de calculer les dates de début des visites d'une tournée Γ, une nouvelle fonction d'évaluation, par programmation dynamique, d'une tournée est proposée. Cet algorithme est une extension de la procédure présentée par (Feillet et al., 2004) pour l'Elementary Shortest

Path Problem with Resource Constraints (ESPPRC) afin de prendre en compte la qualité de service de l'employé.

Cette nouvelle fonction d'évaluation d'une tournée est basée sur la création de labels représentant une partie de la tournée. Tous les labels l contenus dans l'ensemble Λ^i des labels du nœud i (les nœuds modélisent les visites) sont propagés au nœud suivant $j = \Gamma(i+1)$. Pour chaque label $l \in \Lambda^i$, deux nouveaux labels sont créés sur le nœud j. Le premier label généré correspond au cas où l'employé commence immédiatement la visite ($st_i^w = a_i^w$) ; tandis que le second label créé correspond au cas où l'employé attend l'ouverture de sa fenêtre de travail avant de commencer la visite ($st_i^w = TW_{inf}^w$). Ces deux labels correspondent donc aux cas A et B (respectivement Figure 8 et Figure 9 de la section 3.3).

Par abus, de langage $\Gamma(i) < \Gamma(j)$ signifie que la visite i est réalisée avant la visite j dans la tournée Γ. Un label sur un nœud j est défini par :

- La date arrivée A_i de l'employé sur chaque visite i précédant j ($\forall i \in \Gamma | \Gamma(i) \leq \Gamma(j)$).
- La date de début de service st_i de chaque visite i précédant j ($\forall i \in \Gamma | \Gamma(i) \leq \Gamma(j)$).
- La date de fin de service C_i de chaque visite i précédant j ($\forall i \in \Gamma | \Gamma(i) \leq \Gamma(j)$).
- La date de départ D_i de l'employé de chaque visite i précédent j ($\forall i \in \Gamma | \Gamma(i) \leq \Gamma(j)$).
- Le coût CT partiel de la tournée du dépôt jusqu'au nœud courant j.

Les labels sont propagés le long de la tournée. Lors de la propagation d'un label l positionné sur le nœud i vers le nœud j, les informations du nouveau label m sont mises à jour d'après les règles suivantes :

- $m.A_z = l.A_z, \forall z \in \Gamma | \Gamma(z) < \Gamma(j)$).
- $m.st_z = l.st_z, \forall z \in \Gamma | \Gamma(z) < \Gamma(j)$).
- $m.C_z = l.C_z, \forall z \in \Gamma | \Gamma(z) < \Gamma(j)$).
- $m.D_z = l.D_z, \forall z \in \Gamma | \Gamma(z) < \Gamma(j)$).
- $m.A_j = l.D_i + T_{i,j}$: date d'arrivée de l'employé sur la visite j.
- $m.st_j = \max(E_j ; m.A)$ où $m.st_j = \max(\mathrm{TW}_{\mathrm{w}}^{+} ; m.A_j)$: date de début de la visite suivant les deux cas.
- $m.C_j = m.st_j + Pt_j$: date de fin de la visite.
- $m.D_j = m.C_j$: date de départ de la visite.
- $m.CT = l.CT + \lambda_1(T_{i,j} + p_j^w) + \lambda_2(3 - \rho_j^w) + \lambda_3(\psi_j^w + \theta_j^w)$: coût de la tournée sur le nœud j.

L'Algorithme 1 est l'algorithme de principe pour calculer les dates donnant le coût minimal d'une tournée de type WSRP. La ligne 9 correspond à la création d'un label initial sur le nœud de départ I_w. Ce nœud est le premier élément de la tournée Γ de l'employé w. Le label de départ a toutes ses valeurs initialisées à 0. La première boucle, à la ligne 10, correspond au parcours des visites dans l'ordre de la tournée Γ. Celle-ci est privée de la dernière visite L_w, qui est le dépôt d'arrivée. Ligne 11, le nœud j est la visite suivant la visite i dans la tournée Γ. Tous les labels l du nœud i ($\forall l \in \Lambda^i$) sont propagés vers j. A_j (ligne 13) est la date d'arrivée de l'employé sur la visite j. Si cette date d'arrivée est plus petite que la fenêtre de temps $[E_j ; L_j]$ de j (ligne 14), alors un nouveau label m peut être créé (ligne 16) avec, pour date de début, $st_j = max(A_j ; E_i)$ (ligne 15). Ce nouveau label m est inséré dans la liste des labels du nœud j (ligne 17). Si la date de début st_j est inférieure à la fenêtre de temps $[TW_w^- ; TW_w^+]$ de l'employé (ligne 18), alors un nouveau label m' doit être créé (ligne 20) avec $st_j' = TW_w^-$

(ligne 19), puis inséré dans la liste des labels de j (ligne 21). Le meilleur label l_{best} contenu sur le nœud de dépôt d'arrivée ($l_{best} \in \Lambda^{L_w}$) est la tournée avec le coût minimal.

Algorithme 1 : Calculer le coût minimal d'une tournée de WSRP

1. **Sortie**
2. l_{best} : le label avec le coût minimal
3. **Entrée**
4. w : l'employé
5. Γ : la tournée ordonnée
6. **Variables**
7. Λ^i : file de labels sur le nœud i
8. **Début**
9. | l_{init} = creer_label_initial() ; // création du label initial sur le nœud de départ I_w
10. | **Pour** $\forall i \in \Gamma \backslash \mathrm{L_w}$ dans l'ordre de la tournée **Faire**
11. | | $j = \Gamma[i+1]$ // récupère le sommet suivant
12. | | **Pour** $\forall l \in \Lambda^i$ **Faire** // pour chaque label de i
13. | | | $A_j = l.D + T_{i,j}$ // date d'arrivée de l'employé sur j
14. | | | **Si** $A_j \leq L_j$ **Faire** // l'employé doit arriver avant la fin de la fenêtre
15. | | | | $st_j = max(A_j ; E_i)$ // date de début de la visite au plus tôt
16. | | | | m = nouveau_label(l, st_j) // création d'un nouveau label m
17. | | | | $\Lambda^j.push(m)$ // insertion de m dans les labels de j
18. | | | | **Si** $st_j < TW_w^-$ **Faire** // la visite débute avant la fenêtre de l'employé
19. | | | | | $st_j{}' = TW_w^-$ // date de début de la visite
20. | | | | | m' = nouveau_label($l, st_j{}'$) // création d'un nouveau label m'
21. | | | | | $\Lambda^j.push(m')$ // insertion de m' dans les labels de j
22. | | | | **FinSi**
23. | | | **FinSi**
24. | | **FinPour**
25. | **FinPour**
26. | **Retourner** $l_{best} \in \Lambda^{L_w}$ // retour du label contenu sur L_w de coût minimal
27. **Fin**

4. Méthodes exactes pour le WSRP (modèles PLNE et PPC)

4.1. PLNE pour le WSRP (Garaix et al., 2018b)

Cette section introduit un modèle PLNE pour le WSRP (Garaix et al., 2018b). La première partie de la section concerne les variables, la seconde partie la fonction objectif et la troisième partie se focalise sur les contraintes.

4.1.1. Les variables pour le modèle PLNE

Le modèle PLNE utilise quatre variables binaires et une variable continue. Les variables A_i, C_i et D_i associées à la visite i, ne sont pas conservées dans la formulation PLNE car la position et la durée des temps d'attente des employés ne sont pas prises en compte dans la fonction objectif. Par conséquent, l'employé peut attendre où il souhaite, et afin de réduire le

nombre de variables ainsi que l'espace des solutions, il est décidé que l'employé ne peut jamais attendre sur le sommet d'origine. La date C_i de fin de la visite est directement déduite de la date de début st_i ($C_i = st_i + Pt_i$, avec Pt_i le processing time de i). La date de départ D_i de la visite est égale à la date de fin de la visite C_i ($D_i = C_i$). En effet, il n'y a pas de coût d'attente associé à l'employé, donc l'employé peut toujours partir au plus tôt et attendre une fois arrivée à la prochaine visite avant de débuter cette dernière. La date d'arrivée A_i de l'employé sur une visite est calculée à partir de la date de départ D_j de la visite j précédente : $A_i = D_j + T_{j,i}$ (où $T_{j,i}$ est le temps de transport de la visite j à la visite i), or $D_j = C_j = st_j + Pt_j$, donc $A_i = st_j + Pt_j + T_{j,i}$.

Les variables binaires :

- $x_{i,j}^w = 1$ si l'employé w se déplace de la visite $i \in V$ à la visite $j \in V$; $x_{i,j}^w = 0$ sinon.
- $\psi_i^w = 1$ si l'employé w réalise la visite $i \in V$ qui est située en dehors de sa zone géographique de préférences ; $\psi_i^w = 0$ sinon.
- $\theta_i^w = 1$ si l'employé w viole sa fenêtre de temps lorsqu'il effectue la visite $i \in V$; $\theta_i^w = 0$ sinon.
- $y_i = 1$ si la visite i n'est pas réalisée.

Les variables continues :

- st_i est la date de début de la visite $i \in V$.

Par ailleurs, M est une constante entière suffisamment grande.

4.1.2. La fonction objectif pour le modèle PLNE

La fonction objectif f est définie par (Laesanklang et al., 2015b; Garaix et al., 2018b) et repose sur quatre critères : les coûts de production, un coût de préférence entre clients et employés, des pénalités en fonction des violations des fenêtres de temps et des zones géographiques souhaitées par les employés, et une pénalité en fonction des visites non effectuées. Cette fonction objectif est une somme agrégée de ces quatre critères. Elle est très différente des fonctions objectifs « classiques » des problèmes de tournées de véhicules ou des problèmes d'ordonnancement qui portent régulièrement sur des critères de durée et/ou de délais. La fonction objectif définie par (Laesanklang et al., 2015b) permet d'obtenir une solution qui est un compromis entre les coûts opérationnels, la qualité de service pour les employés et les clients, et enfin le nombre de visites effectuées.

$$f = \lambda_1 \sum_{i \in V \cup I_w \cup L_w} \sum_{j \in V \cup I_w \cup L_w} \left(T_{i,j} + p_i^w\right) x_{i,j}^w + \lambda_2 \sum_{i \in \mathrm{V}} \left(3 - \sum_{\mathrm{j} \in V \cup L_w} \rho_i^w x_{i,j}^w\right)$$

$$+\lambda_3 \sum_{i \in \mathrm{V}} (\psi_i^w + \theta_i^w) + \lambda_4 \sum_{i \in \mathrm{V}} y_i$$

$\left(T_{i,j} + p_i^w\right) x_{i,j}^w$ représente le coût opérationnel si un employé w réalise les visites i et j où $T_{i,j}$ est la distance entre ces deux visites et p_i^w est le coût si l'employé w effectue la visite i. Ce critère est minimisé. Ce critère de coût opérationnel est la somme d'un critère issu des problèmes de tournées de véhicules et d'un critère issu des problèmes d'ordonnancement, respectivement la somme des distances parcourues et le coût de production.

$\rho_i^w \in [0\,;\,3]$ est un critère de qualité de service défini entre le client de la visite $i \in V$ et l'employé w (cf. section 2.1). Un ρ_i^w proche de 3 signifie que l'association entre i et w est pleinement satisfaisante. Une valeur proche de 0 signifie au contraire que cette association n'est pas appréciée. L'objectif est de maximiser les préférences, et par conséquent, cela se traduit dans la fonction objectif par minimiser $3 - \rho_i^w$.

La troisième partie de la fonction objectif concerne la qualité de service de l'employé. ψ_i^w et θ_i^w sont des variables binaires valant 1, respectivement, si l'employé w réalise la visite i qui est située en dehors de sa zone de préférence géographique, et si l'employé effectue une visite en dehors de sa propre fenêtre de temps (voir la section 3.2). Ces pénalités sont à minimiser.

y_i est une variable binaire valant 1 si la visite i n'est pas réalisée, sinon elle vaut 0. La somme des y_i est à minimiser. Ce terme permet de maximiser le nombre de visites effectuées.

Le fait que certaines visites puissent ne pas être traitées, et le fait de prendre en compte un critère de qualité de service pour les employés, sont des différences importantes par rapport aux problèmes classiques de tournées de véhicules de la littérature.

Les quatre critères sont hiérarchisés par les coefficients λ_1, λ_2, λ_3 et λ_4, avec $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$. La priorité la plus basse est accordée à la minimisation du coût opérationnel, et la priorité la plus haute est accordée à la maximisation du nombre de visites effectuées.

4.1.3. Les contraintes pour le modèle PLNE

Les contraintes de type flot

Les contraintes (1), (2) et (3) de type flot garantissent la faisabilité des tournées des employés. Ces contraintes sont typiques des modèles mathématiques des problèmes de tournées de véhicules.

La contrainte (1) garantit qu'il n'y a qu'un seul arc sortant du sommet $i \in V$, c'est-à-dire qu'au maximum un employé peut effectuer la visite i.

$$\sum_{j \in V \cup L_w} x_{i,j}^w \leq 1 \qquad \forall i \in V,\ \forall w \in W \qquad (1)$$

La contrainte (2) certifie qu'il y a au plus un arc entrant en chaque sommet $j \in V$ du graphe.

$$\sum_{i \in V \cup I_w} x_{i,j}^w \leq 1 \qquad \forall j \in V,\ \forall w \in W \qquad (2)$$

La contrainte (3) impose qu'il y ait autant d'arcs sortants (un au maximum d'après la contrainte (1)) que d'arcs entrants (un au maximum d'après la contrainte (2)) : un employé w réalisant la visite i doit obligatoirement repartir.

$$\sum_{j \in V \cup L_w} x_{i,j}^w = \sum_{j \in V \cup I_w} x_{j,i}^w \qquad \forall i \in V,\ \forall w \in W \qquad (3)$$

Les contraintes de pénalités

La contrainte (4) calcule le ψ_i^w de l'employé w pour la visite i et la variable $\psi_i^w = 1$ si la visite $i \in V$ n'est pas située dans la zone de préférence de w. La variable ψ_i^w intervient dans la fonction objectif.

$PA_i^w = 0$ signifie que la visite i n'est pas dans une zone géographique souhaitée par l'employé w, et dans ce cas ψ_i^w doit valoir 1. Si l'employé w effectue la visite i avec $PA_i^w = 0$, alors la contrainte (4) peut se réécrire $\psi_i^w \geq \sum_{j \in V \cup I_w} x_{j,i}^w$ et il existe une visite $i \in V$ telle que $x_{j,i}^w = 1$, donc $\sum_{j \in V \cup I_w} x_{j,i}^w = 1$, et par conséquent $\psi_i^w \geq 1$. Si $PA_i^w = 1$ alors la contrainte est trivialement respectée, car elle peut s'écrire $\psi_i^w + M \geq \sum_{j \in V \cup I_w} x_{j,i}^w$, et la fonction objectif impose $\psi_i^w = 0$ (car l'objectif est de minimiser la somme des ψ_i^w).

$$\psi_i^w + M.PA_i^w \geq \sum_{j \in V \cup I_w} x_{j,i}^w \qquad \forall i \in V,\ \forall w \in W \qquad (4)$$

Les contraintes (5) et (6) vérifient si un employé ne respecte pas sa fenêtre de temps en réalisant une visite. La contrainte (5) est dédiée à la violation de la fenêtre de temps inférieure de l'employé. Si l'employé $w \in W$ effectue la visite $i \in V$, alors $\sum_{j \in V \cup I_w} x_{j,i}^w = 1$ et la contrainte (5) peut se réécrire : $M.\theta_i^w \geq TW_{inf}^w - st_i$. Si la date de début st_i de la visite i est strictement plus petite que la borne inférieure de la fenêtre de temps, alors $TW_{inf}^w - st_i > 0$, ce qui implique que $\theta_i^w = 1$. Le même raisonnement peut s'appliquer à la contrainte (6). Si l'employé $w \in W$ effectue la visite $i \in V$, alors $\sum_{j \in V \cup I_w} x_{j,i}^w = 1$ et (6) devient : $M.\theta_i^w \geq st_i + Pt_i - TW_{sup}^w$. Si la date de fin $(st_i + Pt_i)$ est strictement supérieure à la fenêtre de temps de l'employé alors $st_i + Pt_i - TW_{sup}^w > 0$ et impose que $\theta_i^w = 1$.

$$M.\theta_i^w \geq TW_{inf}^w - st_i + \left(\sum_{j \in V \cup I_w} x_{j,i}^w - 1 \right).M \qquad \forall i \in V,\ \forall w \in W \qquad (5)$$

$$M.\theta_i^w \geq st_i + Pt_i - TW_{sup}^w + \left(\sum_{j \in V \cup I_w} x_{j,i}^w - 1 \right).M \qquad \forall i \in V,\ \forall w \in W \qquad (6)$$

La contrainte (7) assure qu'un employé $w \in W$ ne peut réaliser qu'une visite $i \in V$ qui est autorisée d'après le contrat. Dans ce cas $CoTr_i^w = 1$. Si $CoTr_i^w = 0$, alors w ne peut pas réaliser la visite i, et $\sum_{j \in V \cup I_w} x_{j,i}^w \leq 0$ impose que $\forall j \in V$, $x_{j,i}^w = 0$.

$$\sum_{j \in V \cup I_w} x_{j,i}^w \leq CoTr_i^w \qquad \forall i \in V,\ \forall w \in W \qquad (7)$$

La contrainte (8) compte le nombre de visites non effectuées. Si la visite n'est pas réalisée, alors la somme des flots entrant sur cette visite est nulle $(\sum_{j \in V \cup I_w} x_{j,i}^w = 0)$, ce qui implique que $y_i = 1$.

$$y_i + \sum_{j \in V \cup I_w} x_{j,i}^w = 1 \qquad \forall i \in V,\ \forall w \in W \qquad (8)$$

Les contraintes sur les dates :

La contrainte (9) impose que la date de début de la visite $j \in V$, succédant à la visite $i \in V$ dans la tournée de l'employé w ($x_{ij}^{w} = 1$), soit supérieure à la date de fin ($st_i + Pt_i$) plus le temps de transport ($T_{i,j}$) de i vers j. Cette contrainte définit implicitement la date d'arrivée de l'employé au plus tôt sur la visite j.

$$st_i + Pt_i + T_{i,j} \leq st_j + \left(x_{i,j}^{w} - 1\right).M \qquad \forall (i,j) \in V^2 | i \neq j,\ \forall w \in W \qquad (9)$$

Les contraintes (10) et (11) garantissent que la date de début st_i de la visite i est comprise dans la fenêtre de temps $[E_i\ ;\ L_i\ ]$ de la visite.

$$st_i \geq E_i \qquad \forall i \in V,\ \forall w \in W \qquad (10)$$

$$st_i \leq L_i \qquad \forall i \in V,\ \forall w \in W \qquad (11)$$

4.2. Modèle de PPC pour le WSRP

La PPC permet de modéliser des problèmes sous forme non linéaire, et est basée sur des algorithmes de filtrage et de propagation de contraintes. Brièvement, un modèle PPC est défini par : l'ensemble des variables du problème $X = \{x_1, x_2, \dots, x_n\}$; l'ensemble des domaines associés à chaque variable $D = \{D_1, D_2, \dots, D_n\}$; et l'ensemble des contraintes $C = \{C_1, C_2, \dots, C_m\}$ (Régin, 2004; Bourreau et al., 2019b, 2020).

La résolution d'un modèle PPC est équivalente à un parcours arborescent d'un arbre binaire. Chaque nœud de l'arbre correspond à une solution partielle où :

- Un sous-ensemble de variables $X^* \subseteq X$ où les variables sont instanciées, c'est-à-dire, qu'elles prennent une unique valeur déterminée. Leur domaine est réduit à un singleton.
- Un sous-ensemble de variables $X' \subseteq X$ où les variables ne sont pas instanciées, c'est-à-dire que leur domaine comporte plusieurs valeurs.
- $X^* \cup X' = X$ et $X^* \cap X' = \emptyset$: chacune des variables doit appartenir à un unique sous-ensemble.

Lorsque toutes les variables sont instanciées à une unique valeur, alors ce nœud représente une solution. Deux nœuds de l'arbre sont reliés par un branchement correspondant : 1) à la sélection d'une variable dont le domaine n'est pas un singleton ; 2) à l'affectation d'une valeur à cette variable ; 3) au filtrage des domaines des variables ayant une contrainte en commun avec la variable sélectionnée ; et 4) à la propagation des modifications des domaines aux autres variables. Un nœud contenant une ou plusieurs variables avec un domaine vide est une solution partielle qui ne permet par d'obtenir une solution entière respectant toutes les contraintes du modèle et les branchements précédemment faits.

La force des solveurs de PPC réside dans leur capacité, après branchement d'une variable x_b, à filtrer le domaine des variables $x_c \in X$ qui ont une contrainte en commun avec x_b, puis de propager les éventuelles modifications en filtrant les domaines des variables $x_{c'} \in X$ qui ont une contrainte en commun avec x_c. Le filtrage et la propagation permettent de réduire les domaines des variables, et donc de réduire la taille de l'arbre de recherche.

La Figure 13 illustre le principe de la PPC sur un problème à quatre variables binaires. Les nœuds 1, 2 et 3, par exemple, représentent des solutions partielles tandis que le nœud 4 est

une solution finale (et complète). Le nœud 6 est une feuille de l'arbre car le domaine d'une variable est vide à cause des branchements a et b.

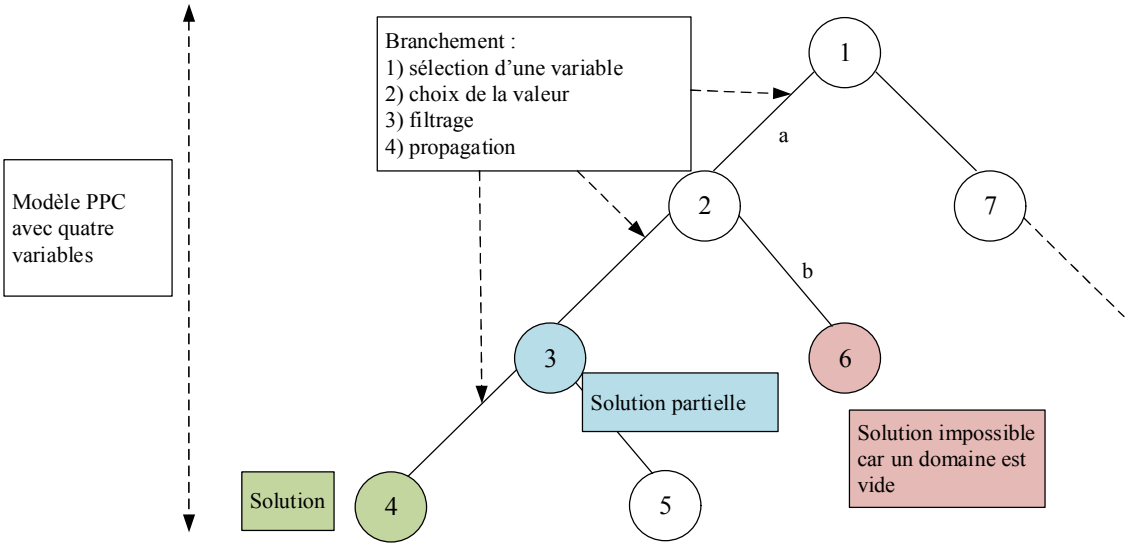


Figure 13 Principe de la PPC

Un problème très contraint permet généralement de bons filtrages et propagations, et la PPC est souvent très performante pour fournir rapidement des solutions de bonne qualité (Bockmayr and Hooker, 2005; Pour et al., 2018; Bourreau et al., 2020).

La résolution des modèles PPC est fortement dépendante des contraintes utilisées pour modéliser les relations entre les variables (certaines contraintes permettent plus de propagations que d'autres), et de l'ordre de branchement (sélection de la variable). Si le branchement a de la Figure 13 permet de nombreuses propagations, alors la taille du sous-arbre ayant pour racine le nœud 2 sera petite. Enfin la résolution dépend également du choix de la valeur de la variable sélectionnée.

De nombreux articles de problèmes d'ordonnancement ou de tournées de véhicules proposent des modélisations PPC et/ou des méthodes hybridées avec une partie en PPC. Récemment, (Zhao and Li, 2014) et (Wang et al., 2015) proposent des modèles PPC pour la planification, respectivement, d'opérations chirurgicales et des blocs opératoires. (Topaloglu and Ozkarahan, 2011) introduisent une génération de colonnes avec le sous-problème résolu par PPC pour un problème d'affectation d'étudiants à des stages. Dans le même courant, (He and Qu, 2012) formulent une génération de colonnes avec le sous-problème résolu en PPC pour le nurse rostering problem. (Novas and Henning, 2014) utilisent la PPC pour résoudre un problème de production avec transport. (Gacias et al., 2010) étudient les algorithmes de type raisonnement énergétique pour la PPC sur un problème d'ordonnancement de machines parallèles avec des contraintes de précédences et des temps d'installation. (Pour et al., 2018) proposent l'utilisation de la PPC pour obtenir rapidement une solution qui est utilisée comme point de départ pour la résolution PLNE d'un problème de planification d'équipe de maintenance sur le réseau ferroviaire danois. (Gedik et al., 2018) introduisent un modèle PPC pour le non-preemptive unrelated parallel machine scheduling problem. (Hojabri et al., 2018) utilisent la PPC comme recherche locale pour un problème de tournées de véhicules avec contraintes de synchronisation. Cette recherche locale est insérée dans un schéma de résolution

globale de type métaheuristique. Tous ces articles démontrent l'intérêt et la capacité de la PPC à résoudre des problèmes de types tournées de véhicules, affectation, ou planification, et laissent penser que la PPC peut être également performante sur le WSRP.

Pour plus de détails sur la PPC, le lecteur peut approfondir le sujet avec l'ouvrage de (Rossi et al., 2006) ou les ouvrages de (Bourreau et al., 2019b, 2020).

4.2.1. Principes de la modélisation du WSRP avec la PPC

La modélisation en PPC d'un problème de WSRP s'apparente à la modélisation PPC des problèmes de tournées de véhicules et notamment du VRP (Vehicle Routing Problem) (Hojabri et al., 2018; Bourreau et al., 2019a, 2020).

Chaque visite est représentée par un nœud et est associée à quatre variables (da_i, st_i, df_i et dd_i) définissant la date d'arrivée (da_i) de l'employé, la date de début de la visite (st_i), la date de fin (df_i) de la visite, et la date (dd_i) de départ de l'employé ; et à trois variables (a_i, s_i et p_i) permettant de définir les tournées. Les dates a_i, st_i, df_i et dd_i nécessitent d'être explicites, car une tournée n'est pas uniquement définie par l'ordre des visites, mais également par ces dates. La présence explicite de ces dates est fréquente pour les problèmes de tournées de véhicules avec des critères de qualité de service où la solution n'est pas semi-active : (Cordeau and Laporte, 2003; Gondran et al., 2018).

Les quatre variables classiques des problèmes de tournées de véhicules liées aux dates de la visite i sont (Figure 14) :

- $A_i = da_i$: la date d'arrivée de l'employé sur le sommet i.
- $B_i = st_i$: la date de début de service sur le sommet i.
- $C_i = df_i$: la date de fin de service sur le sommet i.
- $D_i = dd_i$: la date de départ de l'employé du sommet i.

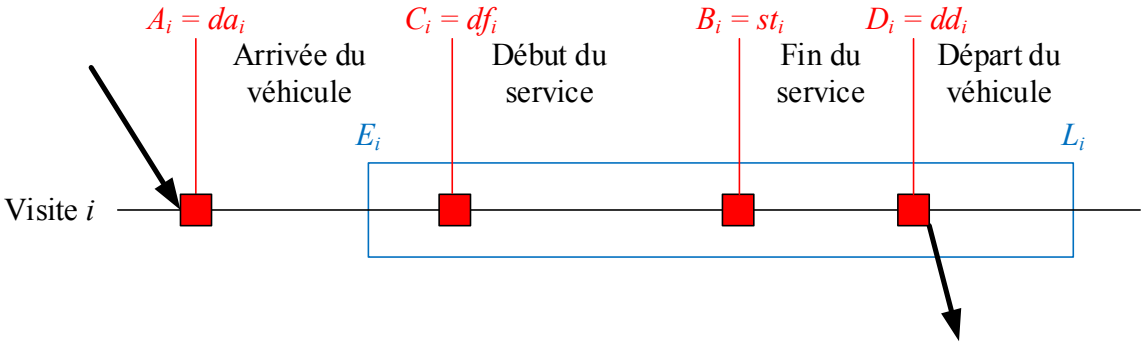


Figure 14 Variables liées aux dates de la visite

De plus, une visite i est caractérisée par :

- a_i, son affectation à un véhicule (ou employé).
- s_i, le successeur de la visite dans la tournée.
- p_i, sa position (ou rang) dans la tournée du véhicule (ou de l'employé).

Ces variables a_i, s_i et p_i sont communément utilisées pour modéliser des problèmes de type tournées de véhicules en programmation par contraintes et proposent des modélisations efficaces (Rousseau et al., 2013; Hojabri et al., 2018; Bourreau et al., 2019a, 2019b, 2020). La

variable a_i permet d'obtenir l'employé (ou le véhicule, suivant le contexte) affecté à la visite i, cette valeur est un entier. La variable s_i est le successeur de la visite i dans la tournée. Ces deux variables (a_i et s_i) « traduisent » les variables binaires de type $x_{i,j}^w$ fréquentes dans les modélisations PLNE des problèmes de tournées de véhicules et de manière plus générale dans les problèmes de flot dans un graphe. Le nombre de variables a_i et s_i est de l'ordre $2|V|$ (où V est l'ensemble des visites) tandis que le nombre de variables $x_{i,j}^w$ dans les modèles PLNE est de l'ordre $|W||V|^2$ (où W est l'ensemble des employés). De plus, toutes les variables a_i et s_i traduisent directement des informations tandis que la majorité des variables $x_{i,j}^w$ est nulle dans la solution. Les variables p_i permettent d'éviter les sous-tours : si deux visites i et j se suivent dans une tournée ($s_i = j$), alors $a_i = a_j$ et $p_j = p_i + 1$.

La Figure 15 illustre les variables liées aux tournées pour les problèmes de tournées de véhicules. Dans cet exemple, il y a cinq visites et deux employés (ou véhicules), chacun effectuant une tournée. La visite 1 est affectée à l'employé w_1 ($a_1 = w_1$), son successeur est la visite 3 ($s_1 = 3$), et son rang dans la tournée est 1 ($p_1 = 1$) car c'est la première visite de l'employé w_1. La visite 5 est la dernière visite de la tournée de w_1 et est en position 3 ($p_5 = 3$).

Ces variables entières et les contraintes non linéaires (qui sont présentées en section 4.2.3) permettent une modélisation efficace des problèmes de tournées de véhicules (Rousseau et al., 2013; Hojabri et al., 2018; Bourreau et al., 2019a, 2019b, 2020).

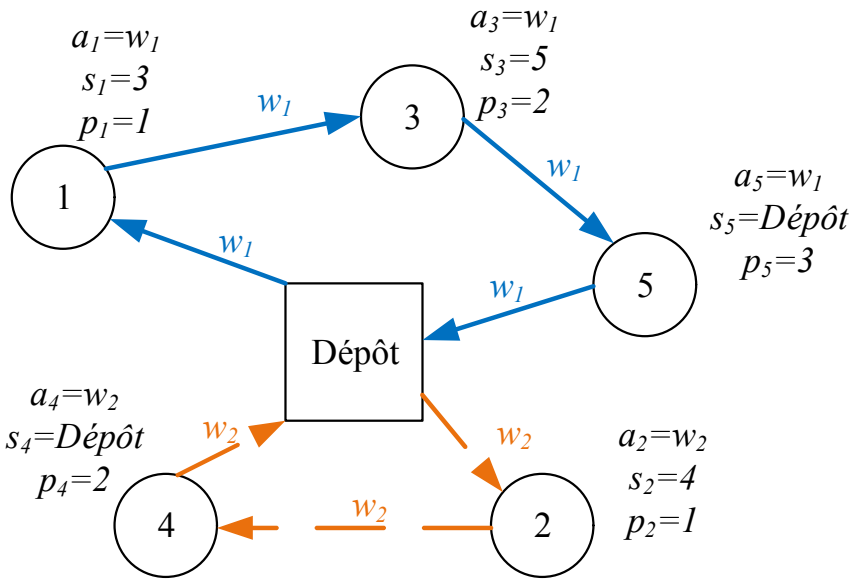


Figure 15 Variables liées aux tournées de véhicules en PPC

Les visites sont numérotées de 1 à $|V|$. Les dépôts du WSRP peuvent être tous différents (incluant le dépôt de départ et le dépôt d'arrivée d'un même employé). Chacun des dépôts est donc numéroté de $|V| + 1$ à $|V| + 2|W|$. Les dépôts numérotés de $I^d = |V| + 1$ à $I^f = |V| + |W|$ sont les dépôts de départ des employés. Enfin, les dépôts numérotés de $L^d = |V| + |W| + 1$ à $L^f = |V| + 2|W|$ sont les dépôts d'arrivée des employés. Les dépôts de l'employé w sont $|V| + w$ (dépôt de départ) et $|V| + |W| + w$ (dépôt d'arrivée). La Figure 16 présente deux tournées effectuées par deux employés où leurs dépôts de départ et d'arrivée sont différents (ils peuvent être situés sur des lieux géographiquement différents). L'employé w_1 débute sa tournée au dépôt 6 et la finit au dépôt 8.

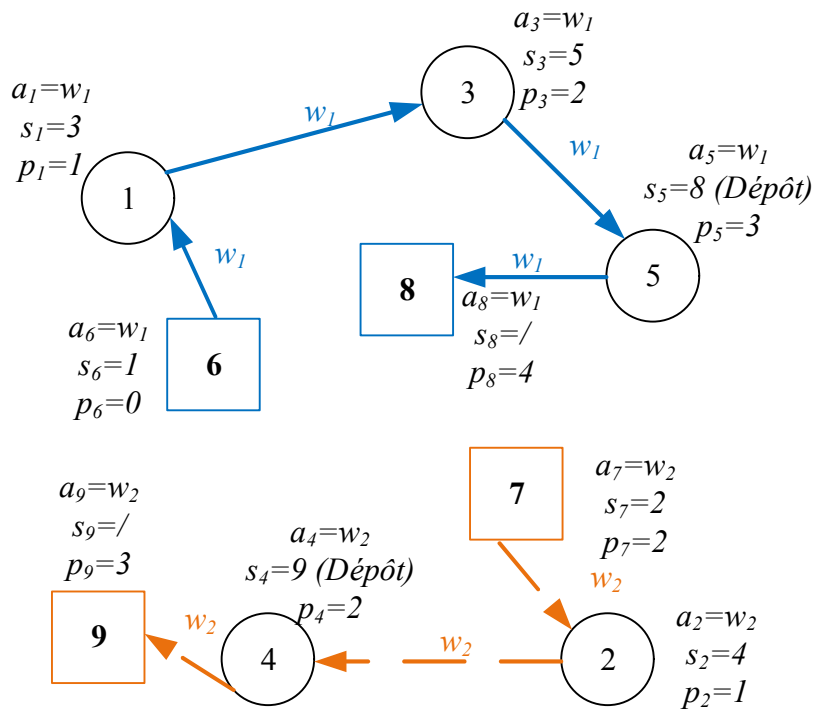


Figure 16 Numérotation des dépôts de départ et d'arrivée

Un modèle de PPC est défini par l'ensemble des variables, la définition de leur domaine, les contraintes et la fonction objectif. Le domaine de chaque variable est introduit dans la section 4.2.2, les contraintes sont présentées dans la section 4.2.3 et la fonction objectif est définie dans la section 4.2.4. De plus, quelques contraintes permettant de renforcer la modélisation PPC sont présentées dans la section 4.2.5.

4.2.2. Définition des variables pour le modèle PPC

Afin de rendre le modèle plus facile à la lecture et plus efficace, deux variables supplémentaires sont introduites : les variables dp_i et les variables $pred_i$ (Bourreau et al., 2019a, 2020).

La variable dp_i est la distance entre la visite i et son successeur s_i : $dp_i = T_{i,s_i}$. Cette variable permet d'élaborer des stratégies de parcours de l'arbre, basées sur les distances entre les visites.

La variable $pred_i$ est « l'inverse » de la variable du successeur (s_i), et désigne le prédécesseur de i. Cet ensemble de variables n'est pas obligatoire, mais celles-ci servent à renforcer le modèle en propageant les informations en « remontant » le long des tournées. Si une information est déduite sur une visite i, les variables successeurs et les contraintes associées transmettent l'information vers les visites succédant i dans la tournée, mais cette information ne peut pas être transmise vers les visites qui précédent i dans la tournée. Grâce aux variables $pred_i$, l'information contenue sur une visite i peut être propagée aux visites la précédant. La création de la variable $pred_i$ impose de former des tournées « fermées », afin d'obtenir une bijection entre les successeurs et les prédécesseurs. La bijection permet une modélisation efficace pour les solveurs PPC. Le successeur d'un dépôt d'arrivée est le dépôt de départ et le prédécesseur d'un dépôt de départ est le dépôt d'arrivée (voir Figure 17).

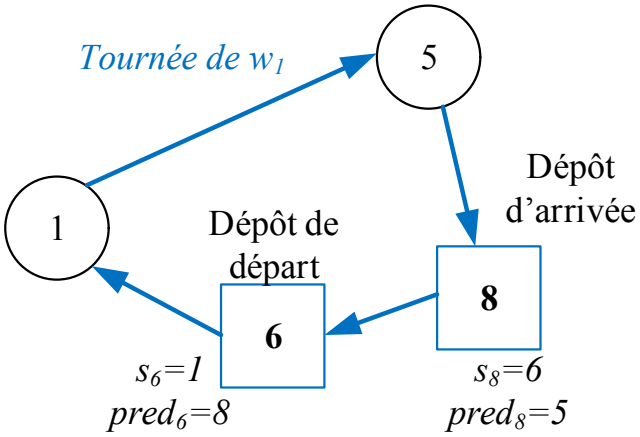


Figure 17 Tournée « fermée »

En résumé, les variables du modèle PPC sont les suivantes :

- a_i, l'affectation de la visite i à un employé.
- s_i, le successeur de la visite i.
- p_i, la position de la visite i dans sa tournée.
- $pred_i$, le prédécesseur de la visite i.
- da_i, la date d'arrivée de l'employé sur le sommet i.
- st_i, la date de début de service sur le sommet i.
- df_i, la date de fin de service sur le sommet i.
- dd_i, la date de départ de l'employé du sommet i.
- dp_i, la distance entre i et son successeur s_i.
- d, la distance totale à minimiser.

H est une constante entière suffisamment grande, représentant l'horizon de temps.

Afin de faciliter la compréhension, la déclaration des domaines des variables est divisée en trois parties : la définition des variables pour les visites, suivie de la définition des variables pour les dépôts de départ, et enfin la définition des variables pour les dépôts d'arrivée. Les déclarations des variables sont différentes, car certaines valeurs sont impossibles suivant le type du nœud : par exemple, le successeur d'un dépôt de départ ne peut pas être un autre dépôt de départ. Dans un souci de clarté, les définitions des domaines sont notées (DV), et les contraintes du modèle PPC sont numérotées (PPC).

Le sommet 0 est fictif. Toutes les variables du sommet 0 sont initialisées à 0 (DV 1).

$$\{a_0 ; s_0 ; p_0 ; pred_0 ; da_0 ; st_0 ; ft_0 ; ddp_0 ; dp_0\} = 0 \qquad \text{(DV 1)}$$

Définition des variables pour les visites

Les définitions des variables concernant les visites (hors dépôts) sont numérotées de (DV 2) à (DV 12).

Une visite peut être affectée à n'importe quel employé numéroté de 1 à $|W|$ (DV 2). Le successeur $s_i = j$ d'une visite est soit une autre visite ($j \in V$) soit un dépôt d'arrivée ($j \in L$) (DV 3). Le rang p_i d'une visite dans une tournée varie entre 1 (car le dépôt de départ est au rang 0) et $|V|$ qui est le nombre de visites (cas où toutes les visites sont dans la même tournée)

(DV 4). Le prédécesseur $pred_i = j$ d'une visite i est soit une autre visite ($j \in V$), soit un dépôt de départ ($j \in I$) (DV 5).

(DV 6) définit la date d'arrivée de l'employé sur la visite i. Celle-ci doit être inférieure à borne supérieure de la fenêtre de temps de la visite ($[E_i\ ;\ L_i]$). L'employé peut arriver sur la visite avant sa borne inférieure, mais il ne peut commencer celle-ci avant la date E_i d'ouverture de la fenêtre de temps (DV 7).

La date de fin df_i de la visite doit être dans la fenêtre de temps $[E_i + Pt_i\ ;\ L_i + Pt_i]$ (DV 8). La date de départ dd_i de la visite doit être supérieure à la date de début au plus tôt de la visite (E_i), mais rien n'oblige l'employé à partir directement, il peut très bien attendre sur la visite (DV 9). La distance entre la visite dp_i et la visite suivante peut prendre n'importe qu'elle valeur comprise entre 0 et la distance maximale (DV 10).

La définition (DV 11) réduit les domaines des s_i en fonction des fenêtres de temps : si la fenêtre de temps de la visite i est située « plus tard » que la fenêtre de temps de la visite j alors, j ne peut pas être le successeur de i (comme le montre la Figure 18). (DV 12) assure qu'un employé w peut être affecté à une visite i d'après les données du problème ($CoTr_i^w = 1$).

$a_i = \{1\ ;\ \|W\|\}$	$\forall i \in V$	(DV 2)
$s_i = \{1\ ;\|V\|\} \cup \{L^d\ ;L^f\}$	$\forall i \in V$	(DV 3)
$p_i = \{1\ ;\|V\|\}$	$\forall i \in V$	(DV 4)
$pred_i = \{1\ ;\|V\|\} \cup \{I^d\ ;I^f\}$	$\forall i \in V$	(DV 5)
$da_i = [0\ ;L_i]$	$\forall i \in V$	(DV 6)
$st_i = [E_i\ ;\ L_i]$	$\forall i \in V$	(DV 7)
$df_i = [E_i + Pt_i\ ;\ L_i + Pt_i]$	$\forall i \in V$	(DV 8)
$dd_i = [E_i + Pt_i\ ;H]$	$\forall i \in V$	(DV 9)
$dp_i = [0\ ;H]$	$\forall i \in V$	(DV 10)
$E_i > L_j => s_i \neq j$	$\forall (i,j) \in V^2 \| i \neq j$	(DV 11)
$a_i \neq w$	$\forall i \in V, \forall w \in W \| CoTr_i^w = 0$	(DV 12)

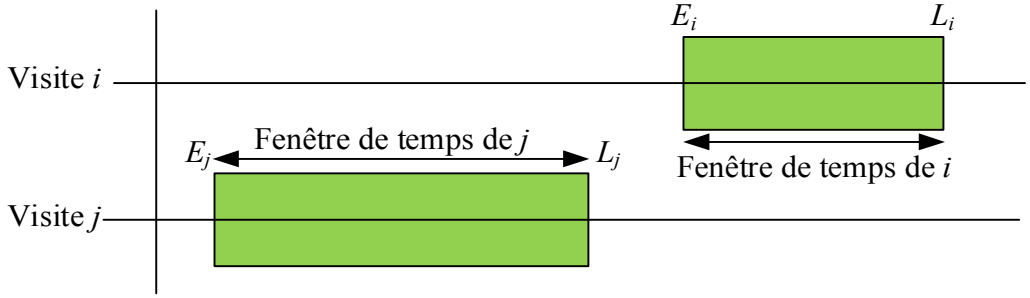


Figure 18 j successeur impossible de i

Définition des variables pour les dépôts de départ

Les dépôts de départ sont numérotés de $I^d = |V| + 1$ à $I^f = |V| + 2|W|$ (où $|V|$ est le nombre de visites et $|W|$ le nombre d'employés) et sont automatiquement affectés à un employé. Le dépôt de départ de l'employé $w = 1$ est $I_{w=1} = |V| + 1$. Le dépôt i est donc affecté à l'employé $i - |V|$. Par exemple, sur la Figure 19, le dépôt de départ de l'employé w_1 est le sommet numéro 6. Le sommet 6 est affecté à l'employé w_1 car $i - |V| = 6 - 5 = 1$. L'affectation du dépôt de départ à un employé est réalisée par (DV 13).

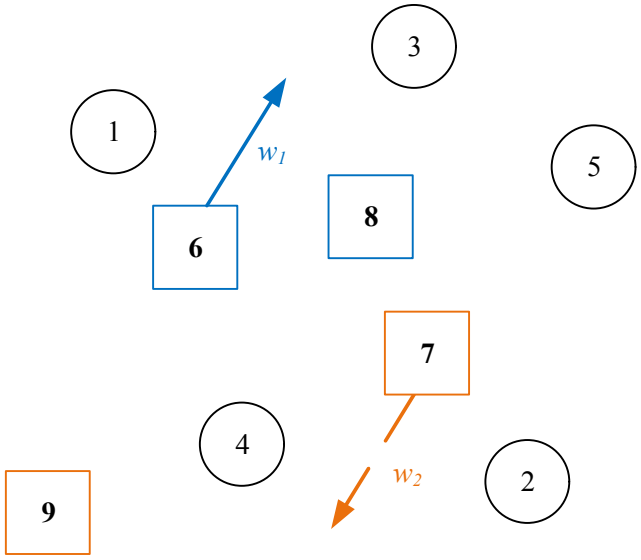


Figure 19 Affectation des dépôts de départ

Le successeur d'un dépôt de départ (DV 14), numéroté i, est soit une visite (numérotée de 1 à $|V|$) soit le dépôt d'arrivée associé au même employé (numéroté $i + |W|$). Le rang p_i d'un dépôt de départ est toujours 0 (DV 15). Le prédécesseur d'un dépôt de départ est le dépôt d'arrivée (numéro $i + |W|$), associé au même employé (DV 16). Les dates d'arrivée da_i, de début st_i, de fin df_i et de départ dd_i sont toutes nulles ((DV 17) à (DV 19)). La distance entre un sommet de départ et le prochain sommet peut prendre n'importe quelle valeur (DV 21).

$$a_i = i - |V| \qquad \forall i \in I \qquad \text{(DV 13)}$$
$$s_i = \{1; |V|\} \cup \{i + |W|\} \qquad \forall i \in I \qquad \text{(DV 14)}$$
$$p_i = 0 \qquad \forall i \in I \qquad \text{(DV 15)}$$
$$pred_i = i + |W| \qquad \forall i \in I \qquad \text{(DV 16)}$$
$$da_i = 0 \qquad \forall i \in I \qquad \text{(DV 17)}$$
$$st_i = 0 \qquad \forall i \in I \qquad \text{(DV 18)}$$
$$df_i = 0 \qquad \forall i \in I \qquad \text{(DV 19)}$$
$$dd_i = 0 \qquad \forall i \in I \qquad \text{(DV 20)}$$
$$dp_i = [0\,; H] \qquad \forall i \in I \qquad \text{(DV 21)}$$

Il est important de noter que pour certains problèmes de tournées de véhicules (et notamment les instances de VRPTW de (Bredström and Rönnqvist, 2007)), l'employé (ou le véhicule) ne peut pas partir du dépôt avant le début de sa fenêtre de temps $[TW_w^{inf}; TW_w^{sup}]$. Les dates da_i, st_i, df_i et dd_i peuvent donc être initialisées avec la valeur TW_w^{inf} ((DV 17B) à (DV 20B)).

$$da_i = TW_w^{inf} \qquad \forall i \in I | i = I_w \qquad \text{(DV 17B)}$$
$$st_i = TW_w^{inf} \qquad \forall i \in I | i = I_w \qquad \text{(DV 18B)}$$
$$df_i = TW_w^{inf} \qquad \forall i \in I | i = I_w \qquad \text{(DV 19B)}$$
$$dd_i = TW_w^{inf} \qquad \forall i \in I | i = I_w \qquad \text{(DV 20B)}$$

Définition des variables pour les dépôts d'arrivée

Similairement aux dépôts de départ, les dépôts d'arrivée sont affectés à un employé donné (DV 22). Un dépôt d'arrivée i est affecté à l'employé $a_i = i - (|V| + |W|)$, car les dépôts

d'arrivée sont numérotés de $L^d = |V| + |W| + 1$ à $L^f = |V| + 2|W|$. Pour le dépôt numéroté 8 sur la Figure 20, celui-ci est affecté à l'employé w_1 car $8 - (5 + 2) = 1$.

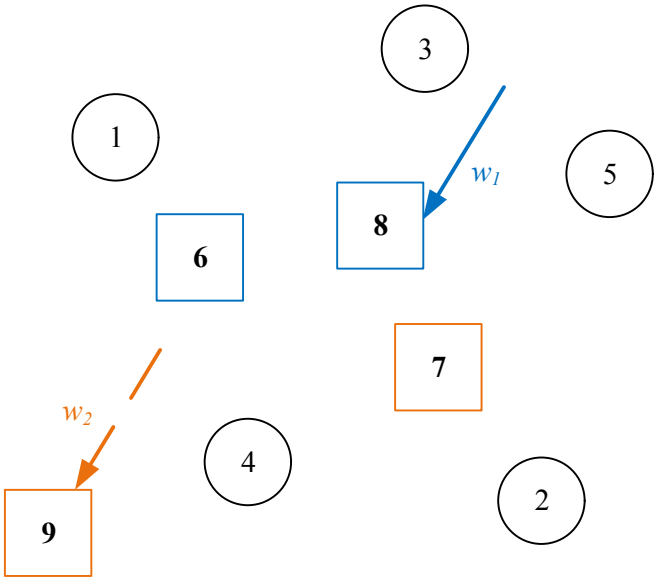


Figure 20 Affectation des dépôts d'arrivée

Le successeur s_i d'un dépôt d'arrivée est par définition le dépôt de départ (DV 23), comme le montre la Figure 17. Le rang p_i d'un dépôt d'arrivée est une valeur comprise entre 1 (car les dépôts au rang $p_i = 0$ sont les dépôts de départ) et $|V| + 1$, dans le cas où toutes les visites sont réalisées par le même employé (DV 24). Le prédécesseur $pred_i$ (DV 25) est soit une visite, soit le dépôt de départ du même employé. Les dates d'arrivée da_i, de début st_i, de fin df_i et de départ dd_i ((DV 26) à (DV 29)) n'impactent pas la solution et peuvent être « relâchées » en définissant des domaines de grandes tailles.

$$a_i = i - (|V| + |W|) \qquad \forall i \in L \qquad \text{(DV 22)}$$
$$s_i = i - |W| \qquad \forall i \in L \qquad \text{(DV 23)}$$
$$p_i = \{1; |V| + 1\} \qquad \forall i \in L \qquad \text{(DV 24)}$$
$$pred_i = \{1; |V|\} \cup \{i - |W|\} \qquad \forall i \in L \qquad \text{(DV 25)}$$
$$da_i = [0\,; H] \qquad \forall i \in L \qquad \text{(DV 26)}$$
$$st_i = [0\,; H] \qquad \forall i \in L \qquad \text{(DV 27)}$$
$$df_i = [0\,; H] \qquad \forall i \in L \qquad \text{(DV 28)}$$
$$dd_i = [0\,; H] \qquad \forall i \in L \qquad \text{(DV 29)}$$
$$dp_i = 0 \qquad \forall i \in L \qquad \text{(DV 30)}$$

Pour les problèmes où les violations des fenêtres de temps sont interdites (notamment pour les instances de VRPTW de (Bredström and Rönnqvist, 2007)), la date d'arrivée da_i au dépôt est importante, car un employé doit être rentré avant la fin de sa fenêtre de temps. Les variables da_i, st_i, df_i, dd_i peuvent être définies en fonction de la fenêtre de temps de l'employé w associé ((DV 26B) à (DV 29B)). Ces contraintes remplacent les contraintes ((DV 26) à (DV 29)).

$$da_i = [TW_w^{inf}\,; TW_w^{sup}] \qquad \forall i \in L | i = L_w \qquad \text{(DV 26B)}$$
$$st_i = [TW_w^{inf}\,; TW_w^{sup}] \qquad \forall i \in L | i = L_w \qquad \text{(DV 27B)}$$
$$df_i = [TW_w^{inf}\,; TW_w^{sup}] \qquad \forall i \in L | i = L_w \qquad \text{(DV 28B)}$$
$$dd_i = [TW_w^{inf}\,; TW_w^{sup}] \qquad \forall i \in L | i = L_w \qquad \text{(DV 29B)}$$

Domaine de la fonction objectif

La valeur d de la fonction objectif est définie par (DV 31).

$$d = [0\,;H] \qquad \text{(DV 31)}$$

4.2.3. Les contraintes pour le modèle PPC

Les contraintes du modèle se décomposent en quatre groupes : les contraintes liées aux tournées, les contraintes liées aux dates, les contraintes liées à la fonction objectif (section 4.2.4) et les contraintes qui sont optionnelles, mais qui permettent de renforcer le modèle PPC (section 4.2.5).

Contraintes liées aux tournées

La contrainte (PPC 1) impose que tous les sommets aient un successeur différent : deux sommets i et j (visites ou dépôts) ne peuvent pas avoir le même successeur ($s_i \neq s_j$). Nts est l'ensemble des sommets du graphe (visites et dépôts compris).

$$s_i \neq s_j \qquad \forall i,j \in Nts^2 \qquad \text{(PPC 1)}$$

Cette contrainte s'écrit habituellement sous la forme de la contrainte (PPC 2) qui utilise une contrainte globale « AllDifferent » (Laurière, 1978; Régin, 1994). Cette contrainte globale est dite parfaite, car elle a été prouvée optimale : la quantité d'information déduite est maximale avec un algorithme de filtrage de complexité minimale. Autrement dit, il n'est possible ni d'obtenir plus de déductions ni de trouver un algorithme de complexité plus faible. Étant globale, cette contrainte est plus efficace que l'écriture deux à deux des contraintes $s_i \neq s_j$.

$$AllDifferent(s_i) \qquad \text{(PPC 2)}$$

Les contraintes (PPC 3) et (PPC 4) garantissent qu'une visite i et son successeur s_i, ou que i et son prédécesseur $pred_i$, sont affectés au même employé.

$$a_i = a_{s_i} \qquad \forall i \in Nts \qquad \text{(PPC 3)}$$

$$a_i = a_{pred_i} \qquad \forall i \in Nts \qquad \text{(PPC 4)}$$

(PPC 5) est la contrainte évitant l'apparition de sous-tour. Chaque sommet possède un rang, et le successeur s_i d'une visite i doit avoir un rang exactement plus grand d'une unité que le rang de i.

$$p_{s_i} = p_i + 1 \qquad \forall i \in V \cup I \qquad \text{(PPC 5)}$$

La contrainte (PPC 6) est optionnelle, et permet d'éviter les sous-tours de taille 1.

$$s_i \neq i \qquad \forall i \in Nts \qquad \text{(PPC 6)}$$

Contraintes liées aux dates

Les dates d'arrivée da_i et de départ dd_i de l'employé sont au plus tôt car aucun critère ne prend en compte la durée des temps d'attente ou leurs positions géographiques. L'employé peut donc attendre où il souhaite. En imposant que les dates da_i et dd_i soient au plus tôt, alors l'arbre de recherche est réduit, car il n'est pas nécessaire de brancher sur ces variables.

La Figure 21 illustre l'arrivée au plus tôt et le départ au plus tôt d'un employé sur une visite. Si l'employé arrive trop tôt par rapport à la date d'ouverture de la fenêtre de temps de la visite, alors celui-ci attend. L'employé a tout intérêt à repartir au plus tôt dès que la visite est

finie car la position où attend l'employé n'a pas d'importance. Par contre la date de début st_i de la visite n'est pas obligatoirement au plus tôt du fait des contraintes du WSRP pénalisant les violations des fenêtres de travail des employés. Travailler avec des dates au plus tôt, quand cela est possible, réduit l'espace de recherche et permet de limiter les branchements dans l'arbre de recherche du solveur PPC. Ainsi les variables da_i et dd_i ne sont pas des variables de décision « libres » et leurs valeurs sont directement induites. En fixant ces variables au plus tôt, celles-ci réduisent la taille de l'arbre de recherche (car elles ne nécessitent pas de branchement), et elles permettent par ailleurs une meilleure propagation car les contraintes utilisant ces variables disposent d'une valeur.

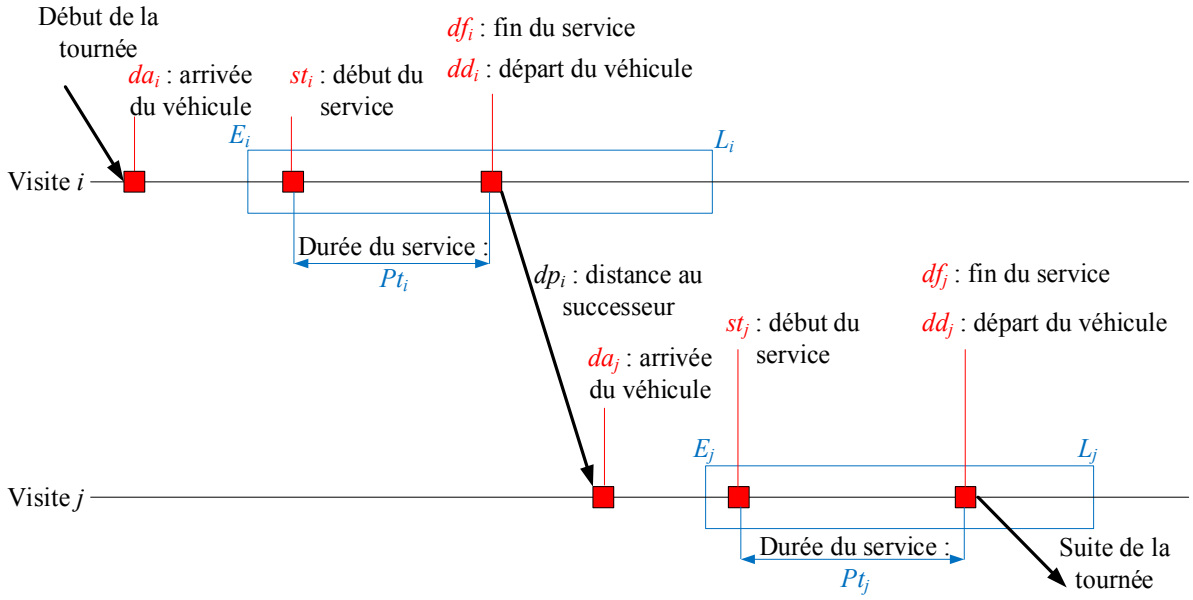


Figure 21 Dates liées aux visites

La contrainte (PPC 7) calcule la date d'arrivée au plus tôt de l'employé sur la visite (ou le dépôt d'arrivée) qui succède à i.

$$da_{s_i} = dd_i + dp_i \qquad \forall i \in V \cup I \qquad \text{(PPC 7)}$$

La date de début st_i d'une visite est imposée par la date d'arrivée de l'employé sur la visite (PPC 8) et la définition de l'intervalle de st_i (DV 7).

$$st_i \geq da_i \qquad \forall i \in V \qquad \text{(PPC 8)}$$

La date de fin df_i d'une visite est égale à sa date de début st_i plus le temps de service (ou processing time) Pt_i de la visite (PPC 9).

$$df_i = st_i + Pt_i \qquad \forall i \in V \qquad \text{(PPC 9)}$$

La date de départ de l'employé de la visite correspond à la date de fin de la visite. L'employé part immédiatement (PPC 10).

$$dd_i = df_i \qquad \forall i \in V \qquad \text{(PPC 10)}$$

4.2.4. Fonction objectif pour le modèle PPC

Les contraintes suivantes permettent d'exprimer la fonction objectif du WSRP.

Un employé a des régions préférées et des régions où il ne souhaite pas travailler. La fonction objectif du WSRP, prend en compte ces préférences et comptabilise des pénalités lorsque l'employé effectue une visite i dans une région non souhaitée. PA_i^w est une donnée du problème telle que $PA_i^w = 1$ si la visite i est dans une région appréciée par l'employé w et $PA_i^w = 0$ sinon. La variable ψ_i permet de savoir si la visite i est affectée à un employé ne souhaitant pas travailler dans la région de i. Si l'employé w est affecté à la visite i, alors $a_i = w$. Si $PA_i^w = 0$ alors $\psi = 1$, si $PA_i^w = 1$ alors $\psi = 0$. La contrainte (PPC 11) met à jour la variable ψ_i.

$PA_i^{a_i} = 0 \Rightarrow \psi_i = 1$ $\quad \forall i \in V$ (PPC 11)
Sinon $\psi_i = 0$

La variable θ_i vaut 1 si la date de début de la visite i est plus tôt que la fenêtre de temps $[TW_{inf}^{w} ; TW_{sup}^{w}]$ de l'employé a_i affecté à cette visite (PPC 12). Si la visite finit après la fenêtre de temps de l'employé affecté, alors $\theta_i = 1$ également (PPC 13). La section 4.1.2 présente en détail la pénalité liée à la variable θ_i.

$st_i \leq TW_{inf}^{a_i} \Rightarrow \theta_i = 1$ $\quad \forall i \in V$ (PPC 12)

$df_i \geq TW_{sup}^{a_i} \Rightarrow \theta_i = 1$ $\quad \forall i \in V$ (PPC 13)

dp_i est la distance entre la visite i (ou un dépôt de départ) et son successeur (PPC 14)

$dp_i = T_{i,s_i}$ $\quad \forall i \in V \cup I$ (PPC 14)

La contrainte (PPC 15) est la somme des dp_i et permet de calculer la distance parcourue par les employés. Les distances entre le dépôt de départ et la première visite, ainsi que la dernière visite et le dépôt d'arrivée sont pris en compte.

$$d = \sum_{\forall i \in V \cup I} dp_i \quad \text{(PPC 15)}$$

(PPC 16) définit la fonction objectif f du WSRP. Grâce à la PPC, il est possible d'écrire directement $PC_i^{a_i}$ ou $\rho_i^{a_i}$, sachant que a_i est une variable de décision.

$$f = \lambda_1 d + \lambda_1 \sum_{i \in V} PC_i^{a_i} + \lambda_2 \sum_{i \in V} \rho_i^{a_i} + \lambda_3 \sum_{i \in V} (\theta_i + \psi_i) \quad \text{(PPC 16)}$$

4.2.5. Contraintes pour renforcer le modèle PPC

Les contraintes introduites dans cette section ne sont pas obligatoires, mais elles permettent d'obtenir un modèle PPC plus efficace. L'efficacité de ces contraintes additionnelles est démontrée, entre autres, dans (Bourreau et al., 2019b, 2020). Ces contraintes peuvent se regrouper en trois catégories : des contraintes sur les prédécesseurs qui sont les « duales » ou les « symétries » des contraintes concernant les successeurs ; des contraintes sur les successeurs en fonction des dates de début des visites ; des contraintes d'interdiction de chevauchement, et enfin des contraintes de liens entre les variables utilisées par la fonction objectif.

Contraintes sur les prédécesseurs

Les contraintes (PPC 17B) à (PPC 19) gèrent les prédécesseurs des visites. La contrainte (PPC 17B) est régulièrement appelée « Channeling » en Choco ou, de manière plus générale, « Inverse » (Hernández, 2008; Cymer, 2012). Cette contrainte définit une bijection entre les

successeurs et les prédécesseurs : si i est le successeur de j, alors j est le prédécesseur de i (voir Figure 22).

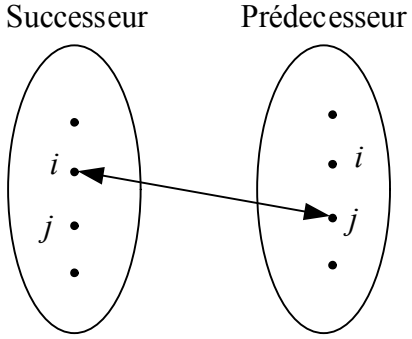


Figure 22 Bijection entre successeurs et prédécesseurs

Afin de faciliter l'implémentation et l'efficacité du modèle, les sommets de dépôts de départ et d'arrivée ont également un successeur et un prédécesseur, formant ainsi des tournées « fermées » (voir Figure 17). L'utilisation de la contrainte globale « Channeling » (PPC 17B) est à privilégier par rapport aux contraintes locales (PPC 17).

$$s_i = j \Leftrightarrow pred_j = i \qquad \forall i \in Nts \qquad \text{(PPC 17)}$$

$$Channeling\ (s_i, pred_i) \qquad \text{(PPC 17B)}$$

La contrainte (PPC 18) met à jour le rang du prédécesseur de i.

$$p_{pred_i} = p_i - 1 \qquad \forall i \in V \cup L \qquad \text{(PPC 18)}$$

Le prédécesseur d'un sommet i (visite, dépôt de départ ou dépôt d'arrivée) ne peut pas être lui-même (PPC 19).

$$pred_i \neq i \qquad \forall i \in Nts \qquad \text{(PPC 19)}$$

Contraintes sur les successeurs en fonction des dates de début

Le prochain ensemble de contraintes permet d'interdire des successeurs d'une visite en fonction des dates de début potentiel des visites. Par exemple, si une visite i finit à la date $ft_i = 30$, si la distance entre i et la visite j est $T_{i,j} = 10$, et si la fenêtre de temps de la date de début st_j de la visite j est $[10\ ;35]$, alors il est évident que la visite j ne peut pas être le successeur de la visite i dans une tournée. j peut donc être supprimée du domaine de s_i.

La première contrainte réduit le domaine de s_i en fonction de la date de début au plus tôt de la visite i, qui est donnée par sa fenêtre de temps. Soit une visite i avec une fenêtre de temps $[E_i\ ;L_i]$ et un temps de service Pt_i, soit une visite j avec une fenêtre de temps $[E_j\ ;L_j]$. La date de début au plus tôt de la visite i est $ES_i = E_i$, la date de fin au plus tôt de la visite i est $EF_i = ES_i + Pt_i$ et la date d'arrivée au plus tôt de l'employé sur la visite j est $EDA_j = EF_i + T_{i,j} = E_i + Pt_i + T_{i,j}$. Si cette date est strictement supérieure à la borne supérieure L_j de la fenêtre de temps de la visite j, alors j ne peut pas être le successeur de i. Ce cas est illustrée par la Figure 23 où les fenêtres de temps des visites sont en vert, le temps de service de i est en rouge, et le temps de transport de la visite i à la visite j est représenté par la flèche. La visite i commence au plus tôt et l'employé ne peut pas atteindre la visite j à temps pour que cette dernière puisse commencer dans sa fenêtre de temps $[E_j\ ;L_j]$. Cette contrainte est exprimée par l'équation (PPC 20).

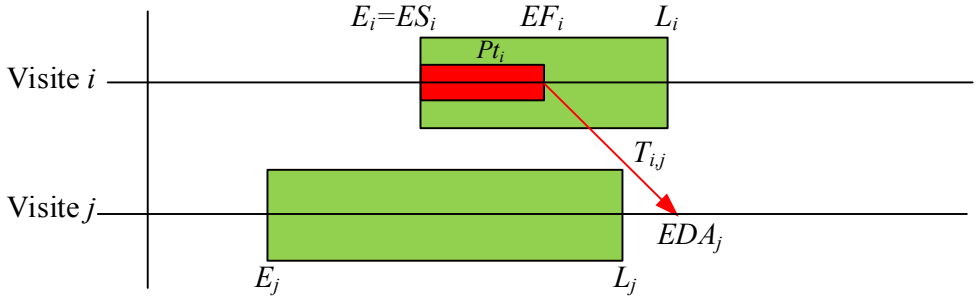


Figure 23 Ordre relatif entre deux visites en fonction des fenêtres de temps

$$s_i \neq j \qquad \forall (i,j) \in V^2 | \; E_i + T_{i,j} + Pt_i \geq L_j \qquad \text{(PPC 20)}$$

La contrainte (PPC 20) peut être généralisée par la contrainte (PPC 21) (Pesant et al., 1998; Rousseau et al., 2013), en prenant en compte la date de début st_i de la visite i. Si la date d'arrivée de l'employé sur la visite j est supérieure à la fenêtre de temps de j ($[E_j ; L_j]$), alors j ne peut pas être le successeur de i (Figure 24).

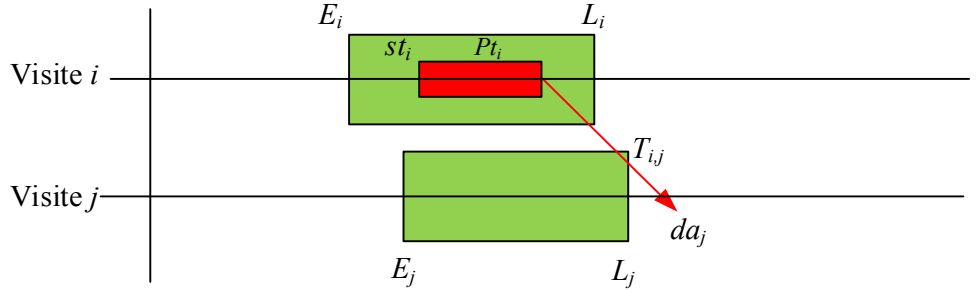


Figure 24 Ordre relatif entre deux visites en fonction de la date de début

$$\text{si } st_i > st_j - \left(T_{i,j} + Pt_i\right) \Rightarrow s_i \neq j \qquad \forall (i,j) \in V^2 \qquad \text{(PPC 21)}$$

Des raisonnements similaires peuvent être appliqués aux prédécesseurs, mais grâce aux contraintes sur les prédécesseurs, si j ne peut pas être le successeur de i, alors i ne peut pas être le prédécesseur de j. Il n'est donc pas nécessaire d'introduire les contraintes similaires avec les prédécesseurs.

Contraintes d'interdiction de chevauchement

Les dernières contraintes qui renforcent le modèle font intervenir la contrainte globale « DiffN » ou « Disjoint » (Beldiceanu and Contejean, 1994). Cette contrainte permet de modéliser très efficacement des problèmes de planification d'opérations (ou activités, ou tâches, suivant la nomenclature utilisée) sous contrainte de ressources. Il a été montré que la contrainte DiffN est particulièrement efficace pour les problèmes d'ordonnancement avec des contraintes de disjonction (Bourreau et al., 2020).

La contrainte DiffN peut être assimilée à un diagramme de Gantt sur lequel des opérations (ou visites suivant la terminologie du problème) doivent être ordonnancées, chaque opération

est caractérisée par sa durée et par la quantité de ressources consommées, formant ainsi un rectangle (Figure 25).

Un rectangle sur le diagramme de Gantt représente donc la consommation de ressources pendant une période, par exemple sur la Figure 25, l'opération v_1 consomme 4 unités de ressources pendant 20 unités de temps. La ressource étant limitée à 4 unités, aucune autre opération ne peut s'effectuer pendant ce laps de temps. De manière générale, deux opérations ne peuvent pas se chevaucher, signifiant que les deux opérations ne peuvent pas s'exécuter simultanément (ou en parallèle), car il n'y a pas assez de ressources disponibles. Ce cas est représenté sur le diagramme par l'interdiction aux rectangles de se superposer. L'objectif de la contrainte DiffN est donc de placer toutes les opérations sans que celles-ci se superposent.

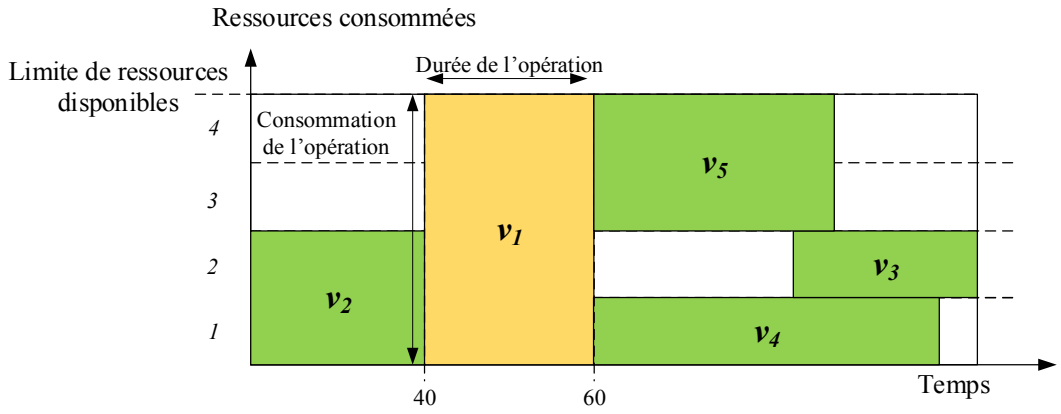


Figure 25 Contrainte DiffN

La contrainte DiffN est très efficace et est applicable au WSRP, car les ressources sont les employés, les opérations sont les visites, et un employé ne peut pas réaliser deux visites en même temps. Les visites ont des durées Pt_i et une consommation de ressources unitaire. Les abscisses sont les dates de débuts et les ordonnées sont les employés, définissant une surface rectangle dans laquelle les visites doivent tenir. Par exemple, sur Figure 26, la visite v_1 est affectée à l'employé w_4 ($a_{v_1} = w_4$) de la date $st_{v_1} = 40$ jusqu'à la date 60. Durant l'intervalle $[40\,;60]$ aucune autre visite ne peut être affectée à l'employé w_4.

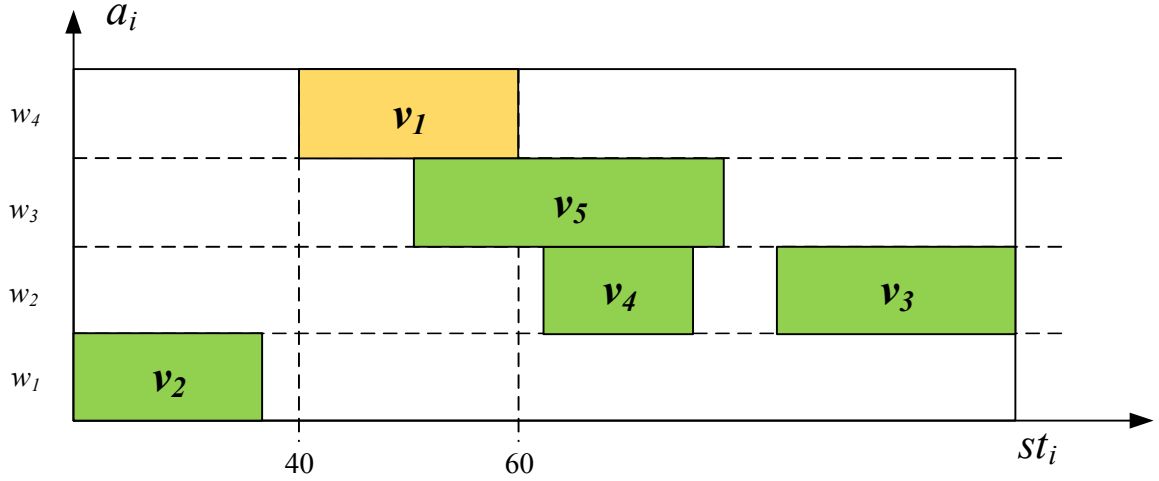


Figure 26 Contrainte DiffN pour les tournées

La force de la contrainte globale « DiffN » est de posséder une vision « globale » des affectations. Au fur et à mesure de la construction de la solution, cette contrainte peut déduire ou propager des informations de manière très efficace. Les contraintes proposées jusqu'à présent pour la modélisation PPC sont majoritairement des contraintes binaires ne faisant intervenir que deux variables, tandis que la contrainte DiffN propage simultanément vers toutes les variables liées aux dates de début (st_i) et les variables d'affectation (a_i).

Pour illustrer la différence entre les deux approches (DiffN et contraintes binaires) et mettre en évidence les déductions possibles par la contrainte DiffN, soit un exemple composé de trois visites i, j et k. Soient $Pt_i = 15$, $Pt_j = 15$ et $Pt_z = 15$ leurs processing times respectifs, et leurs fenêtres de temps sont $[30\,;50]$ (les domaines des dates ont été réduits par propagation et filtrage lors des branchements précédents). Soient deux employés w_1 et w_2.

Si les dates des visites sont comparées deux à deux, alors il est possible de planifier la visite i et la visite j dans l'intervalle de temps $[30\,;50]$ avec des employés différents. Il en va de même pour planifier les visites i et k, et j et k. Par exemple, quelle que soit la date de début de st_i sur l'employé w_1, alors la visite j peut être planifiée à n'importe quelle date pour l'employé w_2. Cette « vision binaire » n'est pas suffisante pour détecter que les trois visites ne peuvent pas être planifiées dans cet intervalle. Les trois visites ayant une durée de service de 15 unités de temps, il est évident qu'il n'existe pas de solution telle qu'elles soient toutes les trois planifiées entre 30 et 50, car seuls deux employés sont disponibles.

Si (et donc en cas de branchement) la visite i débute à la date $st_i = 30$ avec l'employé w_1, alors $ft_i = 45$, et il n'est pas possible d'utiliser cet employé pour une autre visite. Si une autre visite (j par exemple) est affectée au même employé, alors débute au plus tôt à $st_j = 45$, et finit donc à $ft_j = 60$, ce qui est après la fenêtre de temps $[30\,;50]$. L'employé w_2 doit donc réaliser les deux visites, ce qui est impossible.

En adoptant une vision « globale » prenant simultanément en compte les trois visites, il est facile de constater qu'il n'y a pas de solution possible. La contrainte DiffN revient à essayer de planifier les trois visites dans un intervalle $[30\,;50]$ sur un diagramme de Gantt (Figure 27). Il est visuellement vérifiable que cela est impossible. La contrainte globale DiffN a une vision globale qui permet de détecter qu'il n'existe pas de solution, et donc de couper la branche de l'arbre de recherche.

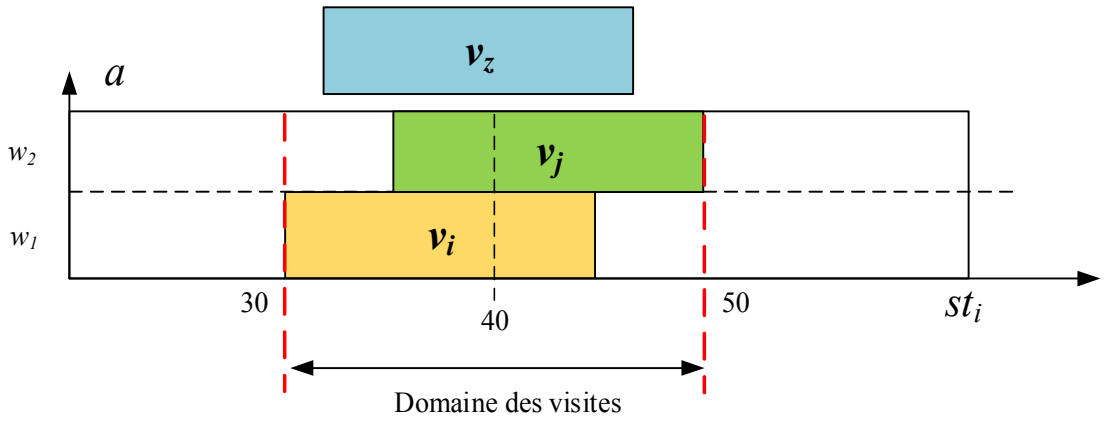


Figure 27 Détection d'un cas impossible par la contrainte DiffN

Si la contrainte DiffN n'est pas utilisée alors le solveur branche $st_i = 30$, puis $st_j = 30$ avant de se rendre compte que ce n'est pas possible car st_z a un domaine vide. Il remonte alors dans l'arbre pour brancher $st_j = 31$, puis une fois que tous les st_j sont testés, le solveur branche sur $st_i = 31$, etc, voir la Figure 28.

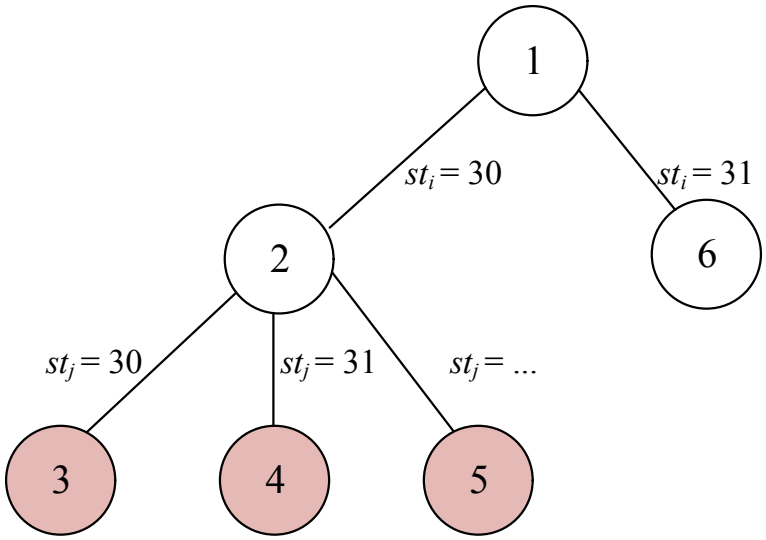


Figure 28 Branchements ne permettant pas d'obtenir de solution

Le premier argument de la contrainte DiffN (PPC 22) est l'ensemble des variables de l'axe des abscisses : les dates de début st_i. Le second argument est l'ensemble des variables de l'axe des ordonnées : les affectations a_i. Le troisième argument est la durée des visites : leur processing time Pt_i. Et le quatrième argument est la consommation en ressources des visites : la consommation est unitaire car une visite n'est réalisée que par un seul employé. Une unique contrainte DiffN suffit donc pour tout le modèle.

$$Diff N(st_i, a_i, Pt_i, ones) \qquad \text{(PPC 22)}$$

Dans le cas présent, les Pt_i et $ones$ sont des données du problème, mais dans le cas général d'une DiffN, ils peuvent être des variables.

Contraintes liant les variables de la fonction objectif

Les contraintes (PPC 12) et (PPC 13) (concernant la violation des fenêtres de temps des employés – ces contraintes sont rappelées ci-dessous) peuvent également être modélisées plus efficacement avec une contrainte DiffN.

$$st_i \leq TW_{inf}^{a_i} \Rightarrow \theta_i = 1 \qquad \forall i \in V \qquad \text{(PPC 12)}$$

$$df_i \geq TW_{sup}^{a_i} \Rightarrow \theta_i = 1 \qquad \forall i \in V \qquad \text{(PPC 13)}$$

La fonction objectif prend en compte les variables θ_i, mais ces contraintes sont des implications et donc aucune déduction ne peut être faite pour les variables st_i à partir des variables θ_i. Des déductions peuvent être uniquement réalisées à partir des variables st_i en direction des variables θ_i. Par exemple, si le solveur sait que pour améliorer le coût de la solution courante, il faut que $\sum_{i \in V} \theta_i < 3$, alors il est obligé de fixer toutes les opérations et leur date avant de savoir si son ordonnancement permet de respecter la contrainte précédente.

Soit l'exemple de la Figure 29 où les quatre employés ont la même fenêtre de temps $[40\,;80]$. Les visites v_1, v_5, v_7 et v_2 sont déjà planifiées, et les visites v_3, v_6, v_4 et v_8 doivent être ordonnancées. Il est visuellement évident qu'une seule visite peut être placée dans la fenêtre de temps des employés, et qu'au minimum trois visites vont entraîner une pénalité dans la fonction objectif. Une contrainte DiffN légèrement adaptée peut permettre de détecter ces cas.

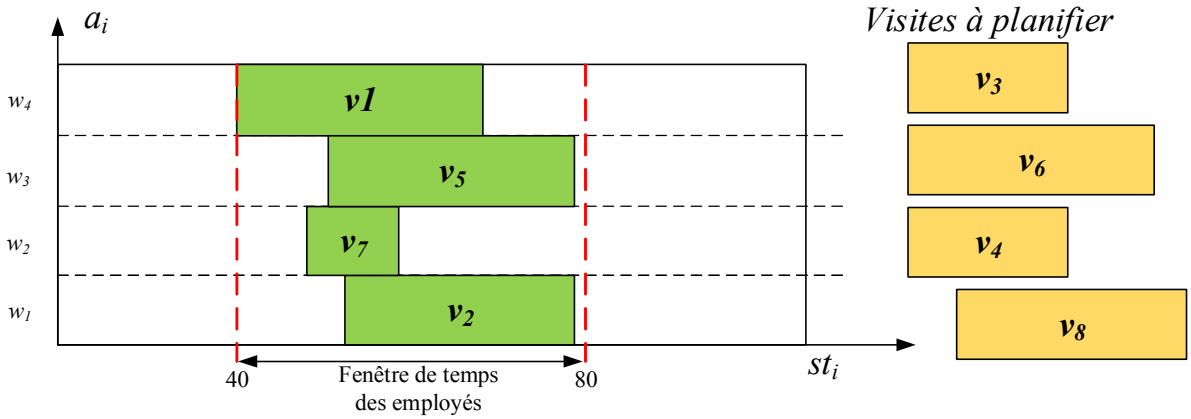


Figure 29 Détection du nombre minimal de violations

Afin d'aider le solveur, il est intéressant d'avoir des contraintes permettant des équivalences pour les contraintes (PPC 12) et (PPC 13). L'idée de la nouvelle contrainte est de généraliser la contrainte (PPC 22), où l'axe des abscisses ne correspond plus aux affectations uniquement, mais simultanément aux affectations et à la pénalité de violations des fenêtres de temps des employés. Une nouvelle variable $A\theta_i$ $(\forall i \in V)$ est créée, avec $A\theta_i = 2|W|(a_i - 1) + \theta_i$ (où $|W|$ est le nombre d'employés). Si $A\theta_i = 0$, par exemple, alors la visite i est affectée à l'employé 1 et en plus $\theta_i = 0$; si $A\theta_i = 1$, alors i est affectée à l'employé 1, et $\theta_i = 1$.

Lors de la planification d'une visite, le solveur connaît donc simultanément la valeur de deux variables. De plus, il est possible d'aider le solveur à détecter les cas où $\theta_i = 1$. En effet, si la visite i commence ou finit en dehors de la fenêtre de temps de l'employé affecté, alors θ_i doit valoir 1 et il est impossible que $\theta_i = 0$. Afin d'interdire $\theta_i = 0$, deux visites fictives sont créées (en orange sur la Figure 30), et sont situées en dehors de la fenêtre de temps $[TW^w_{inf}\,;\, TW^w_{sup}]$ de l'employé. La contrainte DiffN interdit aux visites de se chevaucher, et la visite i (en bleu sur la Figure 30) ne peut donc pas être positionnée en dehors de la fenêtre de temps de l'employé lorsque $\theta_i = 0$.

La Figure 30 illustre la contrainte DiffN. Les deux zones en orange sont les zones interdites (qui sont modélisées par des visites fictives) pour la visite i avec $\theta_i = 0$, car ces deux zones sont situées hors de la fenêtre de temps de l'employé 1. Afin de faciliter la compréhension du schéma, la Figure 30 présente la nouvelle contrainte utilisant la contrainte globale DiffN pour une visite pour un employé. Afin d'obtenir un modèle efficace, toutes les visites et les employés doivent être considérés ensemble dans une même contrainte DiffN.

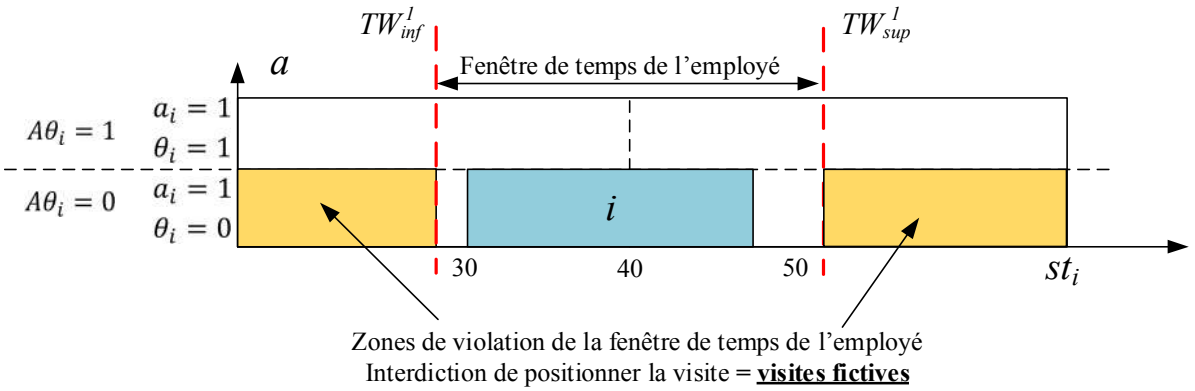


Figure 30 DiffN avec violation des fenêtres de temps

La contrainte DiffN nécessite donc de créer deux nouvelles visites fictives pour chaque employé $w \in W$: la première visite z_1 débute à la date $st_{z_1} = 0$ et sa durée est $Pt_{z_1} = TW^w_{inf}$; la seconde visite z_2 débute à la date $st_{z_2} = TW^w_{sup}$ et sa durée est $Pt_{z_2} = H - TW^w_{sup}$ (où H est la taille de l'horizon). Deux nouveaux ensembles de variables sont définis :

$$st'_j = \{st_i | \forall i \in V\} \cup \{st^w_{z_1} = 0 | \forall w \in W\} \cup \{st^w_{z_2} = TW^w_{sup} | \forall w \in W\}$$

Et

$$Pt'_j = \{Pt_i | \forall i \in V\} \cup \{Pt^w_{z_1} = TW^w_{inf} | \forall w \in W\} \cup \{Pt^w_{z_2} = H - TW^w_{sup} | \forall w \in W\}$$

La DiffN concernant simultanément l'affectation et les violations des fenêtres de temps des employés est définie par la contrainte (PPC 23).

$$Diffn(st_j', A\theta_j, Pt_j', ones) \qquad \text{(PPC 23)}$$

Contraintes liant les violations des fenêtres de temps et les violations de régions

La dernière contrainte pour renforcer le modèle a pour objectif de créer un lien entre la variable θ_i et la variable ψ_i d'une visite i. Ces variables concernent respectivement la violation de la fenêtre de temps de l'employé affecté et la violation de sa région préférée. La construction de cette contrainte est similaire à la contrainte (PPC 23).

Une nouvelle variable $ThPs_i$ est introduite pour chaque visite i et est définie par $ThPs_i = 2\psi_i + \theta_i$. Cette variable permet de connaître simultanément les variables ψ_i et θ_i : $ThPs_i = 0$ si et seulement si $\psi_i = 0$ et $\theta_i = 0$; $ThPs_i = 1$ si et seulement si $\psi_i = 0$ et $\theta_i = 1$; etc.

Cette variable est utilisée dans une contrainte DiffN où les abscisses ne sont plus le temps, mais l'affectation a_i de la visite i. Lorsque $PA^w_i = 0$ (qui est une donnée du problème), alors $\psi_i = 1$. Cette contrainte est imposée en interdisant à la contrainte DiffN de positionner la visite i dans la zone où $\psi_i = 0$ pour l'employé w. Cette interdiction se traduit par la présence d'une visite déjà positionnée. Et inversement, si $PA^w_i = 1$, alors $\psi_i = 0$, et il est impossible de positionner la visite i dans la zone où $\psi_i = 1$ pour l'employé w.

La Figure 31 est une représentation de la contrainte DiffN pour une visite i avec, en abscisse, la variable a_i (l'employé affecté à la visite i) et, en ordonnée, la variable $ThPs_i$. Les rectangles en orange représentent les endroits où il est impossible de positionner la visite i à

cause des préférences des employés, et sont donc des visites fictives. Par exemple, l'employé $w = 1$ (donc $a_i = 1$) apprécie la zone géographique où est située la visite i ($PA_i^w = 1$), donc il est impossible de positionner la visite i sur la partie supérieure du diagramme de Gantt pour l'employé 1, car cette zone représente $\psi_i = 1$. La même remarque est vérifiée pour l'employé $w = 3$. Pour les employés $w = 2$ et $w = 4$, c'est le cas inverse qui se produit : $PA_i^w = 0$, donc si $w = 2$ ou $w = 4$ réalise la visite, alors obligatoirement $\psi_i = 1$. Toutes les visites sont de durée unitaire, mais leur consommation en ressources (c'est-à-dire leur hauteur) varie. La visite i ne consomme qu'une unique ressource, tandis que les visites fictives consomment deux ressources (pour interdire simultanément $\theta_i = 0$ et $\theta_i = 1$, pour un ψ_i donné).

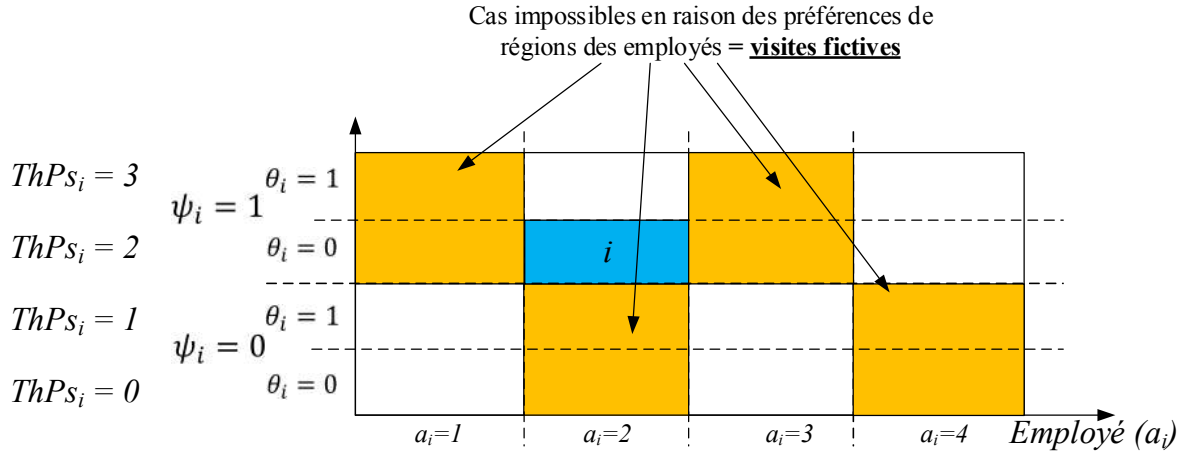


Figure 31 DiffN avec violation des fenêtres de temps et violation des régions

Il y a donc une contrainte DiffN par visite (PPC 24). st_i'' est l'ensemble des dates de début des visites. Soient les ensembles suivants :

- $a_j'' = a_i \cup \{a^w = w | \forall w \in W\}$, l'affectation des visites (fictives comprises).
- $ThPs_j = \{ThPs_i\} \cup \{ThPs^w = 2 - 2.PA_i^w | \forall w \in W\}$, la variable $ThPs_j$ pour chaque visite.
- $Hv_j = \{Hv_i = 1\} \cup \{Hv^w = 2 | \forall w \in W\}$, la consommation en ressource des visites.

La contrainte s'écrit alors :

$$DiffN(st_j'', ThPs_j, ones, Hv_j) \qquad \forall i \in V \qquad \text{(PPC 24)}$$

Les modèles PLNE et PPC ont été implémentés en utilisant, respectivement, les solveurs CPLEX et CHOCO. Les résultats sont donnés dans la section 6.

5. Génération de colonnes pour le WSRP (Garaix et al., 2018a, 2018c)

Le modèle PLNE du WSRP est basé sur un nombre de variables binaires très élevé, et de nombreuses contraintes font intervenir un « BigM ». Ces deux éléments sont connus pour ne pas favoriser la résolution des problèmes d'ordonnancement et/ou de tournées de véhicules par résolution directe en PLNE. Les modèles PPC permettent généralement d'obtenir une première solution en un temps très court, mais requièrent ensuite beaucoup d'efforts pour trouver la solution optimale et prouver que celle-ci l'est. Le WSRP n'échappe pas à la règle, et est difficilement résolu en PLNE (Castillo-Salazar et al., 2016) ou en PPC.

La génération de colonnes est un paradigme régulièrement utilisé pour des problèmes d'ordonnancement et de tournées de véhicules. Elle permet théoriquement d'obtenir des solutions optimales, et requiert généralement un temps de calcul beaucoup plus faible qu'une résolution directe du modèle PLNE.

La méthode de résolution, proposée dans ce chapitre, est basée sur une procédure de Génération de Golonnes (GC), où :

- Le problème maître (section 5.3) correspond soit à :
  - un Set Covering Problem (section 5.3.1) ;
  - un Set Partitioning Problem (section 5.3.2).
- La résolution du sous-problème (ou problème esclave) se fait soit par :
  - PLNE (section 5.4) ;
  - programmation dynamique (section 5.5).
- La solution en nombres entiers est obtenue (section 5.6) soit par :
  - résolution du problème maître en nombres entiers (section 5.6.1).
  - un Branch-and-Price (section 5.6.2).

5.1. État de l'art concernant la génération de colonnes

La Génération de Colonnes (GC) est une méthode très efficace pour résoudre des problèmes de tournées de véhicules et/ou d'ordonnancement. De nombreux articles issus de la littérature l'utilisent. Cette section présente succinctement quelques publications récentes qui concernent des problèmes d'ordonnancement et/ou transport et d'affectation, résolus par génération de colonnes.

(Gérard et al., 2016) abordent un Tour Scheduling Problem with a Multi-skill Heterogeneous Workforce. Ils prennent en compte des contraintes qui incluent (mais qui ne sont pas limitées) les compétences des travailleurs, les périodes de temps de travail, les pauses-repas et les pauses. Le problème combine simultanément l'ordonnancement des jours de congé, l'ordonnancement des quarts de travail, l'affectation des équipes, l'affectation des tâches, la planification de pauses, ainsi que celle des pauses-repas. Ils proposent quatre méthodes : un modèle PLNE, une approche Branch-and-Price avec le problème esclave résolu en programmation dynamique, une diving heuristic et une heuristique gloutonne.

(Gaur et Singh, 2017) résolvent le Cumulative Vehicles Routing Problem à l'aide d'une génération de colonnes. Le Cumulative Vehicles Routing Problem est un modèle simplifié de la consommation de carburant dans les problèmes de tournées de véhicules. Le problème maître correspond à un problème de couverture par ensemble (Set Covering Problem) et le problème esclave est résolu heuristiquement avec une approche de type programmation dynamique.

(Janacek et al., 2017) introduisent une génération de colonnes qui s'applique à l'optimisation des horaires périodiques des équipes des opérateurs ferroviaires. L'objectif de leur problème est d'affecter les équipes appropriées aux trains circulant sur un réseau, parmi l'ensemble des équipes qui sont disponibles à chaque dépôt.

(Gomes et al., 2017) promeuvent une génération de colonnes couplée à un Variable Neighborhood Search (VNS) pour traiter un Nurse Rostering Problem (NRP), où l'objectif est d'affecter des infirmières à des quarts de travail. Ils proposent également deux heuristiques pour obtenir des solutions entières réalisables dans un temps de calcul faible.

(Farham et al., 2018) proposent une génération de colonnes pour le Location-Routing Problem with Time Windows (LRPTW) où le problème esclave est résolu par programmation dynamique. Le LRPTW consiste à ouvrir des dépôts puis à effectuer des livraisons à des clients à partir de ceux-ci. La résolution du LRPTW comprend la sélection d'un ensemble de dépôts où sont stockés les produits à livrer, l'affectation de clients aux dépôts et la détermination des itinéraires des véhicules effectuant les livraisons. L'objectif du LRPTW est de minimiser la somme des coûts d'ouverture des dépôts, l'utilisation des véhicules et les frais de déplacement.

Ces nombreux articles démontrent l'efficacité des générations de colonnes à résoudre des problèmes imbriquant problème d'ordonnancement, problème d'affectation et problème de tournées de véhicules. De plus, les articles ou ouvrages de (Desaulniers et al., 2005, 2001; Desaulniers, 2010) sont des références internationales concernant la génération de colonnes et (Feillet, 2010) propose un tutoriel sur la génération de colonnes basée sur le Vehicle Routing Problem with Time Window (VRPTW).

Ces publications nous poussent à considérer que le WSRP peut se prêter à une résolution par génération de colonnes couplée à des méthodes approchées pour la résolution du problème esclave et également pour l'obtention d'une solution en nombres entiers.

5.2. Définition d'une colonne pour le WSRP

Une colonne k définie une tournée pour un employé donné, et est caractérisée par une donnée E_k et deux variables CT_k et Δ_k^i où :

- $E_k = w$ est le numéro de l'employé réalisant la tournée.
- CT_k est le coût de la tournée.
- Δ_k^i est un vecteur avec $\Delta_k^i = 1$ si la visite i est réalisée au cours de la tournée, et $\Delta_k^i = 0$ sinon.

Le coût CT_k de la colonne k (ou par abus de langage, de la tournée k) est positif et dépend à la fois de l'employé réalisant la tournée, des visites qui lui sont affectées et des dates de début des visites :

$$CT_k = \lambda_1 \sum_{i \in V \cup I_w} \sum_{j \in V \cup L_w} \left(T_{i,j} + p_j^w \right) x_{i,j}^w + \lambda_2 \sum_{j \in V} \sum_{i \in V \cup I_w} \left(3 - \rho_j^w\right) x_{i,j}^w + \lambda_3 \sum_{i \in V} \left(\psi_i^w + \theta_i^w \right)$$

5.3. Le problème maître

Soit $\omega(w)$ l'ensemble des colonnes de l'employé w. Chacune des colonnes représente une tournée pour w. Ω est l'ensemble des colonnes pour l'ensemble des employés : $\Omega = \cup_{w \in W} \omega(w)$. Soit la variable binaire y_k qui détermine si la colonne k est sélectionnée par le problème maître.

Le problème maître peut être formulé comme un Set Partitioning Problem (section 5.3.1) ou comme un Set Covering Problem (section 5.3.2).

5.3.1. Formulation linéaire du problème maître par un Set Partitioning Problem

Le problème maître s'apparente à un Set Partitioning Problem (Garfinkel and Nemhauser, 1969; Karp, 1972; Balinski, 2010).

$$f_{master} = Min \sum_{k \in \Omega} CT_k \, y_k$$

$$s.t.$$

$$\sum_{k \in \Omega} \Delta_k^i y_k = 1 \qquad \forall i \in V \qquad (1)$$

$$\sum_{k \in \Omega | E_k = w} \Delta_k^i \, y_k \leq 1 \qquad \forall w \in W \qquad (2)$$

La fonction objectif du problème maître minimise la somme des coûts des tournées. La contrainte (1) garantit que la visite i est réalisée une et une seule fois. La contrainte (2) certifie que plusieurs colonnes affectées au même employé ne peuvent pas être sélectionnées par le problème maître. Il est important de garder en mémoire que l'employé est une donnée d'entrée du problème esclave, et il n'y a donc pas de problème d'affectation des tournées aux employés. En effet, le coût d'une même tournée est différent suivant l'employé qui la réalise, et certains employés ne peuvent pas réaliser la tournée car ils sont interdits d'exécuter la visite i d'après leur contrat.

La valeur duale de la visite i associée à la contrainte (1) est α_i ; et β_w est la valeur duale de l'employé w associée à la contrainte (2).

5.3.2. Formulation linéaire du problème maître par un Set Covering Problem

(Barnhart et al., 1998) démontrent que pour de nombreux problèmes de tournées de véhicules et d'ordonnancement, le problème maître peut être formulé soit comme un Set Partitioning Problem (SPP), soit comme un Set Covering Problem (SCP) (Karp, 1972; Chvátal, 1979).

Pour le WSRP, une visite ne doit être visitée que par un unique employé. Par conséquent, la formulation la plus « naturelle » du problème maître est la formulation par SPP introduite à la section 5.3.1 où chaque visite est précisément réalisée qu'une seule fois. La formulation du problème maître par SCP est une alternative où une visite peut être réalisée plusieurs fois par des employés différents. En d'autres termes, une visite peut être comprise plusieurs fois dans le sous-ensemble de colonnes sélectionné par le SCP ; tandis qu'elle n'est présente qu'une unique fois dans le sous-ensemble de colonnes sélectionné par le SSP.

Puisque supprimer une visite d'une tournée résulte en une nouvelle tournée de coût inférieur, la solution optimale du Set Covering Problem est une solution optimale du Set Partitioning Problem, comme le souligne (Barnhart et al., 1998).

(Barnhart et al., 1998) promeuvent l'utilisation du SCP car sa relaxation linéaire est beaucoup plus stable et donc facile à résoudre que le SPP, et car il trivial de construire une solution entière réalisable pour le SPP à partir d'une solution du SCP.

Le problème maître sous forme de Set Covering Problem est donné par la formulation suivante :

$$f_{master} = Min \sum_{k \in \Omega} CT_k \, y_k$$

$s.t.$

$$\sum_{\boldsymbol{k \in \Omega}} \boldsymbol{\Delta_k^i y_k \leq 1} \qquad \boldsymbol{\forall i \in V} \qquad \boldsymbol{(1)}$$

$$\sum_{k \in \Omega | E_k = w} \Delta_k^i \, y_k \leq 1 \qquad \forall w \in W \qquad (2)$$

La différence entre le SCP et le SPP est donc uniquement la contrainte (1). Lorsque le problème maître correspond à un Set Covering Problem, une visite peut être couverte par plusieurs tournées. Toutefois, très peu de visites sont réalisées par plusieurs tournées, car lorsqu'une visite est réalisée plusieurs fois, alors le coût de son exécution est compris plusieurs fois dans le coût de la solution finale.

En pratique, peu de visites sont donc couvertes plusieurs fois. Néanmoins, si tel est le cas, une approche gloutonne triviale permet de supprimer la visite (ou les visites) présente plusieurs fois dans les tournées. Cette approche gloutonne consiste à calculer le coût des tournées sans la visite concernée, et à conserver la solution avec le coût le plus faible.

Le choix de la formulation (SPP ou SCP) du problème maître n'intervient pas sur la manière de formuler le problème esclave. Comme pour la formulation en SPP, la valeur duale de la visite i associée à la contrainte (1) est α_i ; et β_w est la valeur duale de l'employé w associée à la contrainte (2).

5.3.3. Initialisation du problème maître

Un des objectifs du WSRP est de maximiser le nombre de visites réalisées, chaque visite non réalisée est pénalisée dans la fonction objectif par un coût λ_4. Cet objectif est traduit de façon indirecte dans le problème maître lors de l'initialisation de la GC.

La génération de colonnes est initialisée avec un ensemble de colonnes, formant une matrice identité notée MI, avec autant de colonnes que de visites. Chaque colonne est associée à un employé fictif dont la tournée ne comporte qu'une seule visite et est donc du type : dépôt – visite i – dépôt ($\exists! \, i \in V \mid \Delta_{k \in MI}^i = 1$). La contrainte (2) du problème maître est relâchée pour l'employé fictif. Le coût de chaque colonne est donc $CT_{k \in MI} = \lambda_4$. Cette initialisation du problème maître garantit la faisabilité de la contrainte (1).

Une solution du problème maître utilisant une colonne k appartenant à la matrice identité ($k \in MI$) signifie qu'il existe une visite $i \in V$ qui n'est satisfaite par aucun employé $w \in W$.

5.4. Problème esclave

Un problème esclave est associé à un employé w donné, et a pour objectif de générer de nouvelles colonnes pour le problème maître. Il y a donc autant de problèmes esclaves que d'employés. Une nouvelle colonne k est ajoutée dans le problème maître si elle a un coût réduit CR_k négatif. Si, après la résolution de tous les problèmes esclaves, aucune nouvelle colonne n'est ajoutée dans le problème maître, alors la solution optimale est atteinte. Le problème esclave dépend de l'employé w associé, de la valeur duale β_w associée à w, et des valeurs duales α_i associées à chaque visite i.

CR_k est le coût réduit d'un problème esclave, il prend en compte le coût CT_k de la tournée (qui est une inconnue) moins la somme des valeurs des variables duales des visites réalisées et la valeur (qui est connue) de la duale β_w de l'employé.

$$CR_k = CT_k - \sum_{j \in V} \sum_{i \in V \cup I_w} \alpha_j x_{ij}^w - \beta_w$$

La fonction objectif du problème esclave est :

$$f = min(CR_k)$$

Les contraintes du problème esclave sont très proches de celles utilisées pour la formulation PLNE du WSRP (section 4.1.3). La différence réside dans le fait que les variables du problème esclave ne sont pas indexées sur l'employé w, puisque celui-ci est une donnée du problème. La contrainte (8) du modèle PLNE (de la section 4.1) n'est pas conservée car elle concerne les visites non réalisées. Une contrainte du problème esclave porte le même numéro que la contrainte du PLNE correspondant. Les contraintes sont les suivantes (et pour plus d'explications, le lecteur peut se référer à la section 4.1.3) :

$$\sum_{j \in V \cup L_w} x_{i,j} \leq 1 \qquad \forall i \in V \qquad (1)$$

$$\sum_{j \in V \cup I_w} x_{i,j} \leq 1 \qquad \forall j \in V \qquad (2)$$

$$\sum_{j \in V \cup L_w} x_{i,j} = \sum_{j \in V \cup I_w} x_{j,i} \qquad \forall i \in V \qquad (3)$$

$$\psi_i + M.Pa_i^w \geq \sum_{j \in V \cup I_w} x_{j,i} \qquad \forall i \in V \qquad (4)$$

$$M.\theta_i \geq TW_{inf}^w - st_i + \left(\sum_{j \in V \cup I_w} x_{j,i} - 1 \right).M \qquad \forall i \in V \qquad (5)$$

$$M.\theta_i \geq st_i + Pt_i - TW_{sup}^w + \left(\sum_{j \in V \cup I_w} x_{j,i} - 1 \right).M \qquad \forall i \in V \qquad (6)$$

$$\sum_{j \in V \cup I_w} x_{j,i} \leq CoTr_i^w \qquad \forall i \in V \qquad (7)$$

$$T_{i,j} + st_i + Pt_i \leq st_j + (x_{i,j} - 1).M \qquad \forall (i,j) \in V^2 \qquad (9)$$

$$st_i \geq E_i \qquad \forall i \in V \qquad (10)$$

$$st_i \leq L_i \qquad \forall i \in V \qquad (11)$$

Le problème esclave peut être résolu par différentes approches, mais seulement une résolution exacte permet de garantir l'optimalité de la solution obtenue par le problème maître. Une résolution exacte par PLNE est souvent coûteuse en temps de calcul et des résolutions approchées ou heuristiques sont souvent favorisées. L'approche utilisée dans ce manuscrit est une résolution du problème esclave par programmation dynamique. La nouveauté de la procédure réside dans la prise en compte de la qualité de service pour les employés.

La résolution d'un problème esclave correspond donc à la recherche d'un nouveau chemin pour un employé donné (w est fixé). Après chaque résolution d'un problème esclave avec obtention d'une nouvelle colonne, le problème maître est de nouveau résolu. Les problèmes esclaves sont résolus cycliquement : le premier problème résolu est pour l'employé $w = 1$, à l'itération suivante le problème esclave est résolu pour l'employé $w = 2$, et ainsi de suite. Lorsque $w = |W|$, le prochain problème résolu est pour $w = 1$.

5.5. Programmation dynamique pour la résolution du problème esclave (Garaix et al., 2018c)

Une nouvelle procédure de programmation dynamique est utilisée pour résoudre les problèmes esclaves. Elle est basée sur celle introduite pour l'Elementary Shortest Path Problem with Resource Constraints (ESPPRC) par (Feillet et al., 2004). Cette procédure permet de prendre en compte le critère de qualité de service des employés, et est une généralisation de la fonction d'évaluation d'une tournée du WSRP présentée dans la section 3.4.

Le problème esclave, associé à l'employé w, consiste à calculer un chemin réalisable avec un coût réduit négatif, qui part du dépôt initial I_w de l'employé, et retourne au dépôt d'arrivée L_w de w. Des tournées partielles sont créées et stockées sur chaque nœud grâce à un label. Un nœud contient un ensemble de labels noté L^{node}. Chaque label L_i d'un nœud i est ensuite propagé vers d'autres nœuds (la liste de ces nœuds est notée lsa) afin d'obtenir un nouveau chemin partiel. La condition élémentaire du chemin garantit qu'un nœud ne peut être visité qu'une unique fois dans la tournée. L'ESPPRC est connu pour être NP-difficile (Dror, 1994).

5.5.1. Définition du label

Pour le problème esclave associé à l'employé $w \in W$, un label L représente une solution partielle (aussi appelée « solution », par abus de langage), modélisée par un chemin dans le graphe. Ce chemin débute au dépôt I_w de w et finit au dépôt d'arrivée L_w. Il est composé, entre ces deux dépôts, des visites que w effectue, et peut être schématisé par $I_w - k - l - L_w$ (avec

$(k, l) \in V^2$). Ajouter une visite m à la tournée de w se traduit dans le chemin par son insertion en fin de tournée, juste avant le retour au dépôt. Le nouveau chemin est donc $I_w - k - l - m - L_w$. L^j est l'ensemble des labels contenus dans le nœud j, et L_i^j est le i° label du nœud j.

Un label L doit inclure toutes les informations suivantes :

- CT : le coût de la tournée (partielle). Il est important de noter que le coût du label comprend le coût de retour de l'employé au dépôt.
- RC : le coût réduit de la tournée (partielle).
- Δ_i : un vecteur avec $\Delta_i = 1$ si la visite i est réalisée au cours de la tournée, et $\Delta_i = 0$ sinon.
- a_j : la date d'arrivée de l'employé sur la visite j.
- st_j : la date de début de la visite j.
- d_j : la date de départ de la visite j.
- p : booléen qui vaut 1 si le label a été propagé, et 0 sinon.
- lsa : liste des sommets atteignables.

Il est important de noter qu'ici, la date de début des visites n'est pas conservée contrairement à la procédure introduite pour l'évaluation d'une tournée de coût minimal (section 4.2.5). En effet, dans le problème maître, il n'est pas nécessaire de connaître ces dates à partir du moment où il est garanti que la tournée est réalisable.

5.5.2. Label initial

Le label initial du problème esclave, associé à l'employé $w \in W$, est situé sur le nœud de départ I_w de w. Ce label initial correspond à une tournée commençant au dépôt initial I_w de w et finissant au dépôt d'arrivée L_w de w. Les caractéristiques du label initial L sont les suivantes :

- $CT = T_{I_w, L_w}$: le coût de la tournée (partielle).
- $RC = CT - \beta_w$ (où β_w est la valeur de la variable duale associée à w).
- $\Delta_i = 0, \forall i \in V$: aucune visite n'est encore effectuée.
- $a_j = 0$.
- $st_j = 0$, la date de début est nulle même si la borne inférieure de la fenêtre de travail ($[TW_{inf}^w ; TW_{sup}^w]$) de w est strictement positive ($TW_{inf}^w > 0$) car un employé peut violer sa fenêtre de temps.
- $d_j = 0$.
- $p = 0$.
- $lsa = \{i \in V \mid CoTr_i^w = 1\}$: la liste initiale des sommets atteignables contient tous les sommets que l'employé w peut visiter ($CoTr_i^w = 1$).

5.5.3. Algorithme de principe

L'Algorithme 2 est l'algorithme de principe du problème esclave pour le WSRP. Λ est la file des nœuds contenant les labels qui n'ont pas été propagés. L'algorithme débute par l'initialisation d'un label (cette initialisation est détaillée en section 5.5.2), situé sur le nœud I_w (le dépôt de départ de w). I_w est donc inséré dans Λ. Ce label est ensuite propagé vers les autres sommets, accessibles par l'employé w. L'algorithme se termine lorsqu'il n'y a plus aucun label non propagé sur les nœuds. La boucle (lignes 14-30) parcourt les nœuds inclus dans Λ. Pour chaque nœud $node \in \Lambda$, L^{node} est la liste des labels contenus sur celui-ci. Chaque label $L_i \in$

L^{node} est propagé (boucle de la ligne 16 à la ligne 30) s'il ne l'a pas déjà été (condition $L_i.p == 0$). Pour chaque successeur possible j, contenu dans la liste $L_i.lsa$ du label L_i (boucle de la ligne 18 à la ligne 28), le label L_i est propagé (ligne 21).

Algorithme 2 : Algorithme de principe du problème esclave

1. **Sortie**
2. L_{best} : le label avec le coût réduit (CR) minimal
3. **Entrée**
4. w : l'employé
5. α : les duales des visites
6. β_w : la valeur de la duale associée à w
7. $Data$: les données du problème (distances, coûts, préférences, etc.)
8. **Variables**
9. L_{init} : le label initial
10. Λ : file de nœuds
11. **Début**
12. | L_{init} = creer_label_initial(β_w) ; // création du label initial
13. | $\Lambda.push(I_w)$ // insertion du dépôt de départ dans la file des nœuds
14. | **Tant que** $\Lambda \neq \emptyset$ **Faire**
15. | | $node = \Lambda.pop()$ // récupère le nœud courant
16. | | **Pour** $\forall L_i \in L^{node} \mid L_i.p == 0$ **Faire** // pour chaque label qui n'a pas déjà été
17. | | // propagé ($L_i.p = 0$)
18. | | | **Pour** $\forall j \in L_i.lsa$ **Faire** // le label L_i est propagé vers les nœuds de lsa
19. | | | Max_label = {1 ; 2} // nombre maximal de label à créer
20. | | | | **Pour** $z = 1$ à Max_label **Faire** // création des différents labels
21. | | | | | P = nouveau_label(L_i, α, j) // création d'un nouveau label P
22. | | | | | **Si** P_non_domine(P, L_i) **Faire** // si P n'est pas dominé
23. | | | | | | insertion_label(P, L^j) // insertion de P dans la liste des labels du nœud j
24. | | | | | | $\Lambda.push(j)$ // insertion de j dans la file des nœuds
25. | | | | | | sauv_best_label(L_{best}, P) // conservation du meilleur label
25. | | | | | **FinSi**
26. | | | | **FinPour**
27. | | | **FinPour**
28. | | | $L_i.p = 1$ // le label a été propagé
29. | | **FinPour**
30. | **FinTantQue**
31. | **Retourner** L_{best}
32. **Fin**

La nouveauté de l'algorithme se situe dans la propagation du label L_i car un ou deux nouveaux labels sont créés (ligne 19), alors que dans les algorithmes classiques, un unique label est fabriqué. Deux labels, dépendant de L_i et des duales α, peuvent être générés pour prendre en compte les différentes dates possibles de début de la visite (voir section 5.5.4).

Un nouveau label P est comparé aux labels déjà présents dans la liste L^j des labels du nœud j (ligne 22). Si P n'est pas dominé, alors *(i)* P est inséré dans L^j (ligne 23) ; *(ii)* le nœud j est inséré dans la file Λ ; et *(iii)* P est comparé au meilleur label L_{best} en fonction du coût réduit. Le meilleur label de coût minimal est stocké dans L_{best} (ligne 25). La notion de dominance entre labels est abordée dans la section 5.5.5.

Le nombre de labels étant exponentiel, il est important de définir des règles de propagation (section 5.5.4) et de dominance efficaces (section 5.5.5). Par ailleurs, des techniques d'accélération de la résolution (section 5.5.6) sont présentées.

5.5.4. Règle de propagation

Le label L appartenant au nœud $j \in V$ est propagé vers un nœud k si et seulement si $k \in L.lsa$. Propager le label du nœud j au nœud k signifie que le nœud k est ajouté à la fin de la tournée, après le nœud j et avant le retour au dépôt (L_w) : la tournée est de la forme $I_w - \cdots - j - k - L_w$.

Après propagation du label L du nœud j au nœud k, un nouveau label P est obtenu et situé au nœud k. Les différents attributs du label P doivent être mis à jour grâce aux règles suivantes :

- $P.RC = L.RC + \lambda_1(T_{k,L_w} + T_{j,k} - T_{j,L_w} + p_k^w) + \lambda_2(3 - \rho_k^w) + \lambda_3(\psi_k^w + \theta_k^w) - \alpha_k$.
- $P.CT = L.CT + \lambda_1(T_{k,L_w} + T_{j,k} - T_{j,L_w} + p_k^w) + \lambda_2(3 - \rho_k^w) + \lambda_3(\psi_k^w + \theta_k^w)$.
- $\forall v \in V \backslash k, P.\Delta_v = L.\Delta_v$ et $P.\Delta_k = 1$.
- $P.a_k = L.d_j + T_{j,k}$.
- $P.d_k = P.st_k + Pt_k$: l'employé repart au plus tôt.
- $P.p = 0$.
- $P.lsa = L.lsa \backslash \{k\}$ et la liste lsa doit en plus être mise à jour.
- $P.st_k$ peut prendre plusieurs valeurs du fait que la violation de la fenêtre de temps des employés est autorisée.

Les employés peuvent débuter et/ou finir une visite en dehors de leur fenêtre de travail, mais dans ce cas, une pénalité est comptée dans la fonction objectif (voir section 2.2). Par conséquent, suivant la date de début st_k de la visite, le coût de la tournée (et donc le coût réduit également) est impacté. Par exemple, il peut être plus intéressant pour un employé de commencer une visite en dehors de sa fenêtre de travail afin de réaliser plus de visites durant sa tournée. A contrario, dans certains cas, il peut être plus intéressant de retarder le début de la visite d'un employé afin que celui-ci ne viole pas sa fenêtre de temps de travail et qu'aucune pénalité ne soit comptée.

La prise en compte de cette pénalité dans la résolution du sous-problème mène à la création d'une nouvelle procédure de programmation dynamique. Lors de la propagation d'un label dans les algorithmes classiques pour l'ESPPRC, un unique nouveau label est créé. Pour l'ESPPRC avec qualité de service des employés, un ou deux labels peuvent être générés suivant la date de début de la visite.

Le choix de la date de début de la visite a déjà été discuté dans la section 3.3, et seuls les points essentiels sont rappelés ici : (1) l'employé doit toujours partir au plus tôt (dès que la visite est finie) ; (2) à partir d'un label père, deux labels fils sont générés. Les deux labels fils se différencient par leur date de début de la visite :

- Pour le premier label fils, $st_k = max(a_k, E_k)$: la visite débute sans tenir compte de la fenêtre de travail de l'employé (il y a un risque de pénalité).
- Pour le second label fils, $st_k = max(a_k, TW_w^-)$: la visite commence lorsque la fenêtre de travail de w est ouverte (il n'y a pas de pénalité concernant cette visite).

Faisabilité d'un label et mise à jour de la liste lsa

Un label L sur le nœud j est propagé vers un nœud k, si et seulement si :

1) $CoTr_k^w = 1$: w est autorisé à réaliser la visite k.
2) L'employé peut arriver sur la visite k avant la fin de la fenêtre de temps de celle-ci, puisque les violations des fenêtres de temps des visites sont interdites (uniquement celles des employés sont autorisées). La date d'arrivée a_k de l'employé w sur le nœud k doit être inférieure ou égale à la borne supérieure de la fenêtre de temps $[E_k ; L_k]$ de la visite k : $a_k \leq L_k$. Si l'employé arrive après la fenêtre de temps de la visite ($a_k > L_k$), alors la visite k ne peut pas être accomplie lors de cette tournée : passer par le nœud engendre un coût supplémentaire de transport inutile. Il est donc préférable d'éviter ce nœud.
3) La date de début st_k de la visite k est comprise dans la fenêtre de temps $[E_k ; L_k]$.

La liste lsa des sommets visitables à partir du nœud k peut être mise à jour en vérifiant que, $\forall v \in lsa$, les trois contraintes précédemment citées sont vérifiées. Mettre à jour cette liste permet de limiter la propagation des labels vers des sommets inatteignables et facilite l'écriture d'une règle de dominance.

5.5.5. Règle de dominance

L'utilisation d'une règle de dominance (Current and Marsh, 1993; Feillet et al., 2004) est facultative, car l'algorithme énumère tous les chemins possibles à partir du nœud I_w pour l'employé w. Toutefois, une règle de dominance est cruciale dans la conception d'un algorithme efficace pour identifier les labels qui ne nécessitent pas d'être propagés.

Soient deux labels P et L situés sur un nœud j. Le label P domine le label L si et seulement si : $P.RC < L.RC + \sum_{i \in \{L.lsa\}} \alpha_i$. Dans ce cas, le chemin représenté par le label L ne pourra jamais être meilleur que celui du label P. Cette règle de dominance n'est pas très forte, mais permet de ne pas perdre l'optimalité.

5.5.6. Accélération de la résolution par programmation dynamique

Il existe de nombreuses méthodes permettant d'accélérer une génération de colonnes utilisant la programmation dynamique pour résoudre les problèmes esclaves (Desaulniers et al., 2001). Deux méthodes sont retenues dans les travaux faisant objet de ce manuscrit.

La première méthode d'accélération consiste à obtenir plusieurs colonnes (représentant plusieurs chemins distincts) lors de la résolution d'un problème esclave, et non plus à conserver uniquement la meilleure colonne. L'objectif est d'obtenir rapidement une base conséquente pour le problème maître.

La seconde méthode d'accélération implique de ne pas résoudre optimalement le problème esclave, mais de stopper la résolution lorsqu'un nombre suffisant de chemins avec un coût réduit négatif, est trouvé. Résoudre optimalement un problème esclave à l'itération i ne certifie pas que la colonne k^i de coût réduit négatif (et optimale lors de cette résolution) appartiendra à la solution finale optimale à l'itération z. La résolution optimale nécessite un temps de calcul plus important, sans garantir l'utilisation de la colonne dans la solution finale. Il n'est donc pas forcément utile de résoudre tous les problèmes esclaves à l'optimal. Toutefois, lors des itérations finales, le problème esclave doit être résolu optimalement afin de prouver l'optimalité de la solution.

La résolution du problème esclave par programmation dynamique permet, théoriquement, d'obtenir la solution optimale du problème maître. Toutefois, le nombre exponentiel de labels est un frein considérable à la résolution exacte du problème esclave, ce qui entraîne un temps de résolution très important. Le nombre maximal de labels par nœud est donc limité par une constante. Cette constante est un paramètre important de la programmation dynamique permettant d'obtenir un compromis entre un temps de résolution court (peu de labels par nœud) et une solution proche de la solution exacte (un grand nombre de labels par nœud). L'impact de ce paramètre est évalué dans les études numériques.

5.6. Solution en nombres entiers

Une génération de colonnes impose que la contrainte d'intégrité de la variable x_z soit relâchée et ne garantit donc pas d'obtenir une solution entière. Afin d'obtenir une solution entière, deux méthodes sont développées dans ce chapitre : une approche heuristique et une approche exacte.

5.6.1. Résolution en nombres entiers du problème maître à la racine

À la fin de la génération de colonnes, sur le nœud racine, le problème maître est de nouveau résolu en imposant que la variable y_k soit entière. Seules les tournées générées durant les itérations de la génération de colonnes sont donc prises en compte, les tournées ne peuvent pas être modifiées, et aucune nouvelle tournée ne peut être créée.

Cette méthode, rapide et heuristique, est très dépendante des colonnes obtenues et stockées durant le déroulement de la génération de colonnes, et ne garantit pas l'optimalité de la solution. En effet, pour que la solution optimale soit obtenue, il faut que toutes les colonnes qui la composent soient déjà contenues dans la base.

Cette approche équivaut à résoudre le problème maître en nombres entiers sur le nœud racine du Branch-and-Price.

5.6.2. Approche exacte : Branch-and-Price

La seconde approche développée pour obtenir une solution en nombres entiers est une recherche arborescente de type Branch-and-Price. Cette méthode est itérative et consiste à résoudre le problème restreint, à chaque nœud de l'arbre, par la génération de colonnes. Une solution entière représente une feuille de l'arbre tandis qu'une solution fractionnaire est un nœud père et engendre deux fils. Si la solution obtenue à un nœud de l'arbre de recherche n'est pas entière, alors deux branches (ou fils) sont créées avec une contrainte supplémentaire : sur une branche la contrainte impose que deux visites se suivent dans les tournées, tandis que la seconde branche interdit à deux visites d'être successives dans les tournées.

Si la solution obtenue à la fin de la génération de colonnes est fractionnaire, alors une matrice à deux dimensions $ArcUsage$, est construite. Pour chaque colonne, $Suc_{i,j} = 1$ si la visite i est directement suivie de la visite j ; sinon $Suc_{i,j} = 0$. $ArcUsage$ est la somme de la valeur des y_k, associée à une colonne qui comprend la visite i immédiatement suivie de la visite j :

$$ArcUsage_{i,j} = \sum_{k \in \Omega | Suc_{i,j}=1} y_k \qquad \forall (i,j) \in V^2$$

L'arc avec l'$ArcUsage$ le plus proche de 0.5 est sélectionné et une contrainte est imposée sur celui-ci pour obtenir les deux nœuds fils. La première branche impose que, quel que soit l'employé $w \in W$, si la visite i est effectuée durant une tournée alors la visite j doit être réalisée immédiatement après. La contrainte sur la seconde branche interdit que la visite j soit effectuée directement après la visite i.

Afin d'accélérer la recherche arborescente, des coupes sont effectuées : si la solution fractionnaire S_f d'un nœud est supérieure à la meilleure solution entière trouvée S^* alors il n'est pas utile de résoudre le problème associé à ce nœud. La recherche arborescente permet, en théorie, d'obtenir la solution optimale du problème, mais elle est parfois limitée afin de conserver un temps de calcul raisonnable.

6. Études numériques

Dans la littérature, il existe un jeu d'instances pour le WSRP introduit par (Castillo-Salazar et al., 2014). Toutefois quelques interrogations sur ces instances (section 6.1) ne nous ont pas permis de les utiliser. Par conséquent, les différentes méthodes de résolution sont testées sur un nouveau jeu d'instances pour le WSRP (section 6.2).

6.1. Instances de (Castillo-Salazar et al., 2014)

(Castillo-Salazar et al., 2014) introduisent un jeu d'instances pour le WSRP, qui est repris dans de nombreux articles (voir Tableau 1, page 154), et notamment par (Laesanklang and Landa-Silva, 2017). Ces instances appellent quelques commentaires :

- Les fenêtres de temps concernant les dates de début des visites sont des singletons du type $[a\,;a]$ (où $a \in \mathbb{N}$), et ne permettent pas de flexibilité sur ces visites. La date de début d'une visite est donc directement induite par les données du problème.
- Les coefficients choisis posent également question. Le coefficient pour le critère 1 (distance plus coût d'exécution) est inférieur à 10^{-4}. Ce critère n'intervient donc presque pas sur la solution (encore moins lorsque les résultats publiés sont à 10^{-2} de précision). Le problème est donc « sous contraint » à propos de la partie transport.
- Par ailleurs, de nombreuses visites n'ont qu'un seul employé autorisé à les réaliser ($i \in V, !\, w \in W | CoTr_i^w = 1$). Il n'y a donc pas de problème d'affectation. Pour d'autres visites, par exemple pour la visite i, les coûts p_i^w, ainsi que les préférences ρ_i^w et Pa_i^w, sont tellement en faveur d'un employé w^* par rapport aux autres employés qu'il est quasiment obligatoire d'affecter i à l'employé w^* afin d'obtenir une solution avec un coût raisonnable. Par exemple, soit une visite i, un unique employé w^* et les autres employés $w' \in W | \, w' \neq w^*$: $p_i^{w^*} = 1$ et $p_i^{w'} = 1000$ sont le coût de réalisation de la visite ; $\rho_i^{w^*} = 3$ et $\rho_i^{w'} = 0$ sont les valeurs de la qualité de service des clients (une valeur proche de 3 indique qu'un employé est vivement souhaité, au contraire une valeur proche de zéro signifie que l'employé n'est pas désiré) ; $Pa_i^{w^*} = 1$ et $Pa_i^{w'} = 0$ sont les valeurs de la qualité de service des employés concernant la zone géographique (1 signifie que l'employé veut bien se déplacer dans cette

zone, 0 indique que l'employé ne souhaite pas aller dans cette zone). Avec de telles valeurs, il est évident qu'il vaut mieux privilégier l'unique employé w^* que tout autre employé w'.

Les auteurs de l'article (Laesanklang and Landa-Silva, 2017) offrent le détail des solutions (affectation, tournées, dates de début, etc.) correspondant aux résultats de leur article. Les valeurs des solutions optimales S^*, ainsi que des solutions S^h (obtenues par une heuristique) issues de l'article (Laesanklang and Landa-Silva, 2017) sont respectivement retranscrites dans les colonnes deux et trois du Tableau 5. Les auteurs fournissent également un exécutable permettant d'évaluer leur solution et donnant la valeur S^t de la quatrième colonne du Tableau 5. La cinquième colonne du tableau présente les résultats S^e obtenus par notre propre fonction d'évaluation des solutions. La dernière colonne est une résolution (par CPLEX) des instances par PLNE avec notre propre implémentation des contraintes, pour obtenir la valeur S'. Un doute persiste sur plusieurs points :

1) Le format des données de l'instance C-02 n'a pas pu être relu, car des informations sont manquantes.

2) Les valeurs S^h des solutions des instances D-01 à E-07 issues de l'article (Laesanklang and Landa-Silva, 2017) (troisième colonne) sont différentes des valeurs obtenues S^t par leur outil d'évaluation des solutions (quatrième colonne). Par exemple, dans l'article, pour l'instance D-01, $S^h = 107.77$ et leur outil d'évaluation obtient $S^t = 176.16$.

3) Il persiste un doute sur la fonction d'évaluation utilisée car un recodage de la fonction d'évaluation n'a pas permis de confirmer les valeurs numériques publiées pour certaines instances. Par exemple, pour l'instance A-01, la solution évaluée par leur outil vaut $S^t = 3.49$ et la valeur de la solution par notre fonction d'évaluation est $S^e = 3.17$. Un autre exemple pour l'instance D-01, $S^t = 176.16$ et $S^e = 176.37$.

4) Il est probable qu'il y ait de mauvaises compréhensions des contraintes à respecter car pour chaque instance, la valeur S^* de la solution optimale donnée dans l'article (Laesanklang and Landa-Silva, 2017) ne correspond pas à la solution optimale S' obtenue par PLNE avec CPLEX. Par exemple, pour l'instance A-01, la valeur de la solution optimale obtenue par PLNE est $S' = 3.12$, et la solution dite optimale est $S^* = 3.49$ d'après l'article (Laesanklang and Landa-Silva, 2017). La solution optimale fournie par CPLEX pour l'instance A-01 est entièrement décrite plus tard (par le Tableau 7 et le Tableau 8).

Tableau 5 Résultats numérique d'après (Laesanklang and Landa-Silva, 2017)

	Solutions issues de l'article de (Laesanklang and Landa-Silva, 2017)			**Solutions obtenues par PLNE**	
		Même solution en entrée			
	Coût de la solution dite "optimale" S^*	Coût de la solution dans l'article S^h	Coût de la solution avec l'exécutable S^t	Coût de la solution par notre évaluation S^e	Coût de la solution optimale S'
A-01	3.49	3.49	**3.49**	**3.17**	**3.12**
A-02	2.49	2.49	2.49	1.62	1.61
A-03	3.00	3.02	3.00	2.05	1.98
A-04	1.42	1.42	1.42	0.60	0.52
A-05	2.42	2.42	2.42	1.57	1.57
A-06	3.55	3.55	3.55	2.55	2.52
A-07	3.71	3.71	3.71	2.94	2.56
B-01	1.70	1.74	1.70	0.59	0.29
B-02	1.75	1.75	1.75	0.84	0.68
B-03	1.72	1.80	1.72	0.38	0.23
B-04	2.07	2.08	2.07	0.51	0.43
B-05	1.82	1.92	1.82	0.52	0.36
B-06	1.62	1.65	1.62	0.49	0.26
B-07	1.79	1.81	1.79	0.52	0.26
C-01	/	116.77	116.96	116.76	113.41
C-02	3.15	3.15	3.15	/	/
C-03	/	104.97	11.17	11.15	103.45
C-04	11.15	11.15	11.15	11.17	11.13
C-05	12.34	12.41	12.34	12.44	12.43
C-06	/	180.93	180.87	181.02	140.32
C-07	4.30	4.30	4.30	4.30	4.30
D-01	**/**	**170.77**	**176.16**	**176.37**	**/**
D-02	/	176.98	171.56	171.88	/
D-03	/	181.87	181.88	182.00	/
D-04	/	169.13	175.35	175.46	/
D-05	/	165.14	162.69	162.77	/
D-06	/	178.81	178.54	178.95	/
D-07	/	178.92	178.02	177.92	/
E-01	/	1.21	1.26	1.22	/
E-02	/	1.21	2.28	1.23	/
E-03	/	1.26	1.58	1.25	/
E-04	/	1.03	1.42	1.34	/
E-05	/	2.24	2.78	2.30	/
E-06	/	1.30	0.95	1.34	/
E-07	/	1.73	1.76	1.75	/

Le Tableau 6 donne la valeur des quatre critères de la fonction objectif de la solution optimale trouvée par CPLEX pour chaque instance des datasets A, B et C. Pour les instances des datasets D et E, CPLEX n'arrive pas à fournir la solution optimale au bout d'une heure de calcul.

Par exemple, pour l'instance A-01, la valeur de la solution est $S' = 3.12$, la valeur du critère C1 (coût opérationnel plus distance) vaut 555.60, le critère C2 (qualité de service d'un point de vue clients) est 60.38, la qualité de service d'un point de vue employé (critère C3) est 4, et le critère C4 vaut 0 (nombre de visites non effectuées).

Tableau 6 Détail des solutions optimales trouvées par PLNE

		Critère C1	**Critère C2**	**Critère C3**	**Critère C4**
	S'	$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}(T_{i,j} + p_i^w)x_{i,j}^w$	$\sum_{i\in\mathcal{V}}\left(3 - \sum_{j\in\mathcal{V}}\rho_i^w\, x_{i,j}^w\right)$	$\sum_{i\in\mathcal{V}}(\psi_i^w + \theta_i^w)$	$\sum_{i\in\mathcal{V}} y_i$
A-01	3.12	555.60	60.38	4	0
A-02	1.61	529.95	73.62	2	0
A-03	1.98	639.78	96.21	3	0
A-04	0.52	407.90	69.92	0	0
A-05	1.57	237.25	25.15	1	0
A-06	2.52	511.73	70.08	4	0
A-07	2.56	247.71	24.33	3	0
B-01	0.29	951.24	98.16	0	0
B-02	0.68	318.43	28.48	0	0
B-03	0.23	1235.32	197.3	0	0
B-04	0.43	378.64	77.68	0	0
B-05	0.36	1153.75	164.56	0	0
B-06	0.26	917.87	156.85	0	0
B-07	0.26	1000.82	167.56	0	0
C-01	113.41	43797.19	167.88	136	1
C-02					
C-03	103.45	349828.82	155.39	128	1
C-04	11.13	4625.08	28.56	18	0
C-05	12.43	3304.08	21.23	21	0
C-06	140.32	38755.94	181.46	119	2
C-07	4.30	1408.63	7.99	5	0

Pour l'instance A-01, Le Tableau 7 et le Tableau 8 présentent le détail du calcul de la solution optimale obtenue par PLNE, pour les critères C1 et C2 (Tableau 7), et pour le critère C3 (Tableau 8). La première colonne (w) correspond au numéro de l'employé affecté à la tournée qui est reportée dans la seconde colonne.

Pour le Tableau 7, la distance parcourue par l'employé $w = 1$ est $\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} T_{i,j}x_{i,j}^w = 21.79$ (troisième colonne), et la somme des p_i^w de cette tournée est $\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} p_i^w x_{i,j}^w = 25.98$ (cinquième colonne). La dernière colonne de ce tableau indique la valeur du critère C2 pour chaque tournée. Par exemple, cette valeur est $\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} \rho_i^w\, x_{i,j}^w = 3.88$ pour la tournée de l'employé $w = 1$. Les dernières lignes de chaque colonne correspondent à la somme de celle-ci pour l'ensemble des tournées. $\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}(T_{i,j} + p_i^w)x_{i,j}^w = 555.6$ est la valeur finale du critère C1, et $\sum_{i\in\mathcal{V}}\left(3 - \sum_{j\in\mathcal{V}}\rho_i^w\, x_{i,j}^w\right) = 35.62$ est la valeur totale du critère C2.

Tableau 7 Détail des critères C1 et C2 pour la solution optimale de l'instance A-01

w	Tournée	Critère C1			Critère C2	
		$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} T_{i,j}x_{i,j}^{w}$	Détail des p_i^w	$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} p_i^w x_{i,j}^{w}$	ρ_j^w des visites	$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} \rho_i^w\ x_{i,j}^{w}$
1	0 – 18 – 16 – 0	21.79	0 – 6.44 – 19.54 – 0	25.98	0 – 1.96 – 1.92 - 0	3.88
2	/	0.00	/	0.00	/	0.00
3	0 – 24 – 27 – 15 – 22 – 0	23.80	0 – 0.00 – 4.63 - 6.20 – 6.20 - 0	17.03	0 – 1.39 – 2.52 – 1.93 – 1.39 - 0	7.23
4	0 – 13 – 19 – 17 – 5- 25 - 0	41.44	0 – 0.00 – 0.00 – 6.44 – 6.44 – 3.17 - 0	16.05	0 – 1.98 – 1.95 – 1.98 – 1.86 – 1.98 - 0	9.75
5	0 – 23 – 10 – 29 – 0	21.57	0 – 4.63 – 6.20 – 7.78 - 0	18.61	0 – 1.87 – 1.86 – 1.91 - 0	5.64
6	/	0.00	/	0.00	/	0.00
7	0 – 1 – 4 – 21 – 0	1.90	0 – 0.00 – 9.63 – 6.38 – 0	16.01	0 – 1.96 – 1.92 – 1.96 - 0	5.84
8	0 – 6 – 28 – 0	23.70	0 – 9.63 – 4.76 - 0	14.39	0 – 1.97 – 2.34 - 0	4.31
9	0 – 31 – 14 – 16 – 0	40.20	0 – 1.51 – 12.87 – 19.36 - 0	33.74	0 – 2.97 – 1.94 – 1.88 -0	6.79
10	0 – 2 – 0	32.20	0 – 0.00 - 0	0.00	0 – 1.50 - 0	1.50
11	/	0.00	/	0.00	/	0.00
12	0 – 7 – 12 – 3 – 0	23.20	0 – 9.63 – 6.38 – 6.38 - 0	22.39	0 – 1.85 – 1.95 – 1.94 - 0	5.74
13	/	0.00	/	0.00	/	0.00
14	/	0.00	/	0.00	/	0.00
15	0 – 11 – 8 – 30 – 0	50.79	0 – 6.54 – 6.54 – 4.88 -0	17.97	0 – 1 – 1.89 – 2.64 - 0	5.53
16	/	0.00	/	0.00	/	0.00
17	0 – 20 – 0	49.80	0 – 0.00 - 0	0.00	0 – 1.67 - 0	1.67
18	/	0.00	/	0.00	/	0.00
19	0 – 26 – 9 – 0	43.11	0 – 0.00 – 0.00 -0	0.00	0 – 1.50 – 1.00 - 0	2.50
20	/	0.00	/	0.00	/	0.00
21	/	0.00	/	0.00	/	0.00
22	/	0.00	/	0.00	/	0.00
23	/	0.00	/	0.00	/	0.00
Somme		**373.5**		**182. 16**		**60.38**
$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}(T_{i,j} + p_i^w)x_{i,j}^w$			**555.6**			
$\sum_{i\in\mathcal{V}}(3 - \sum_{j\in\mathcal{V}} \rho_i^w\ x_{i,j}^w)$					**35.62=96-60.38**	

Tableau 8 Détail du critère C3 pour la solution optimale de l'instance A-01

w	Tournée	Critère C3			
		$\sum_{i \in V} \psi_i^w$	Fenêtre de temps des employés	Date de début et de fin des opérations	$\sum_{i \in V} \theta_i^w$
1	0 – 18 – 16 – 0	0	[525 ;899]	0 – 540/ – 720/899 - 0	0
2	/	0	/		0
3	0 – 24 – 27 – 15 – 22 – 0	0	[480 ; 1139]	0 – 480/ - 720/ - 960/ - 1040/1099 -0	0
4	0 – 13 – 19 – 17 – 5- 25 - 0	0	[420 ; 1319]	0 – 465/ - 480/ - 555/ - 750/ - 1110/1139 - 0	0
5	0 – 23 – 10 – 29 – 0	0	[480 ; 1139]	0- 525/ - 600/ - 1035/1109 - 0	0
6	/	0	/		0
7	0 – 1 – 4 – 21 – 0	0	[420 ; 1079]	0- 465/ - 600/ - 690/749 - 0	0
8	0 – 6 – 28 – 0	0	[420 ; 839]	0 - 615/ - 720/764 - 0	0
9	0 – 31 – 14 – 16 – 0	0	[480 ; 959]	0- 510/ - 570/ - 720/899 - 0	0
10	0 – 2 – 0	0	[0 ; 0]	0- 480/ - 0	1
11	/	0	/		0
12	0 – 7 – 12 – 3 – 0	0	[510 ; 989]	0- 510/ - 645/ - 720/ 779- 0	0
13	/	0	/		0
14	/	0	/		0
15	0 – 11 – 8 – 30 – 0	0	[570 ; 884]	0- 570/ - 750/ - 825/869 - 0	0
16	/	0	/		0
17	0 – 20 – 0	0	[0 ; 0]	0- 1200/ - 0	1
18	/	0	/		0
19	0 – 26 – 9 – 0	0	[0 ; 0]	0- 480/ - 1200/ - 0	2
20	/	0	/		0
21	/	0	/		0
22	/	0	/		0
23	/	0	/		0
	Somme	**0**			**4**
	$\sum_{i \in V} (\boldsymbol{\psi_i^w + \theta_i^w})$				**4**

Le Tableau 8 complète le Tableau 7 et définit le critère C3 (violation des fenêtres de temps des employés, et violation des régions préférées des employés). $\sum_{i\in\mathcal{V}} \psi_i^w = 0$ signifie qu'il n'y a aucune violation des régions, par exemple pour la tournée de $w = 1$. θ_i^w concerne les violations des fenêtres de temps et $\sum_{i\in\mathcal{V}} \theta_i^w = 1$ signifie qu'il y a une violation de la fenêtre de temps d'un employé. Pour la tournée $w = 10$, la visite débute à la date 480, et l'employé a pour fenêtre de temps [0 ; 0], il y a donc une violation.

Le Tableau 9 résume les valeurs obtenues pour les différents critères, les coefficients λ, ainsi que la valeur totale de la solution, qui est $S' = 3.1186$.

Tableau 9 Résumé des valeurs des différents critères

	C1	**C2**	**C3**	**C4**
Équation	$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}(T_{i,j} + p_i^w)x_{i,j}^w$	$\sum_{i\in\mathcal{V}}\left(3 - \sum_{j\in\mathcal{V}} \rho_i^w\, x_{i,j}^w\right)$	$\sum_{i\in\mathcal{V}}(\psi_i^w + \theta_i^w)$	$\sum_{i\in\mathcal{V}} y_j$
Valeur	555.6	35.62	4	0
Coefficient λ	9.86071931186378E-06	0.03125	0.50	8.00
λ. valeur	0.00547861564	1.113125	2	0
TOTAL (S')		**3.1186**		

Face aux difficultés présentées dans cette section, de nouvelles instances sont proposées pour tester les différentes approches de résolution du WSRP.

6.2. Instances GGLT

Face aux difficultés présentées dans les sections précédentes, et pour le cas de l'étude du problème du WSRP puis du GWSRP (qui est l'objet du quatrième chapitre du manuscrit), il a été introduit de nouvelles instances, nommées GGLT (Garaix-Gondran-Lacomme-Tchernev). Ces nouvelles instances sont plus générales que celles utilisées par (Castillo-Salazar et al., 2014) et (Laesanklang and Landa-Silva, 2017). Elles définissent, par exemple, des fenêtres de temps qui ne se réduisent pas à une date unique, et des clients ayant une liste d'employés potentiels qui ne se réduit pas à une unique possibilité d'affectation client/employé.

Les distances présentent dans ces instances sont issues des différentes instances de (Castillo-Salazar et al., 2014). Les autres autres valeurs sont générées aléatoirement en suivant une loi normale.

Les instances sont téléchargeables à partir de la page web :
http://fc.isima.fr/~gondran/researches.html.

6.2.1. Présentation des instances et des études

Le Tableau 10 présente les caractéristiques des instances GGLT. Le nombre d'employés varie de 3 (instance 5) à 55 (instance 13) et le nombre de visites de 13 (instance 5) à 177 (instance 13).

Tableau 10 Présentation des instances GGLT

Instance	Nombre de visites	Nombre d'employés	λ_1	λ_2	λ_3	λ_4
1	32	5	0.1	10	100	10000
2	31	6	0.1	10	100	10000
3	38	7	0.1	10	100	10000
4	28	8	0.1	10	100	10000
5	13	2	0.1	10	100	10000
6	28	4	0.1	10	100	10000
7	36	9	0.1	10	100	10000
8	71	34	0.1	10	100	10000
9	30	10	0.1	10	100	10000
10	62	16	0.1	10	100	10000
11	57	17	0.1	10	100	10000
12	61	21	0.1	10	100	10000
13	177	55	0.1	10	100	10000

Le Tableau 11 présente les différentes expériences réalisées. Une partie des expériences concerne une résolution directe des instances par PLNE et PPC (ligne 1 et 2 du tableau). Une autre partie des expériences porte sur la génération de colonnes. Pour les expériences Exp 1, Exp 2 et Exp 3, le problème maître correspond au Set Partitioning Problem (SPP, voir la section 5.3.1) ; tandis que pour l'Exp 4, le problème maître correspond au Set Covering Problem (SCP, voir la section 5.3.2). La résolution du problème esclave est soit en PLNE (expérience Exp 1), soit par programmation dynamique (expériences Exp 2, Exp 3 et Exp 4). Pour obtenir une solution en nombres entiers (voir la section 5.6), un Branch-and-Price (B&P) est utilisé pour l'expérience Exp 2, et une résolution heuristique est appliquée aux expériences Exp 1, Exp 3 et Exp 4.

Tableau 11 Présentation des expériences

	Résolutions directes		**Génération de colonnes**					
			Problème maître		Problème esclave		Solution en nombres entiers	
	PLNE	PPC	SPP	SCP	PLNE	Prog. dyn.	B&P	PLNE
PLNE	X							
PPC		X						
Exp 1			X		X			X
Exp 2			X			X	X	
Exp 3			X			X		X
Exp 4				X		X		X

La résolution par CPLEX (PLNE) des instances est limitée à 10 800 secondes et, de plus, CPLEX est exécuté sur un seul thread (résolution directe par PLNE ou dans la génération de colonnes). Le nombre de labels dans la programmation dynamique est limité à 250, le nombre de colonnes à remonter est 10 par résolution du problème esclave. La profondeur de l'arbre du Branch-and-Price est limitée à 10 nœuds. La génération de colonnes et la résolution par PPC sont limitées à 3600 secondes.

Les différentes études sont toutes effectuées dans les mêmes conditions sous Windows 7, avec un ordinateur Intel Core i7-4790 3.60 GHz avec 16.0 Go de RAM, ce qui équivaut à 2671 MFlops selon (Dongarra, 2014) (http://www.roylongbottom.org.uk/linpackresults.htm) et http://asteroidsathome.net/boinc/cpu_list.php.

La génération de colonnes (problème esclave inclu) est implémentée en C++. Le modèle linéaire est résolu par CPLEX 12.7.0. Le modèle PPC est implémenté en Java et utilise le solveur Choco 4.10.0.

6.2.2. Résolutions des instances GGLT par PLNE et PPC

Le Tableau 12 présente les résultats des résolutions directes par PLNE et PPC. Le tableau introduit pour la résolution PLNE avec CPLEX : la borne inférieure ($Binf$), la borne supérieure ($Bsup$) et le Gap sont donnés. Une borne inférieure égale à la borne supérieure signifie que CPLEX a résolu l'instance à l'optimalité. C'est le cas notamment dans l'instance 1, avec $Binf = 697.6$, et $Bsup = 697.6$. Le gap est donc $Gap = 0.0\%$, et le temps de résolution est donné par la colonne TT, ici $TT = 6079.5$ secondes. L'instance 3 n'est pas résolue optimalement en 10800.5 secondes, le gap est donc non nul ($Gap = 1.0\%$). La meilleure solution connue, pour cette instance, a pour coût 886.7.

Tableau 12 Résultats numériques lors des résolutions directes pour les instances GGLT

	PLNE				**PPC**			
Instance	$Binf$	$Bsup$	$Gap(\%)$	TT	S	T	TT	Gap_PLNE
1	**697.6**	**697.6**	0.0	6079.5	813.7	3480.6	3600.0	16.6
2	**203.3**	**203.3**	0.0	1540.4	381.1	3024.2	3600.0	87.5
3	877.9	886.7	1.0	10800.5	1204.4	2207.9	3600.0	35.8
4	**516.8**	**516.8**	0.0	3414.0	533.1	2528.5	3600.0	3.2
5	**105.0**	**105.0**	0.0	24.3	**105.0**	18.8	85.0	0.0
6	**920.9**	**920.9**	0.0	87.8	1420.1	3365.4	3600.0	54.2
7	**458.5**	**458.5**	0.0	437.1	696.0	3318.2	3600.0	51.8
8	606.2	617.5	1.8	10822.8	3042.8	3536.4	3600.0	392.8
9	**124.0**	**124.0**	0.0	6.5	181.5	3591.8	3600.0	46.4
10	**478.9**	**478.9**	0.0	357.1	876.1	3377.6	3600.0	82.9
11	**409.3**	**409.3**	0.0	839.6	1404.8	3596.9	3600.0	243.2
12	394.5	402.3	1.9	10937.6	2445.8	3236.2	3600.0	508.0
13	4737.5	7550.6	37.3	10817.1	/	/	3600.0	/
Moy 1-12				**3778.9**		2940.2	3307.1	126.9
Moy 1-13				**4320.3**		/	/	/

La dernière partie Tableau 12 introduit les résultats obtenus par résolution du modèle PPC. S est le coût de la meilleure solution trouvée, T est le temps pour obtenir cette solution et TT est le temps total de résolution. Une seule instance est résolue optimalement. Il s'agit de l'instance 5, dont la meilleure solution est trouvée en 18.8 secondes, puis prouvée optimale en un temps total de 85.0 secondes. Le Gap_PLNE, en pourcentages, est le gap entre la meilleure solution trouvée par PLNE ($Bsup$) et la solution trouvée par PPC. Ce gap est très important pour la PPC.

Ces résultats, concernant la PPC, ne sont pas surprenants. En effet, les modèles PPC permettent en général de trouver une première solution rapidement (exceptée pour l'instance 13), mais ont ensuite des difficultés à trouver une solution de très bonne qualité. D'autres stratégies, de parcours de l'arbre de recherche et de propagation de contraintes, devraient être étudiées afin d'améliorer ces résultats.

Ces premiers résultats montrent la difficulté des méthodes exactes pour résoudre les instances de façon optimale, puisque la PLNE ne résout que neuf instances et la PPC une seule instance.

Le Tableau 13 présente la convergence des méthodes de résolution par PLNE et PPC pendant la première minute d'exécution. La valeur de la meilleure solution courante S est mesurée à quatre instants, au bout de 1, 15, 30 et 60 secondes. Les valeurs de la première solution, ainsi que le temps pour la trouver, sont également retranscrites.

Par exemple, pour l'instance 1, la première solution trouvée par CPLEX avec le modèle PLNE a pour valeur $S = 150439.0$ au bout de 1.1 seconde, tandis que la première solution trouvée par Choco a pour valeur $S = 826.2$ au bout de 0.5 seconde. Au bout de 1 seconde, la solution CPLEX vaut $S = 150439.0$ (c'est également la première solution), tandis que la solution Choco vaut 818.7. Après 15 secondes, la solution CPLEX est $S = 30657.2$, et la solution Choco est $S = 815.4$. Après 30 secondes, CPLEX n'a pas réussi à améliorer la solution courante, mais au bout de 60 secondes, sa meilleure solution est $S = 10685.5$. À l'inverse, Choco améliore sa solution au bout de 30 secondes ($S = 815.0$), mais ne parvient pas à améliorer la solution par la suite.

La Figure 32 présente les courbes de convergence de la PLNE et la PPC pour l'instance 1, sur une échelle semi-logarithmique. La courbe de convergence de la PPC (en trait bleu plein) est quasiment une droite, tandis que la courbe de convergence de la PLNE (en orange et en pointillé) descend régulièrement. La courbe de la PLNE reste supérieure à celle de la PPC pour cette fenêtre de temps (d'une minute).

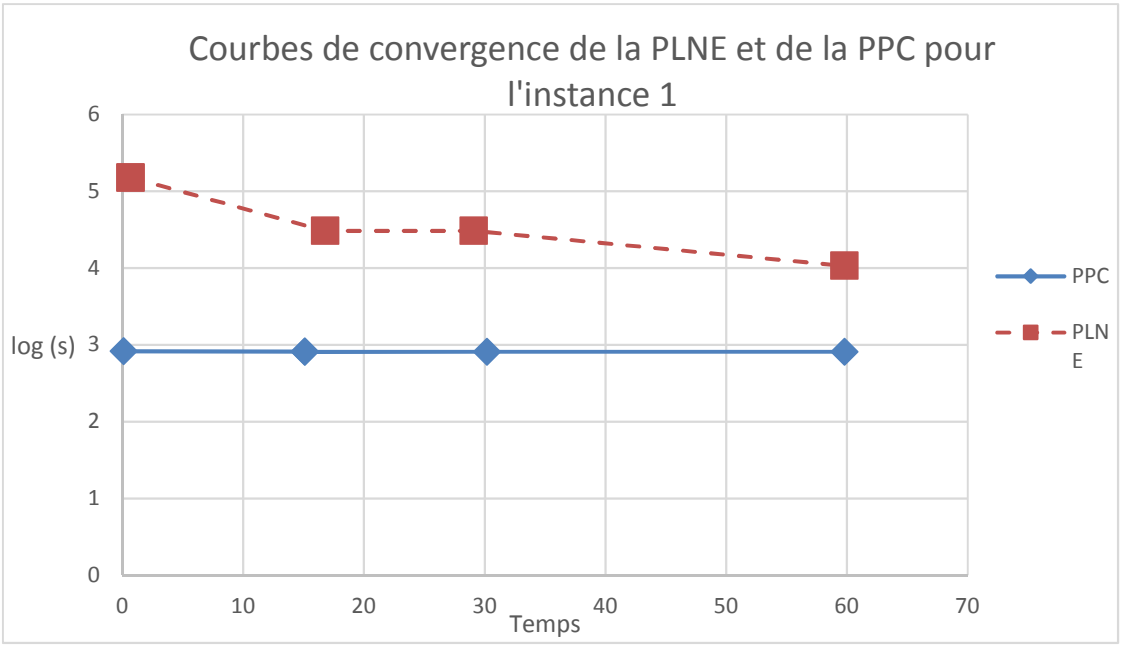


Figure 32 Courbes de convergence pour l'instance 1

Ces constats illustrent parfaitement l'avantage des modélisations de type PPC qui permettent de trouver très rapidement une solution de départ, alors que les modèles de type PLNE ont souvent plus de difficultés à trouver la première solution entière du problème. C'est ce type de comportements qui est observé ici.

Les méthodes de PLNE ont plus de difficultés pour trouver la première solution, mais peuvent converger plus « régulièrement » vers la solution optimale. Ceci est observé sur l'instance numéro 2, par exemple. La première solution obtenue par Choco est $S = 947.6$ en 0.1 seconde, tandis que la première solution CPLEX est $S = 130123.0$ en 0.7 seconde. Au bout de 1 seconde, Choco a une solution de valeur $S = 513.2$ et CPLEX conserve la solution antérieure $S = 130123.0$. Par contre, après 15 secondes, CPLEX a une solution de valeur S =364.3, tandis que Choco a une solution de valeur 513.2. Pour les instances 11 et 12, Choco a toujours une valeur de solution meilleure que celle proposée par CPLEX au bout de 60 secondes. La Figure 33 représente les courbes de convergence de la PLNE et de la PPC pour l'instance 2 sur une échelle semi-logarithmique. La courbe de convergence de la PLNE (en orange) débute plus haut que celle de convergence de la PPC (en bleu). Toutefois, la courbe de la PLNE passe en dessous de celle de la PPC au bout d'une quinzaine de secondes.

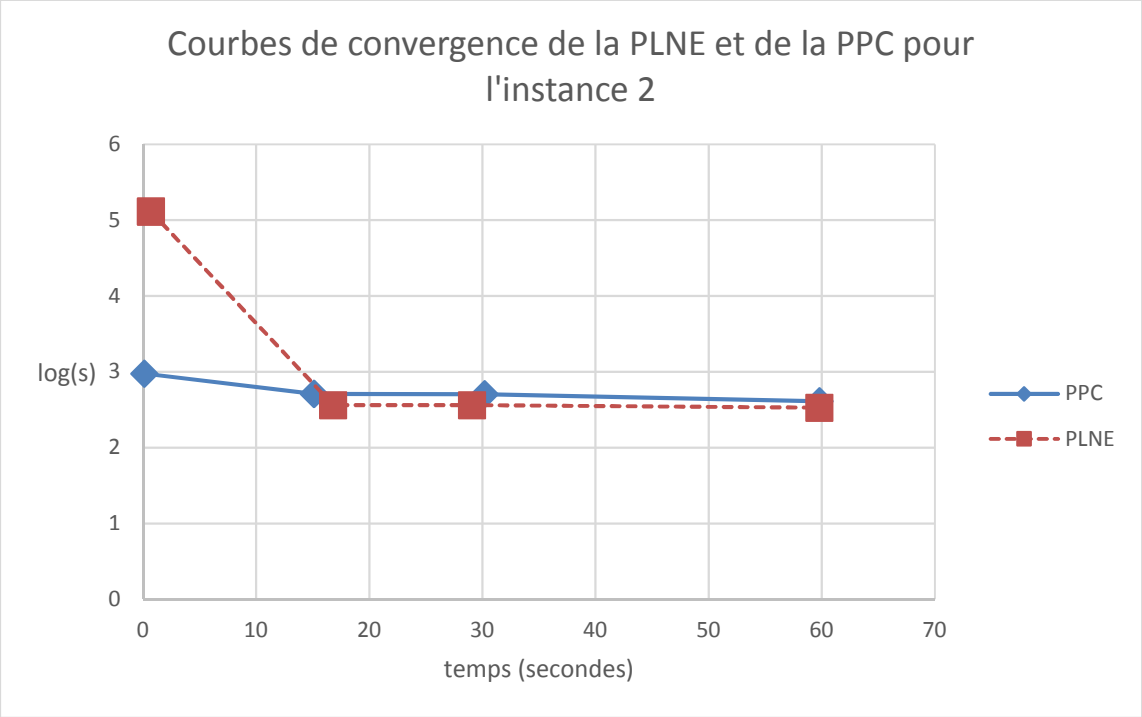


Figure 33 Courbes de convergence pour l'instance 2

Pour l'instance 13, Choco ne parvient pas à trouver une première solution, tandis que CPLEX réussit, mais celle-ci est de très mauvaise qualité.

Les moyennes des temps de résolution et des valeurs des solutions des 12 premières instances sont calculées (dernière ligne du Tableau 13). Par exemple, pour CPLEX, la première solution a en moyenne un coût de 15167.4 et est obtenue en 3.7 secondes, tandis que la première solution de Choco est obtenue en 5.1 secondes et a un coût moyen de 1670.0. Toutefois, il faut considérer les moyennes avec précaution, car il y a une grande disparité entre les instances.

Tableau 13 Convergence en PLNE et PPC

	PLNE										**PPC**									
	1° sol		1 s		15 s		30 s		60 s		1° sol		1 s		15 s		30 s		60 s	
	S	*T*	*S*	*T*	*S*	*T*	*S*	*T*	*S*	*T*	*S*	*T*	*S*	*T*	*S*	*T*	*S*	*T*	*S*	*T*
1	150439.0	1.1	150439.0	1.1	30657.2	13.4	/	/	10685.5	59.5	**826.2**	**0.5**	**818.7**	**1.0**	**815.4**	**8.5**	**815.0**	**29.5**	**815.0**	**29.5**
2	130123.0	0.7	130123.0	0.7	**364.3**	**16.8**	**364.3**	**29.1**	**337.6**	**59.8**	**947.6**	**0.1**	**513.2**	**15.1**	513.2	15.1	507.2	30.2	409.6	59.8
3	70724.6	1.0	70724.6	1.0	21314.2	16.3	21314.2	31.3	11195.0	61.5	**1853.6**	**0.1**	**1646.7**	**0.8**	**1453.0**	**16.2**	**1335.5**	**31.6**	**1335.5**	**31.6**
4	40377.6	1.0	40377.6	1.0	**519.9***	**14.9**	/	/	/	/	**539.0**	**0.1**	**539.0**	**0.1**	535.4	7.9	535.3	34.1	/	/
5	70056.6	1.1	70056.6	1.1	**107.5**	**13.5**	**105.0**	**24.3**	**105.0**	**24.3**	**213.9**	**0.0**	**107.1**	**1.0**	105.9	17.8	105.0	18.7	/	/
6	100592.0	0.3	100592.0	0.3	**1387.9**	**15.8**	**1200.2**	**31.0**	**922.0**	**59.3**	**2011.3**	**0.1**	**1730.0**	**0.8**	/	/	1640.9	35.4	1638.1	71.0
7	60481.5	1.6	60481.5	1.6	20551.6	16.7	**673.9**	**32.0**	**673.9**	**57.4**	**1443.2**	**0.2**	**924.8**	**0.6**	**713.0**	**14.9**	710.1	26.0	705.0	62.5
8	230497.0	17.9	/	/	230497.0	17.9	/	/	200531.0	62.2	**3908.2**	**47.8**	**/**	**/**	**/**	**/**	**/**	**/**	**3458.1**	**60.0**
9	210047.0	0.9	210047.0	0.9	**124.0***	**5.9**			/	/	**1836.2**	**0.2**	**1198.5**	**1.0**	914.3	15.3	827.4	29.7	812.0	64.1
10	240272.0	5.4	/	/	220294.0	15.7	**588.5**	**25.9**	**510.0**	**63.3**	**1403.3**	**6.1**	/	/	**891.7**	**16.4**	891.1	26.3	/	/
11	240207.0	6.9	/	/	180267.0	17.0	180267.0	32.1	/	/	**2182.3**	**1.4**	**2182.3**	**1.425s**	**1537.3**	**13.4**	**1534.5**	**24.8**	**1533.5**	**50.6**
12	330191.0	7.0	/	/	330191.0	17.1	190294.0	32.2	190294.0	59.8	**2874.9**	**4.0**	/	/	**2653.8**	**10.7**	/	/	**2653.8**	**10.7**
13	**632888.0**	**38.3**	**/**	**/**	**/**	**/**	**/**	**/**	**632888.0**	**59.8**	/	/	/	/	/	/	/	/	/	/
Moy 1-12	156167.4	3.7	156167.4	3.7	86356.3	15.1	54717.1	24.2	49680.4	46.7	1670.0	5.1	1487.2	7.1	1314.3	15.4	1288.7	28.7	1241.0	43.2

En gras : meilleure solution

Légende * solution optimale

/ Pas de solution, ou la solution n'est pas améliorée

6.2.3. Résolutions des instances GGLT par GC

Le Tableau 14 présente les résultats obtenus par résolution avec la génération de colonnes suivant les différentes expériences (cf. la section 6.2.1).

La Figure 34 résume les quatre expériences réalisées avec la génération de colonnes. Le problème maître correspond soit au Set Partitioning Problem (partie gauche de l'arbre), soit au Set Covering Problem (partie droite de l'arbre). Le problème esclave est soit résolu par PLNE, soit par Programmation Dynamique. La solution en nombres entiers est soit obtenue par Branch-and-Price, soit par PLNE (résolution du problème maître en nombres entiers).

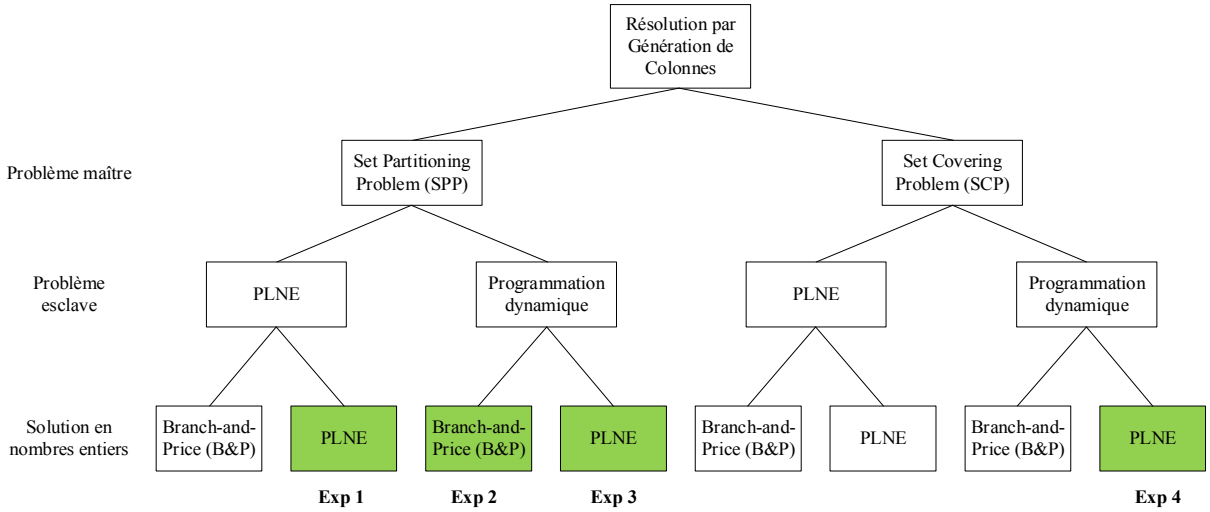


Figure 34 Expériences menées avec la génération de colonnes

Le Tableau 14 rappelle succinctement les résultats ($Bsup$ et TT) obtenus par PLNE. L'expérience Exp 1 est la résolution par génération de colonnes, avec le problème esclave résolu par PLNE et la solution finale obtenue par résolution du problème maître en nombres entiers (voir la Figure 34). SR est la valeur de la solution fractionnaire, S est la valeur de la solution finale en nombres entiers, TT est le temps total de résolution, et Gap_PLNE (en pourcentages) est le gap entre S et la solution $Bsup$ obtenue par PLNE.

Concernant l'expérience Exp 1, hormis pour les instances 4, 5, 7 et 9, le temps limite de calcul (3600 secondes) est atteint. La résolution du sous-problème par PLNE est longue et la génération de colonnes n'a donc pas le temps de s'exécuter totalement. Le coût des solutions est très élevé, car le coefficient λ_4, de la fonction objectif, pénalisant le nombre de visites non réalisées est de l'ordre de 10^4. Par exemple, une solution de coût $S = 8452.4$ (ce qui est le cas de l'instance 1), signifie que 8 visites n'ont pas été réalisées.

Il n'est pas surprenant qu'une telle approche de résolution ne fournisse pas de résultat de bonne qualité, car le nombre de colonnes à la fin de la génération de colonnes est très petit puisque peu d'itérations sont faites, et une unique colonne est remontée à chaque itération. Lors de la résolution en nombres entiers, il est très difficile de trouver une solution où toutes les visites sont réalisées une seule fois. La résolution finale du problème maître en nombres entiers ne permet donc pas de trouver une solution de bonne qualité.

Tableau 14 Résultats numériques de la GC pour les instances GGLT

	PLNE		**Exp 1**				**Exp 2**			**Exp 3**				**Exp 4**			
Instance	$Bsup$	TT	SR	S	TT	Gap_PLNE	S	TT	Gap_PLNE	SR	S	TT	Gap_PLNE	$S\ R$	S	TT	Gap_PLNE
1	**697.6**	6079.5	702.1	8452.0	3638.5	1111.6	698.0	27.0	0.1	698.0	699.1	13.0	0.2	698.0	698.9	13.1	0.2
2	**203.3**	1540.4	231.8	3204.3	3608.8	1476.1	203.8	8.5	0.3	203.8	203.8	8.9	0.3	205.2	205.2	6.8	0.9
3	886.7	10800.5	1433.4	10683.3	3647.1	1104.8	887.1	29.1	0.1	887.7	888.6	26.2	0.2	885.1	885.5	21.7	-0.1
4	**516.8**	3414.0	516.8	516.8	2441.2	0.0	517.8	42.2	0.2	518.2	518.6	10.6	0.4	518.5	518.9	8.0	0.4
5	**105.0**	24.3	105.0	105.0	789.5	0.0	105.0	1.1	0.0	105.0	105.0	1.2	0.0	105.0	105.0	1.2	0.0
6	**920.9**	87.8	966.5	6064.3	3650.1	558.5	928.0	30.4	0.8	933.7	938.8	13.7	1.9	927.9	927.9	18.3	0.8
7	**458.5**	437.1	458.2	458.8	3459.0	0.1	459.3	35.4	0.2	459.3	459.3	35.5	0.2	458.4	458.6	32.5	0.0
8	617.5	10822.8	9926.6	30781.0	3633.3	4884.8	621.0	668.5	0.6	622.1	622.1	622.3	0.7	625.9	625.9	473.0	1.4
9	**124.0**	6.5	124.0	124.0	2924.1	0.0	140.5	14.9	13.3	141.1	141.1	14.2	13.8	133.8	133.8	14.9	7.9
10	**478.9**	357.1	3267.7	27900.0	3631.9	5725.9	515.6	704.8	7.7	528.2	528.6	552.5	10.4	530.3	530.4	517.4	10.8
11	**409.3**	839.6	5956.5	25813.0	3631.9	6206.6	430.0	292.3	5.1	430.0	430.0	374.5	5.1	473.6	474.0	133.0	15.8
12	402.3	10937.6	10623.1	28179.0	3632.1	6904.5	412.1	98.8	2.4	412.1	412.1	122.6	2.4	407.7	407.7	97.2	1.3
13	7550.6	10817.1	98046.0	50256.0	3616.3	565.6	9425.8	3608.0	24.8	9346.3	9372.2	3615.6	24.1	9517.7	9596.5	3606.7	27.1
Moy 1-12		**3778.9**			**3223.9**	**2331.1**		**162.7**	**2.5**			**149.6**	**3.0**			**111.4**	**3.3**
Moy 1-13		**4320.3**			**3254.1**	**2195.3**		**427.8**	**4.3**			**416.2**	**4.6**			**380.3**	**5.1**

Notons toutefois que pour l'instance 7, la solution fractionnaire est $SR = 458.2$, tandis que la solution optimale, trouvée par CPLEX en nombres entiers est $Bsup = 458.5$. La solution en nombres entiers obtenue par Exp 1 pour cette instance est $S = 458.8$, soit un Gap_PLNE de 0.1%.

Les expériences Exp 2, Exp 3 et Exp 4 concernent la génération de colonnes avec programmation dynamique pour résoudre le problème esclave (Tableau 11). Le problème maître de l'Exp 2 et l'Exp 3 est un Set Partitioning Problem (cf la Figure 34). La solution en nombres entiers est obtenue par Branch-and-Price pour l'Exp 2, et par résolution en PLNE pour l'Exp 3. Le problème maître de l'Exp 4 correspond à un Set Covering Problem, et la solution en nombres entiers est obtenue par PLNE.

Le Set Partitioning Problem retourne une solution composée d'un sous-ensemble de colonnes où une visite ne peut appartenir qu'à une seule colonne (cf. la section 5.3.1). Le Set Covering Problem retourne un sous-ensemble de colonnes où une visite peut appartenir qu'à plusieurs colonnes (même si dans la pratique, c'est rarement le cas). Si une visite appartient à plusieurs tournées, alors un algorithme glouton permet de ne conserver qu'une seule occurrence (voir la section 5.3.2).

Les deux dernières lignes du Tableau 14 sont les valeurs moyennes pour les instances 1 à 12 et pour les instances 1 à 13, pour chaque expérience. En effet, la dernière instance a tendance à « lisser » les résultats puisqu'elle possède un coût et un temps de résolution beaucoup plus importants que les autres instances.

En moyenne, sur les instances 1 à 12, l'Exp 2 est la meilleure puisque le Gap_PLNE est de 2.5%, alors que pour l'Exp 3 il est de 3.0% et pour l'Exp 4, il est de 3.3%. Toutefois, l'Exp 2 est en moyenne plus longue à s'exécuter que les autres approches : 162.7 secondes contre 149.6 secondes pour l'Exp 3, et 111.4 secondes pour l'Exp 4. L'Exp 3 offre donc un bon compromis entre les deux autres méthodes puisque son Gap_PLNE moyen et son temps moyen d'exécution arrivent en seconde place. Les trois méthodes ont des temps de calcul moyens entre 111.4 secondes et 162.7 secondes, alors que la résolution par PLNE a un temps de calcul moyen de 3778.9 secondes, soit environ 23 à 34 fois plus grand.

D'une manière générale, les résultats montrent que les différentes approches basées sur la génération de colonnes sont capables de trouver des résultats proches des solutions optimales, en un temps de calcul nettement inférieur à la résolution par PLNE. De ce point de vue, l'approche proposée a atteint son objectif qui est de permettre la résolution efficace d'instances de tailles significatives, c'est-à-dire, de fournir des solutions de bonne qualité dans des temps de calcul courts.

6.3. Influence du paramétrage de la GC sur les résultats

Plusieurs paramètres influencent les résultats obtenus par la génération de colonnes, notamment la profondeur de l'arbre du Branch-and-Price : un arbre trop petit ne permettra pas forcément d'atteindre la solution optimale, tandis qu'un arbre trop profond risque de requérir un grand temps de calcul. De même, pour le nombre de labels lors de la programmation dynamique, un faible nombre réduit la possibilité de trouver la solution optimale, tandis qu'un grand nombre ralentit la résolution. Ces deux paramètres font objets des sections, respectivement, 6.3.1 et 6.3.2.

6.3.1. Impact de la profondeur de l'arbre

Cette section vise à étudier l'impact de la profondeur de l'arbre sur la qualité de la solution lorsqu'un Branch-and-Price est utilisé pour obtenir une solution en nombres entiers. L'expérience se déroule dans les mêmes conditions que l'expérience Exp 2 (voir le Tableau 11 ou la Figure 34), c'est-à-dire que le problème maître correspond à un Set Partitioning Problem et le problème esclave est résolu par programmation dynamique. Deux paramétrages de la profondeur de l'arbre sont étudiés : dans le premier cas, la profondeur est limitée à 10 nœuds, tandis que dans le second cas, la profondeur est limitée à 100 nœuds.

Le Tableau 15 présente les résultats de l'expérience Exp 2 avec 10 et 100 nœuds de profondeur dans l'arbre du Branch-and-Price : S est la valeur de la solution et TT est le temps total d'exécution. La dernière colonne du tableau présente le Gap entre les deux exécutions. Ce gap est toujours nul et signifie que les deux exécutions trouvent des solutions de même coût. Toutefois, le temps d'exécution de l'expérience avec 100 nœuds est très légèrement supérieur à l'autre (435.1 secondes contre 427.8 secondes). L'instance 13 est un cas particulier, car la génération de colonnes est limitée à 3600 secondes et n'a pas le temps de s'exécuter entièrement sur cette instance.

Ces résultats, identiques pour les deux exécutions, s'expliquent d'une part par le fait que pour un grand nombre d'instances, la solution entière obtenue à la racine du Branch-and-Price est déjà très proche de la solution optimale (voir le Tableau 14). Et d'autre part, par le fait que la résolution du sous-problème n'est pas optimale.

Tableau 15 Impact de la profondeur de l'arbre

	Exp 2 : Profondeur max 10 nœuds		**Exp 2 : Profondeur max 100 nœuds**		
Instance	S	TT	S	TT	$\boldsymbol{Gap}$ (%)
1	698.0	27.0	698.0	29.0	0.0
2	203.8	8.5	203.8	10.2	0.0
3	887.1	29.1	887.1	31.4	0.0
4	517.8	42.2	517.8	47.1	0.0
5	105.0	1.1	105.0	1.4	0.0
6	928.0	30.4	928.0	36.5	0.0
7	459.3	35.4	459.3	38.6	0.0
8	621.0	668.5	621.0	719.0	0.0
9	140.5	14.9	140.5	14.9	0.0
10	515.6	704.8	515.6	723.7	0.0
11	430.0	292.3	430.0	296.4	0.0
12	412.1	98.8	412.1	100.5	0.0
13	9425.8	3608.0	9425.8	3608.2	0.0
AVG		**427.8**		**435.1**	**0.0**

En conclusion, ces résultats montrent que la profondeur de l'arbre du Branch-and-Price a un faible impact sur la résolution des instances.

6.3.2. Impact du nombre de labels

Cette étude se focalise sur l'impact du nombre de labels pour la résolution du problème esclave par programmation dynamique. L'étude est menée dans les mêmes conditions expérimentales que l'Exp 3 : le problème maître correspond à un Set Partitioning Problem et le problème esclave est résolu par programmation dynamique (voir Tableau 11 ou la Figure 34). Le nombre de labels maximum par nœud lors de la résolution du problème esclave est soit 250, soit 500.

Le Tableau 16 présente les résultats des deux exécutions de l'expérience Exp 3 avec 250 labels et 500 labels. Le Gap, en pourcentages, est le gap des valeurs des solutions entre les deux exécutions. Un gap positif signifie que la solution obtenue avec 500 labels est meilleure que celle obtenue avec 250 labels : c'est notamment le cas pour l'instance 1 ($Gap = 0.2\%$). La valeur S de la solution avec 250 labels est $S = 699.1$, et celle avec 500 labels est $S = 698.0$. Pour l'instance 2, les solutions obtenues sont identiques ($Gap = 0.0\%$), mais le temps d'exécution TT avec 500 labels est environ 2,5 fois plus grand (21.6 secondes contre 8.9 secondes). Trois instances ont un $Gap < 0.0\%$ (instances 3, 8 et 13). Pour les instances 3 et 8, ce phénomène peut être dû au fait que le nombre maximal de 500 labels est atteint sur certains nœuds, et que les labels supprimés diffèrent de ceux qui le sont dans le cas de 250 labels. Pour l'instance 13, la limite du temps d'exécution de la génération de colonnes empêche cette dernière de s'exécuter entièrement. La résolution d'un sous-problème avec 500 labels est plus longue qu'avec 250 labels. La génération de colonnes avec 500 labels effectue moins d'itérations avant d'être stoppée. Le gap moyen entre les deux exécutions est de -0.5%, mais est très impacté par l'instance 13. Le gap moyen sans cette instance est de 0.9%. Le temps moyen, de la génération de colonnes (pour les instances 1 à 12) avec 250 labels, est de 149.6 secondes et avec 500 labels de 402.1 secondes, soit un facteur d'environ 2.7.

Tableau 16 Impact du nombre de labels

	Exp3 : 250 labels			**Exp3 : 500 labels**			
Instance	*SR*	*S*	*TT*	*SR*	*S*	*TT*	***Gap*** (%)
1	698.0	699.1	13.0	697.8	698.0	35.2	0.2
2	203.8	203.8	8.9	203.8	203.8	21.6	0.0
3	887.7	888.6	26.2	888.5	896.0	65.9	-0.8
4	518.2	518.6	10.6	517.6	518.0	46.2	0.1
5	105.0	105.0	1.2	105.0	105.0	3.0	0.0
6	933.7	938.8	13.7	929.5	932.4	34.5	0.7
7	459.3	459.3	35.5	458.3	458.5	121.9	0.2
8	622.1	622.1	622.3	640.8	640.8	1462.3	-3.0
9	141.1	141.1	14.2	125.6	125.6	123.2	11.0
10	528.2	528.6	552.5	523.8	523.8	1614.6	0.9
11	430.0	430.0	374.5	430.0	430.0	742.9	0.0
12	412.1	412.1	122.6	405.5	405.5	554.0	1.6
13	9346.3	9372.2	3615.6	10970.1	11026.8	3671.9	-17.7
AVG 1-12			**149.6**			**402.1**	**0.9**
AVG 1-13			**416.2**			**653.6**	**-0.5**

En conclusion de cette expérience, l'exécution avec 500 labels permet d'améliorer les résultats en moyenne, mais nécessite un temps de calcul beaucoup plus conséquent.

7. Conclusion

Ce chapitre se focalise sur un problème de workforce avec ordonnancement et transport, où ce dernier est explicitement pris en compte. Dans ce type de problème, la fonction objectif se concentre en général sur les coûts opérationnels, et éventuellement sur la satisfaction des clients, mais très rarement sur la qualité de service du point de vue des employés. Le WSRP étudié dans ce chapitre prend en considération la qualité de service aussi bien pour les clients que pour les employés.

Ce chapitre aborde la notion de qualité de service dans les problèmes d'ordonnancement avec transport, sous un angle différent de celui du chapitre 2. Comme pour le chapitre précédent, le problème traité se modélise sous la forme d'un graphe et la prise en compte de la notion de qualité de service aboutit à la définition de solutions qui ne sont pas semi-actives.

Une des contributions de ce chapitre est l'introduction d'une fonction d'évaluation d'une tournée – modélisée sous forme de graphe – pour maximiser la qualité de service de l'employé qui la réalise. Cette fonction est basée sur un algorithme à labels pour résoudre une extension de l'Elementary Shortest Path Problem with Resource Constraints avec qualité de service. Les différentes tournées d'un problème de WSRP sont indépendantes les unes des autres, et la fonction d'évaluation peut donc être appliquée itérativement pour chacune des tournées, afin de trouver la solution globale maximisant la qualité de service.

Ce chapitre propose également la modélisation en PLNE et en PPC du WSRP, ainsi qu'une génération de colonnes, avec le sous-problème résolu par PLNE ou par programmation dynamique. Cette génération de colonnes n'est pas exacte, car la résolution du sous-problème par programmation dynamique est restreinte par le nombre de labels, et la profondeur du Branch-and-Price est également limitée. Toutefois, cette génération de colonnes permet d'obtenir des solutions de bonne qualité en un temps de calcul restreint. Un nouveau jeu d'instances est introduit pour le WSRP.

Le futur objectif des problèmes de type WSRP est d'introduire des contraintes de coordination entre les visites, et cette finalité est abordée dans le chapitre 4. L'ajout de contraintes de coordination modifie la nature du problème, car les tournées ne sont plus indépendantes les unes des autres et l'évaluation d'une tournée est indépendante des autres tournées.

8. Bibliographie

Algethami, H., Landa-Silva, D., 2017. Diversity-based adaptive genetic algorithm for a Workforce Scheduling and Routing Problem, in: Evolutionary Computation (CEC), 2017 IEEE Congress On. IEEE, pp. 1771–1778.

Algethami, H., Landa-Silva, D., 2015. A Study of Genetic Operators for the Workforce Scheduling and Routing Problem. Presented at the MIC 2015: The XI Metaheuristics International Conference, Agadir, Morocco.

Algethami, H., Landa-Silva, D., Martínez-Gavara, A., 2017. Selecting Genetic Operators to Maximise Preference Satisfaction in a Workforce Scheduling and Routing Problem, in: Proceedings of the 6th International Conference on Operations Research and Enterprise Systems. Presented at the ICORES, Porto, Portugal, pp. 416–423. https://doi.org/10.5220/0006203304160423

Algethami, H., Martínez-Gavara, A., Landa-Silva, D., 2018. Adaptive multiple crossover genetic algorithm to solve workforce scheduling and routing problem. Journal of Heuristics 1–40. https://doi.org/10.1007/s10732-018-9385-x

Algethami, H., Pinheiro, R.L., Landa-Silva, D., 2016. A genetic algorithm for a workforce scheduling and routing problem. 2016 IEEE Congress on Evolutionary Computation (CEC) 927–934.

Attia, E.-A., Duquenne, P., Le-Lann, J.-M., 2014. Considering skills evolutions in multi-skilled workforce allocation with flexible working hours. International Journal of Production Research 52, 4548–4573. https://doi.org/10.1080/00207543.2013.877613

Bachouch, R.B., Guinet, A., Hajri-Gabouj, S., 2011. A Decision-Making Tool for Home Health Care Nurses' Planning. Supply Chain Forum: An International Journal 12, 14–20. https://doi.org/10.1080/16258312.2011.11517250

Balinski, M., 2010. Integer programming: Methods, uses, computation, in: 50 Years of Integer Programming 1958-2008. Springer, Berlin, Heidelberg, pp. 133–197.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H., 1998. Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research 46, 316–329. https://doi.org/10.1287/opre.46.3.316

Beldiceanu, N., Contejean, E., 1994. Introducing global constraints in CHIP. Mathematical and Computer Modelling 20, 97–123. https://doi.org/10.1016/0895-7177(94)90127-9

Bockmayr, A., Hooker, J.N., 2005. Constraint programming, in: Handbooks in Operations Research and Management Science. pp. 559–600.

Bourreau, É., Gondran, M., Lacomme, P., 2019a. Efficient Constraint Programming Approaches for routing problem: a case study for the VRP, in: 7th Vehicle Routing and Logistics Optimization (VeRoLog). Seville, Spain.

Bourreau, É., Gondran, M., Lacomme, P., Vinot, M., 2020. Programmation Par Contraintes : démarches de modélisation pour des problèmes d'optimisation. Ellipses.

Bourreau, É., Gondran, M., Lacomme, P., Vinot, M., 2019b. De la programmation linéaire à la programmation par contraintes. Ellipses.

Bredström, D., Rönnqvist, M., 2007. A Branch and Price Algorithm for the Combined Vehicle Routing and Scheduling Problem With Synchronization Constraints. NHH Dept. of Finance & Management Science Discussion Paper, (2007/7). https://doi.org/10.2139/ssrn.971726

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R., 2016. Workforce scheduling and routing problems: literature survey and computational study. Annals of Operations Research 239, 39–67.

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R., 2014. Computational Study for Workforce Scheduling and Routing Problems: ICORES 2014 434–444.

Chvátal, V., 1979. A Greedy Heuristic for the Set-Covering Problem. Mathematics of OR 4, 233–235. https://doi.org/10.1287/moor.4.3.233

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579–594. https://doi.org/10.1016/S0191-2615(02)00045-0

Cordeau, J.-F., Laporte, G., Pasin, F., Ropke, S., 2010. Scheduling technicians and tasks in a telecommunications company. Journal of Scheduling 13, 393–409. https://doi.org/10.1007/s10951-010-0188-7

Current, J., Marsh, M., 1993. Multiobjective transportation network design and routing problems: Taxonomy and annotation. European Journal of Operational Research 65, 4–19.

Cymer, R., 2012. Dulmage-Mendelsohn Canonical Decomposition as a generic pruning technique. Constraints 17, 234–272. https://doi.org/10.1007/s10601-012-9120-4

de Armas, J., Lalla-Ruiz, E., Expósito-Izquierdo, C., Landa-Silva, D., Melián-Batista, B., 2015. A hybrid GRASP-VNS for ship routing and scheduling problem with discretized time windows. Engineering Applications of Artificial Intelligence 45, 350–360. https://doi.org/10.1016/j.engappai.2015.07.013

Desaulniers, G., 2010. Branch-and-Price-and-Cut for the Split-Delivery Vehicle Routing Problem with Time Windows. Operations Research 58, 179–192. https://doi.org/10.1287/opre.1090.0713

Desaulniers, G., Desrosiers, J., Solomon, M.M., 2005. Column generation, Springer Science & Business Media.

Desaulniers, G., Desrosiers, J., Solomon, M.M., 2001. Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems, in: Essays and Surveys in Metaheuristics. Springer US, Boston, MA, pp. 309–324. https://doi.org/10.1007/978-1-4615-1507-4_14

Dongarra, J.J., 2014. Performance of various computers using standard linear equations software. Report CS-89-85, University of Manchester.

Dror, M., 1994. Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. Operations Research 42, 977–978. https://doi.org/10.1287/opre.42.5.977

Eveborn, P., Flisberg, P., Rönnqvist, M., 2006. Laps Care—an operational system for staff planning of home care. European Journal of Operational Research 171, 962–976. https://doi.org/10.1016/j.ejor.2005.01.011

Farham, M.S., Süral, H., Iyigun, C., 2018. A column generation approach for the location-routing problem with time windows. Computers & Operations Research 90, 249–263. https://doi.org/10.1016/j.cor.2017.09.010

Feillet, D., 2010. A tutorial on column generation and branch-and-price for vehicle routing problems. 4or 8, 407–424.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44, 216–229.

Fikar, C., Hirsch, P., 2017. Home health care routing and scheduling: A review. Computers & Operations Research 77, 86–95. https://doi.org/10.1016/j.cor.2016.07.019

Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., Soumis, F., 2019. Column generation for vehicle routing problems with multiple synchronization constraints. European Journal of Operational Research 272, 699–711. https://doi.org/10.1016/j.ejor.2018.06.046

Gacias, B., Artigues, C., Lopez, P., 2010. Parallel machine scheduling with precedence constraints and setup times. Computers & Operations Research 37, 2141–2151. https://doi.org/10.1016/j.cor.2010.03.003

Garaix, T., Gondran, M., Lacomme, P., Landa-Silva, D., Tchernev, N., 2018a. Decomposition methods for the workforce scheduling and routing problem, in: 29th European Conference on Operational Research (EURO). Valencia, Spain.

Garaix, T., Gondran, M., Lacomme, P., Mura, E., Tchernev, N., 2018b. Workforce Scheduling Linear Programming Formulation. IFAC-PapersOnLine 51, 264–269. https://doi.org/10.1016/j.ifacol.2018.08.289

Garaix, T., Gondran, M., Lacomme, P., Mura, E., Tchernev, N., 2018c. Workforce Scheduling Resolution based on a column generation scheme. Presented at the 19ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Lorient, France.

Garfinkel, R.S., Nemhauser, G.L., 1969. The Set-Partitioning Problem: Set Covering with Equality Constraints. Operations Research 17, 848–856. https://doi.org/10.1287/opre.17.5.848

Gedik, R., Kalathia, D., Egilmez, G., Kirac, E., 2018. A constraint programming approach for solving unrelated parallel machine scheduling problem. Computers & Industrial Engineering 121, 139–149. https://doi.org/10.1016/j.cie.2018.05.014

Gérard, M., Clautiaux, F., Sadykov, R., 2016. Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce. European Journal of Operational Research 252, 1019–1030. https://doi.org/10.1016/j.ejor.2016.01.036

Goel, A., Meisel, F., 2013. Workforce routing and scheduling for electricity network maintenance with downtime minimization. European Journal of Operational Research 231, 210–228. https://doi.org/10.1016/j.ejor.2013.05.021

Gomes, R.A.M., Toffolo, T.A.M., Santos, H.G., 2017. Variable neighborhood search accelerated column generation for the nurse rostering problem. Electronic Notes in Discrete Mathematics 58, 31–38. https://doi.org/10.1016/j.endm.2017.03.005

Gondran, M., Huguet, M.-J., Lacomme, P., Quilliot, A., Tchernev, N., 2018. A Dial-a-Ride evaluation for solving the job-shop with routing considerations. Engineering Applications of Artificial Intelligence 74, 70–89. https://doi.org/10.1016/j.engappai.2018.05.010

Haas, C.T., Rodriguez, A.M., Glover, R., Goodrum, P.M., 2001. Implementing a multiskilled workforce. Construction Management and Economics 19, 633–641. https://doi.org/10.1080/01446190110050936

He, F., Qu, R., 2012. A constraint programming based column generation approach to nurse rostering problems. Computers & Operations Research 39, 3331–3343. https://doi.org/10.1016/j.cor.2012.04.018

Hernández, B.M., 2008. The systematic generation of channelled models in constraint satisfaction. PhD thesis, University of York, Department of Computer Science.

Hojabri, H., Gendreau, M., Potvin, J.-Y., Rousseau, L.-M., 2018. Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. Computers & Operations Research 92, 87–97. https://doi.org/10.1016/j.cor.2017.11.011

Janacek, J., Kohani, M., Koniorczyk, M., Marton, P., 2017. Optimization of periodic crew schedules with application of column generation method. Transportation Research Part C: Emerging Technologies 83, 165–178. https://doi.org/10.1016/j.trc.2017.07.008

Karlsson, J., Rönnqvist, M., Bergström, J., 2004. An optimization model for annual harvest planning. Canadian Journal of Forest Research 34, 1747–1754.

Karp, R., 1972. Reducibility among Combinatorial Problems. Complexity of computer computations, Springer, Boston, MA 85–103.

Laesanklang, W., Landa-Silva, D., 2017. Decomposition techniques with mixed integer programming and heuristics for home healthcare planning. Annals of Operations Research 256, 93–127. https://doi.org/10.1007/s10479-016-2352-8

Laesanklang, W., Landa-Silva, D., Arturo Castillo Salazar, J., 2015a. Mixed Integer Programming with Decomposition to Solve a Workforce Scheduling and Routing Problem, in: Proceedings of the 5th International Conference on Operations Research and Enterprise Systems. Presented at the ICORES, Lisbon, Portugal, pp. 283–293. https://doi.org/10.5220/0005223602830293

Laesanklang, W., Landa-Silva, D., Castillo-Salazar, J.A., 2016a. An investigation of heuristic decomposition to tackle workforce scheduling and routing with time-dependent activities constraints, in: International Conference on Operations Research and Enterprise Systems. Springer, pp. 239–260.

Laesanklang, W., Landa-Silva, D., Castillo-Salazar, J.A., 2016b. Mixed Integer Programming with Decomposition for Workforce Scheduling and Routing with Time-dependent Activities Constraints, in: Proceedings of 5th the International Conference on

Operations Research and Enterprise Systems. Presented at the ICORES, Rome, Italy, pp. 330–339. https://doi.org/10.5220/0005757503300339

Laesanklang, W., Pinheiro, R.L., Algethami, H., Landa-Silva, D., 2015b. Extended decomposition for mixed integer programming to solve a workforce scheduling and routing problem, in: International Conference on Operations Research and Enterprise Systems. Springer, pp. 191–211.

Laurière, J.L., 1978. A Language and a Program for Stating and Solving Combinatorial Problems Artificial Intelligence. Artificial intelligence 10, 29–127.

Maenhout, B., Vanhoucke, M., 2009. The impact of incorporating nurse-specific characteristics in a cyclical scheduling approach. Journal of the Operational Research Society 60, 1683–1698. https://doi.org/10.1057/jors.2008.131

Mathlouthi, I., Gendreau, M., Potvin, J.-Y., 2018. Mixed integer linear programming for a multi-attribute technician routing and scheduling problem. INFOR: Information Systems and Operational Research 56, 33–49. https://doi.org/10.1080/03155986.2017.1335047

Misir, M., Verbeeck, K., De Causmaecker, P., Berghe, G.V., 2010. Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem, in: Evolutionary Computation (CEC), 2010 IEEE Congress On. IEEE, pp. 1–8.

Mundschenk, M., Drexl, A., 2007. Workforce planning in the printing industry. International Journal of Production Research 45, 4849–4872. https://doi.org/10.1080/00207540600813238

Novas, J.M., Henning, G.P., 2014. Integrated scheduling of resource-constrained flexible manufacturing systems using constraint programming. Expert Systems with Applications 41, 2286–2299. https://doi.org/10.1016/j.eswa.2013.09.026

Paraskevopoulos, D.C., Laporte, G., Repoussis, P.P., Tarantilis, C.D., 2017. Resource constrained routing and scheduling: Review and research prospects. European Journal of Operational Research 263, 737–754. https://doi.org/10.1016/j.ejor.2017.05.035

Pesant, G., Gendreau, M., Potvin, J.-Y., Rousseau, J.-M., 1998. An Exact Constraint Logic Programming Algorithm for the Traveling Salesman Problem with Time Windows. Transportation Science 32, 12–29. https://doi.org/10.1287/trsc.32.1.12

Pinheiro, R.L., Landa-Silva, D., Atkin, J., Pinheiro, R.L., Landa-Silva, D., Atkin, J., 2016. A Variable Neighbourhood Search for the Workforce Scheduling and Routing Problem, in: du Plessis, M.C., Snášel, V., Muda, A.K. (Eds.), Advances in Nature and Biologically Inspired Computing. Springer, Cham, pp. 247–259. https://doi.org/10.1007/978-3-319-27400-3_22

Pour, S.M., Drake, J.H., Ejlertsen, L.S., Rasmussen, K.M., Burke, E.K., 2018. A hybrid Constraint Programming/Mixed Integer Programming framework for the preventive signaling maintenance crew scheduling problem. European Journal of Operational Research 269, 341–352. https://doi.org/10.1016/j.ejor.2017.08.033

Régin, J.-C., 2004. HdR : Modélisation et contraintes globales en programmation par contraintes. Université de Nice.

Régin, J.-C., 1994. A filtering algorithm for constraints of difference in CSPs. Proceedings AAAI-94 362–367.

Rossi, F., van Beek, P., Walsh, T., 2006. Handbook of Constraint Programming, Elsevier.

Rousseau, L.-M., Gendreau, M., Pesant, G., 2013. The Synchronized Dynamic Vehicle Dispatching Problem. INFOR: Information Systems and Operational Research 51, 76–83. https://doi.org/10.3138/infor.51.2.76

Saadat, M., Tan, M.C.L., Owliya, M., Jules, G., 2013. Challenges and trends in the allocation of the workforce in manufacturing shop floors. International Journal of Production Research 51, 1024–1036. https://doi.org/10.1080/00207543.2012.662603

Topaloglu, S., Ozkarahan, I., 2011. A constraint programming-based solution approach for medical resident scheduling problems. Computers & Operations Research 38, 246–255. https://doi.org/10.1016/j.cor.2010.04.018

Wang, T., Meskens, N., Duvivier, D., 2015. Scheduling operating theatres: Mixed integer programming vs. constraint programming. European Journal of Operational Research 247, 401–413. https://doi.org/10.1016/j.ejor.2015.06.008

Wirojanagud, P., Gel, E.S., Fowler, J.W., Cardy, R., 2007. Modelling inherent worker differences for workforce planning. International Journal of Production Research 45, 525–553. https://doi.org/10.1080/00207540600792242

Wongmongkolrit, S., Rassameethes, B., 2010. An algorithm of workforce scheduling for maintenance optimization. International Journal of Management Science and Engineering Management 5, 163–169. https://doi.org/10.1080/17509653.2010.10671104

Zhao, Z., Li, X., 2014. Scheduling elective surgeries with sequence-dependent setup times to multiple operating rooms using constraint programming. Operations Research for Health Care 3, 160–167. https://doi.org/10.1016/j.orhc.2014.05.003

Chapitre IV : Generalised Workforce Scheduling and Routing Problem

Ce chapitre introduit un nouveau problème : le Generalised Workforce Scheduling and Routing Problem – GWSRP – (Bourreau et al., 2019a), qui est une extension du Workforce Scheduling and Routing Problem (WSRP) abordé au chapitre 3.

Le GWSRP prend simultanément en considération un critère (similaire au WSRP) de qualité de service pour les clients et les employés, et des contraintes de coordination issues des problèmes de production et des problèmes de transport. Ces contraintes portent sur les dates des visites. Les contraintes de coordination modifient la nature du problème puisque les visites et les tournées des employés ne sont plus indépendantes les unes des autres : les contraintes de coordination interconnectent les itinéraires des différents employés. Un modèle PPC permet l'évaluation d'un ensemble de tournées pour afin d'obtenir obtenir les dates de début maximisant la qualité de service des employés et respectant toutes les contraintes de coordination.

Ce chapitre présente et formalise les neuf contraintes de coordination prises en considération dans le GWSRP. Par ailleurs, ce chapitre propose une approche – Constraint-Programming based Decomposition Method (CPDM) – composée de trois phases pour la résolution du GWSRP. La première phase utilise une génération de colonnes et consiste à construire un ensemble de tournées sans prendre en compte de contraintes de coordination. Grâce à un modèle PLNE, la seconde phase permet la sélection d'un sous-ensemble de tournées, dans laquelle les contraintes de coordination sont insérées itérativement lors de la troisième phase. Cette dernière phase est réalisée par un modèle PPC. Cette approche est testée sur un nouveau jeu d'instances dédié au GWSRP.

1. Introduction

Le Generalised Workforce Scheduling and Routing Problem – GWSRP – (Bourreau et al., 2019a) est une généralisation du WSRP présenté au chapitre 3. Le GWSRP prend en compte des contraintes de coordination entre les visites. Une partie des contraintes de coordination traitée dans ce chapitre se retrouve également dans la littérature sous le nom de contraintes temporelles. Le GWSRP est un problème d'ordonnancement avec tournées de véhicules et contraintes de coordination, où les employés sont non-identiques. Les deux principales difficultés du GWSRP résident dans la définition de la qualité de service pour les employés qui dépend des dates de visites, et dans la prise en compte de contraintes de coordination qui interconnectent les tournées. La qualité de service (QoS) des employés du GWSRP est identique à celle du WSRP, traitée dans le chapitre 3.

L'importance de contraintes de coordination vient de nombreuses applications réelles issues des secteurs industriels de production et/ou maintenance (Chankov et al., 2018; Parragh and Doerner, 2018), des secteurs aéroportuaires (Fink et al., 2019), ou encore des services à domicile (Rasmussen et al., 2012; Fikar and Hirsch, 2017). Il est fréquent, dans le cadre de soins à domicile, que deux aides-soignants soient nécessaires pour transporter une personne ; ou qu'un certain délai soit respecté entre l'injection de deux piqûres par une

infirmière ; ou qu'un kinésithérapeute et un médecin doivent visiter un patient, et évidemment dans des horaires distincts, afin de pouvoir promulguer les soins dans les meilleures conditions. (Bachouch et al., 2011) soulignent l'importance de la coordination des soins médicaux et paramédicaux puisque les soignants et aides-soignants, les infirmières, les services médicaux et sociaux font partie d'une chaîne logistique spécialisée en soins. Ces problématiques de coopération et de coordination apparaissent également dans les chaînes logistiques industrielles, notamment les problèmes de production avec transport. Des problèmes plus théoriques comme le VRPMS – VRP with Multiple Synchronization constraints – (Hojabri et al., 2018) ou le VRPTWSyn – Vehicle Routing and Scheduling Problem with Time Windows and Synchronized visits – (Bredström and Rönnqvist, 2008; Afifi et al., 2016; Liu et al., 2018) découlent de ces préoccupations. Un état de l'art plus pointu de ces problèmes est présenté à la section 1.1.

Une des contributions de ce chapitre est l'introduction du GWSRP et la formalisation des différentes contraintes de coordination. Une autre contribution porte sur un schéma itératif de résolution combinant une Génération de Colonnes (GC) pour obtenir des tournées sans coordination et un modèle de Programmation Par Contraintes (PPC) pour coordonner les visites.

Ce chapitre contient cinq parties : la première est une introduction du GWSRP et comprend également un état de l'art. La seconde partie concerne la définition formelle des contraintes de coordination. La troisième partie présente les difficultés liées à l'évaluation d'un graphe comprenant des contraintes de coordination. La quatrième partie se focalise sur une approche de résolution. La cinquième partie est une évaluation numérique des performances de l'approche proposée.

1.1. Problèmes de tournées de véhicules avec coordination

Le problème du GWSRP se rapproche des problèmes de type VRP avec des contraintes spécifiques. La littérature du VRP est très riche et une revue exhaustive de celle-ci est impossible. Les références les plus importantes de ce domaine incluent notamment les livres de (Toth and Vigo, 2002, 2014; Gendreau et al., 2008), ainsi que les états de l'art de (Braekers et al., 2016), pour le VRP, et (Gendreau and Tarantilis, 2010) plus spécifiquement pour le VRP avec fenêtre de temps. Cette section ne s'intéresse qu'aux articles présentant des problèmes de tournées de véhicules avec des contraintes de coordination.

(Drexl, 2012) présente un état de l'art des problèmes de VRP avec des contraintes de synchronisation au sens très large, il recense une centaine d'articles. Ils proposent cinq types de synchronisation :

- Synchronisation des tâches : la synchronisation d'une tâche (qui est la terminologie utilisée dans l'article, mais qui peut être également vue comme une visite ou une opération) fait référence au fait qu'il faut décider quel véhicule/employé réalise la tâche. La synchronisation des tâches correspond donc à une affectation.
- Synchronisation des mouvements : elle prévaut lorsque deux types de véhicules doivent se déplacer d'un endroit à l'autre et en même temps, car l'un des véhicules est incapable de se déplacer seul (Hollis et al., 2006; Xiang et al., 2006).
- Synchronisation des chargements : elle garantit qu'aucun chargement n'est perdu et que la bonne quantité de chargement est livrée à un client. L'exemple classique de ce type de synchronisation est le Split-Delivery VRP, où chaque véhicule délivre une partie de la demande (Golden et al., 2008; Desaulniers, 2010) ;

- Synchronisation des ressources : la ressource utilisée est limitée et mutualisée. Ce type de synchronisation correspond, entre autres, au problème de Resource-Constrained Shortest Path (Feillet et al., 2004; Desaulniers et al., 2005).
- Synchronisation des opérations : la synchronisation d'opérations correspond à la coordination de différents véhicules pour réaliser des tâches qui peuvent être soit au même endroit, soit à des endroits différents. (Drexl, 2012) estime qu'il y a opération de synchronisation si la planification d'une opération affectée à un véhicule a un impact sur les tournées des autres véhicules. Ces synchronisations peuvent être spatiales (Nguyen et al., 2010), ou temporelles (Bredström and Rönnqvist, 2008; Dohn et al., 2009; Rasmussen et al., 2012). C'est ce type de synchronisations temporelles qui est étudié dans ce chapitre sous le nom de « contraintes de coordination », car de nombreux articles utilisent déjà le nom de contrainte de synchronisation pour définir une contrainte imposant que deux visites débutent simultanément. Les contraintes de coordination présentées dans ce chapitre incluent les contraintes de synchronisation et proposent également d'autres contraintes.

L'état de l'art de ce chapitre se concentre sur les articles qui traitent de la synchronisation des opérations (d'après la définition de (Drexl, 2012)) et qui ont eu un grand impact sur la recherche dans ce domaine, mais aussi sur les articles les plus récents.

(Eveborn et al., 2006) font partie des premiers qui ont étudié le Home Health Care problem (HHC) avec des contraintes de synchronisation : plusieurs membres du personnel soignant doivent être réunis pour commencer ensemble une visite. Ils mettent en évidence la difficulté des contraintes de synchronisation, car celles-ci créent des relations entre les tournées des différents employés. Ils proposent une heuristique où les tournées peuvent violer les contraintes de fenêtre de temps et de synchronisation, mais ces violations ont un coût de pénalité.

(Bredström and Rönnqvist, 2008) présentent un Vehicle Routing and Scheduling Problem with Time Windows with temporal precedence and synchronization constraints (VRSPTW-TC, renommé VRPTWSyn – Vehicle Routing and Scheduling Problem with Time Windows and Synchronized visits – dans des articles plus contemporains) qui est un problème intégré de tournées de véhicules et d'ordonnancement avec des fenêtres de temps, prenant en compte des contraintes de précédence temporelle et des contraintes de synchronisation. (Bredström and Rönnqvist, 2008) présentent un modèle mathématique et une heuristique basée sur une relaxation de leur PLNE. Ils introduisent également un jeu d'instances qui est utilisé dans de nombreuses publications récentes. Les contraintes de synchronisation surviennent lorsqu'il faut plusieurs employés pour effectuer une visite. Dans ce cas, la date de visite de chaque employé doit être la même (la durée de service de la visite est identique pour les deux employés). Les contraintes de précédence modélisent la nécessitée de faire différentes visites chez le même client dans un ordre donné, avec un délai minimal entre ces visites. Les contraintes temporelles correspondent à des time-lags minimaux. (Bredström and Rönnqvist, 2008) démontrent l'intérêt du VRSPTW-TC avec des contraintes de synchronisation et temporelles pour deux applications :

- Le Home Health Care Problem (Eveborn et al., 2006; Fikar and Hirsch, 2017) où certains patients requièrent que deux employés soient présents simultanément pour donner un bain par exemple, et où certains clients doivent recevoir une injection d'un médicament avant et après le repas par une infirmière qualifiée.

- Les problèmes de Forest Operations (Karlsson et al., 2004) où il y a deux types de camions, dont certains ne possèdent pas de grue pour charger les arbres et nécessitent donc l'aide d'un autre camion avec grue.

(Bredström and Rönnqvist, 2007) proposent un Branch-and-Price pour le VRSPTW-TC, introduit par (Bredström and Rönnqvist, 2008) – la chronologie (surprenante) des dates est due aux délais d'édition. Au cours de la génération de colonnes, les contraintes de synchronisation sont relâchées et la difficulté réside dans le Branch-and-Price afin de satisfaire les contraintes de synchronisation. (Bredström and Rönnqvist, 2007) proposent un Branch-and-Price où les branchements sont effectués sur les fenêtres de temps des visites.

(Dohn et al., 2011) s'intéressent au VRP-TW avec des contraintes temporelles de synchronisation entre les visites. Ils proposent un Branch-and-Price avec quatre formulations différentes du problème maître. Les branchements du Branch-and-Price s'effectuent sur les fenêtres de temps des visites afin d'obtenir une solution respectant les contraintes de coordination. Ils définissent cinq types de contraintes de coordination :

- Synchronisation : les visites doivent commencer à la même date.
- Overlap : les deux visites doivent avoir une durée minimale en commun.
- Time-lag minimal : un délai minimal entre deux visites doit être respecté, l'ordre des visites est une donnée du problème.
- Time-lag maximal : un délai maximal entre deux visites doit être respecté, l'ordre des visites est une donnée du problème.
- Time-lag minimal + Time-lag maximal : la visite doit respecter un time-lag minimal et un time-lag maximal.

Contrairement au GWSRP, (Dohn et al., 2011) ne prennent pas en compte des critères de qualité de service, et les employés (ou véhicules dans leur cas) sont tous identiques. Les différences entre les problèmes sont présentées en détail à la section 1.1.

(Rasmussen et al., 2012) s'intéressent au Home Care Crew Scheduling problem (HCCS ou HHC pour Home Health Care problem) qui est une généralisation du VRP-TW introduit par (Dohn et al., 2011). Ils proposent une méthode exacte avec un Branch-and-Price. Les deux articles diffèrent car (Rasmussen et al., 2012) prend en compte un critère de préférence entre les patients et le personnel soignant. Le HHC ne prend pas en considération un critère de qualité de service (QoS) pour les employés dans leur fonction objectif, ni la notion de région de préférence, et le HHC a un critère de priorité des visites. Les différences entre le HHC et le GWSRP sont présentées en détail à la section 1.2.

(Rousseau et al., 2013) étudient le Synchronized Dynamic Vehicle Dispatching Problem (SDVDP), qui est une extension du Real Time Vehicle Dispatching Problems (RTVDP) avec des contraintes de synchronisation. Les clients apparaissent dynamiquement dans le problème. Il existe deux types de clients : les réguliers et les spéciaux. Un client régulier doit être servi uniquement par un véhicule régulier, tandis qu'un client spécial doit être visité par un véhicule régulier et, un plus, un véhicule spécial. La date de visite du véhicule spécial doit être comprise dans une fenêtre de temps qui dépendant de la date de visite du véhicule régulier. (Rousseau et al., 2013) résolvent le SDVDP avec une métaheuristique hybride couplée à un modèle de programmation par contraintes. Les tournées sont construites itérativement par insertion des visites grâce au solveur de programmation par contraintes. Cette dernière n'a pas le droit de modifier les tournées déjà existantes (c'est-à-

dire, intervertir l'ordre des visites, déplacer une visite dans une autre tournée, etc.), mais doit uniquement placer la nouvelle visite.

(Labadie et al., 2014) traitent du VRPTWSyn qui est un problème de tournées de véhicules avec des fenêtres de temps et visites synchronisées (Vehicle Routing Problem with Time Windows and Synchronized visits – VRPTWSyn). Ce problème, sous un autre nom, est identique au VRSPTW-TC de (Bredström and Rönnqvist, 2008, 2007). (Labadie et al., 2014) proposent un modèle linéaire ainsi qu'une heuristique. Cette dernière est composée d'une heuristique constructive, pour obtenir une première solution, suivit d'un Iterated Local Search. Leur heuristique est basée sur un vecteur de permutation des clients qui est décodé pour obtenir une solution faisable. Tous les vecteurs de permutation n'amènent pas à une solution réalisable.

(Mankowska et al., 2014) s'intéressent à un Home Health Care Routing and Scheduling Problem with Interdependent Service qui est une généralisation du HHC avec des contraintes de synchronisation et de time-lag (maximal et minimal). Ils proposent un modèle mathématique ainsi qu'une heuristique. Ils utilisent une représentation des solutions en forme de matrice où chaque ligne de la matrice modélise une tournée constituée de visites. Ils insèrent itérativement les clients dans la matrice, et si un client nécessite plusieurs visites liées par des contraintes de coordination, alors les visites sont simultanément insérées dans la matrice.

(Afifi et al., 2016) étudient le VRPTWSyn introduit par (Bredström and Rönnqvist, 2008) et proposent un Simulated Annealing based Algorithm (SA-ILS). Afin de converger, ils s'autorisent à avoir des routes qui violent les fenêtres de temps des visites, mais qui ont un coût de pénalité. Leur recherche locale utilise des mouvements classiques des problèmes de VRP, notamment les opérateurs 2-opt (Potvin et al., 1996), or-opt (Solomon and Desrosiers, 1988), remplacement, et single-move. L'opérateur choisi lors de la recherche locale est aléatoirement sélectionné. Si l'opérateur échoue dans l'obtention d'une solution meilleure, alors celui-ci est supprimé de la liste des opérateurs potentiels. Les travaux de (Afifi et al., 2013) sont une version préliminaire de ceux publiés en revue (Afifi et al., 2016).

(Haddadene et al., 2016) présentent un GRASP×ILS pour le VRPTWSyn. Ils proposent également un nouveau modèle mathématique qui permet d'éviter la duplication des nœuds modélisant les visites qui nécessitent plusieurs employés. Cette nouvelle formulation permet de conserver un graphe de taille raisonnable. La force de leur métaheuristique repose sur une recherche locale qui, d'une part, insère une visite dans une tournée, et d'autre part calcule en temps constant si l'insertion de la visite permet de respecter les fenêtres de temps, les contraintes de synchronisation et de précédence.

(Hojabri et al., 2018) étudient un VRP with Multiple Synchronization constraints (VRPMS) qui permet de modéliser des problèmes où un véhicule décharge du matériel, tandis qu'un autre véhicule vient l'installer. Dans le VRPMS, il existe deux types de clients (les clients réguliers et les clients spéciaux) ; et deux types de véhicules (les véhicules réguliers et les véhicules spéciaux). Un client régulier doit être visité par un véhicule régulier, donnant lieu à une visite régulière. Un client spécial doit être visité par un véhicule régulier et par un véhicule spécial, donnant lieu à deux visites : une régulière et une spéciale. Dans ce cas, les visites doivent être synchronisées. Une visite régulière (quel que soit le type de client) a une fenêtre de temps $[E_i ; L_i]$. La visite spéciale a une fenêtre de temps $[E_i^* ; L_i^*]$, qui est dépend de la date de début st_i de la visite régulière i associée au même client c : $[E_i^* ; L_i^*] = [st_i - \delta ; st_i + \gamma]$, où δ et γ sont des données du problème. Si $\delta = \gamma = 0$, alors les visites

doivent respecter une contrainte de type synchronisation (contrainte de visites simultanées). Si $\delta > 0$ alors il s'agit d'une contrainte de type time-lag maximal. Et si $\delta > 0$ et $\gamma > 0$, alors il s'agit d'une contrainte où deux visites doivent s'exécuter en même temps pendant une certaine durée. (Hojabri et al., 2018) proposent un Large Neighborhood Search (LNS) couplé à de la Programmation Par Contraintes (PPC). Une heuristique construit une solution initiale, permettant d'obtenir un ensemble de tournées C. Certains clients sont supprimés des tournées, formant ainsi un ensemble de tournées C', et le module PPC essaie de les insérer dans les tournées C' de manière à respecter les fenêtres de temps et les contraintes de coordination.

(Liu et al., 2018) présentent un « Adaptive Large Neighborhood Search heuristic » (ALNS) pour le VRPTWSyn (ou VRSPTW-TC suivant la terminologie utilisée par (Bredström and Rönnqvist, 2008, 2007)). (Liu et al., 2018) proposent une heuristique qui créée une solution initiale, puis qui supprime un certain nombre de clients qu'elle essaie de réinsérer à de meilleures places. La méthode permet de générer des routes irréalisables, d'un point de vue des fenêtres de temps, et ces routes sont associées à une pénalité mesurant l'infaisabilité dans la fonction objectif. Ils proposent également une formulation en PLNE. Ils n'ont que des contraintes de coordination dite de synchronisation, où les visites doivent débuter en même temps.

(Masmoudi and Cheikhrouhou, 2018) étudient l'Heterogeneous Vehicle Routing Problem with Synchronisation visit and Break (HVRPSB), qui est une extension VRPTWSyn avec une flotte de véhicules hétérogènes, et qui prend en compte des pauses obligatoires pour les chauffeurs. Ils proposent un modèle linéaire pour leur problème ainsi qu'un Adaptive Large Neighborhood Search (ALNS).

(Parragh and Doerner, 2018) étudient deux problèmes : le VRPTWSyn et le Service Technician Routing and Scheduling Problem (STRSP). Tous deux prennent en compte des contraintes de synchronisation. Ils proposent un Adaptive Large Neighborhood Search (ALNS) pour résoudre les deux problèmes, avec trois approches pour synchroniser les visites. La première approche réduit les fenêtres de temps des visites à synchroniser à une date unique, garantissant ainsi que la date de début des visites est identique pour les différents employés. La seconde approche consiste à essayer de synchroniser toutes les visites par un PLNE. La troisième approche synchronise directement les visites lors de leur insertion dans une tournée.

(Fink et al., 2019) présentent une génération de colonnes pour un problème de tournées de véhicules, avec de multiples contraintes de synchronisation. Ce problème est appelé : Abstract Vehicle Routing Problem with Worker and Vehicle Synchronization (AVRPWVS). Ils proposent deux modèles de multi-commodity flow basés sur un « time-space network » (un nœud pour la date de début et un nœud pour la date de départ), et incluant les cinq types de synchronisation introduites par (Drexl, 2012). Dans l'article de (Fink et al., 2019), ces contraintes de coordination concernent le chargement, le déchargement et le ravitaillement d'avions dans un aéroport. La durée des opérations varie en fonction du nombre d'employés affectés. Pour respecter les règles de sécurité et pour augmenter l'utilisation des travailleurs, un employé ne doit pas attendre plus longtemps qu'un certain temps donné sur le tarmac (par exemple 10 minutes), à la fois avant l'exécution de la tâche et après celle-ci. Si le travailleur n'a pas été affecté à une nouvelle tâche avant que le temps d'attente maximal de l'employé soit atteint, il doit alors retourner au dépôt. Les employés utilisent des véhicules pour se déplacer d'une tâche à l'autre. Les véhicules sont mutualisés et une capacité de transport (exprimée en termes de nombre d'employés déplaçables simultanément) est imposée. La

solution recherchée est au plus tôt afin de réduire le risque de prendre du retard. Si une tâche est légèrement retardée, il est probable que ce retard n'influence que peu la planification des autres tâches, car la solution au plus tôt contient généralement des temps d'attente conséquents.

Le Tableau 1 présente une synthèse des articles abordés dans cet état de l'art. Pour chaque article le problème traité est rappelé ainsi que la méthode utilisée : GWSRP est le Generalised Workforce Scheduling and Routing Problem ; VRPTWSyn (ou VRPTW-TC) est le Vehicle Routing and Scheduling Problem with Time Windows with temporal precedence and synchronization constraints introduit par (Bredström and Rönnqvist, 2008) ; HHC correspond aux Home Health Care problems ; VRPMS est un VRP with Multiple Synchronization constraints introduit par (Rousseau et al., 2013; Hojabri et al., 2018) ; HVRPSB est l'Heterogeneous Vehicle Routing Problem with Synchronisation visit and Break (Masmoudi and Cheikhrouhou, 2018) ; STRSP est le Service Technician Routing and Scheduling Problem de (Parragh and Doerner, 2018) ; et enfin AVRPWVS signifie Abstract Vehicle Routing Problem with Worker and Vehicle Synchronization (Fink et al., 2019). Parmi les méthodes recensées, il y a la PLNE, les heuristiques, la Programmation Par Contraintes (PPC), les couplages heuristique-PPC, et les méthodes à base de Génération de Colonnes (GC) et Branch-and-Price (B&P).

La majorité des articles se focalisent sur le VRPTWSyn ou sur le HHC avec des contraintes de coordination (les articles qui traitent du HHC sans contrainte de coordination ne sont pas présentés dans ce manuscrit). Par ailleurs, la plus grande partie des articles proposent une approche heuristique, et deux articles présentent une approche heuristique couplée à de la PPC (Rousseau et al., 2013; Hojabri et al., 2018). Quelques articles proposent également des méthodes basées sur la génération de colonnes (Bredström and Rönnqvist, 2007; Dohn et al., 2011; Rasmussen et al., 2012; Fink et al., 2019). Aucun article ne propose une résolution uniquement par PPC.

Tableau 1 Résumé des problèmes et des méthodes publiés dans la littérature

Article	**Problème abordé**					**Méthode utilisée**				
	GWSRP	**VRPTWSyn/ VRPTW-TC**	**HCC**	**VRPMS**	**Autres**	**PLNE**	**Heuristique**	**PPC**	**Heuristique + PPC**	**GC / B&P**
(Eveborn et al., 2006)			X				X			
(Bredström and Rönnqvist, 2007)		X								X
(Bredström and Rönnqvist, 2008)		X				X	X			
(Dohn et al., 2011)		X								X
(Rasmussen et al., 2012)			X							X
(Afifi et al., 2013)		X					X			
(Rousseau et al., 2013)				Dynamique (SDVDP)			X		X	
(Labadie et al., 2014)		X				X	X			
(Mankowska et al., 2014)			X			X	X			
(Afifi et al., 2016)		X					X			
(Haddadene et al., 2016)		X					X			
(Hojabri et al., 2018)				X					X	
(Liu et al., 2018)		X				X	X			
(Masmoudi and Cheikhrouhou, 2018)					HVRPSB		X			
(Parragh and Doerner, 2018)		X			STRSP	X	X			
(Fink et al., 2019)					AVRPWVS					X

1.2. Le Generalised Workforce Scheduling and Routing Problem

De manière similaire au WSRP, le GWSRP consiste à affecter un employé à chaque visite, à déterminer la tournée de chaque employé, et à planifier les horaires de début des visites. La fonction objectif du GWSRP est identique à celle du WSRP, avec la minimisation des coûts opérationnels (distance plus coût de réalisation), la maximisation de la qualité de service (QoS) des clients, la maximisation de la qualité de service (QoS) des employés, et enfin la maximisation du nombre de visites réalisées. La différence entre ces deux problèmes réside dans la prise en considération de contraintes de coordination dans le cadre du GWSRP. Pour le GWSRP, ces contraintes imposent une dépendance entre les tournées, tandis que pour le WSRP, les tournées sont indépendantes les unes des autres.

Les données du GWSRP sont les mêmes que celles du WSRP, qui sont introduites dans le chapitre 3 :

- V : l'ensemble des visites.
- W : l'ensemble des employés.
- I : l'ensemble des dépôts de départ.
- I_w : le dépôt de départ de l'employé w.
- L : l'ensemble des dépôts de retour.
- L_w : le dépôt de retour de l'employé w.
- Nts : l'ensemble de tous les nœuds du graphe, $Nts = V \cup I \cup L$.
- Pt_i : la durée de service de la visite i (ou processing time).
- $[E_i\ ;\ L_i]$: la fenêtre de temps de la visite i.
- PC_i^w : le coût de réalisation de la visite i par l'employé w.
- ρ_i^w : la qualité de service (QoS) de la visite i pour l'employé w.
- $CoTr_i^w$: le contrat de travail autorisant ($CoTr_i^w = 1$) – ou non ($CoTr_i^w = 0$) – l'employé w à réaliser la visite i.
- $T_{i,j}$: le temps de transport de la visite i à la visite j.
- $[TW_{inf}^w\ ;\ TW_{sup}^w]$: la fenêtre de temps de l'employé w.
- PA_i^w: une donnée binaire valant 1 si w apprécie la région où est située i, et valant 0 s'il ne l'apprécie pas.

Les contraintes de coordination du GWSRP couvrent un large spectre de contraintes issues des problèmes de production et/ou maintenance, des problèmes de tournées de véhicules et des problèmes de service à domicile (notamment le Home Health Care problem). Ces contraintes englobent, entre autres, des contraintes disjonctives, des contraintes de délai minimal, de délai maximal, et de simultanéité.

La Figure 1 présente une classification des problèmes d'ordonnancement et de tournées de véhicules en fonction de la prise en compte des contraintes de coordination et des différences entre les employés. Le GWSRP appartient à la famille des problèmes d'ordonnancement et de tournées de véhicules avec des employés non-indentiques et des contraintes de coordination.

Néanmoins, la frontière entre les problèmes théoriques avec et sans contraintes de coordination n'est pas toujours clairement définie et, par exemple, certaines publications traitant des problèmes de Home Health Care (HHC ou HHC Scheduling – HHCS) sont sans contrainte de coordination (Bertels and Fahle, 2006; Chahed et al., 2009; Maenhout and

Vanhoucke, 2009), alors que d'autres prennent en compte ces contraintes (Blais et al., 2003; Eveborn et al., 2006; Bachouch et al., 2011). Des remarques similaires s'appliquent au WSRP sans contrainte de coordination dans (Castillo et al., 2009; Goel and Meisel, 2013; Laesanklang et al., 2015; Algethami et al., 2017) (voir chapitre 3), et avec contraintes de coordination dans (Laesanklang et al., 2016). De manière générale, les problèmes de tournées de véhicules (VRP) sont classiquement étudiés sans contrainte de coordination, mais des études récentes incluent ces considérations. (Drexl, 2012) présente un état de l'art des VRP avec ces contraintes et leurs applications.

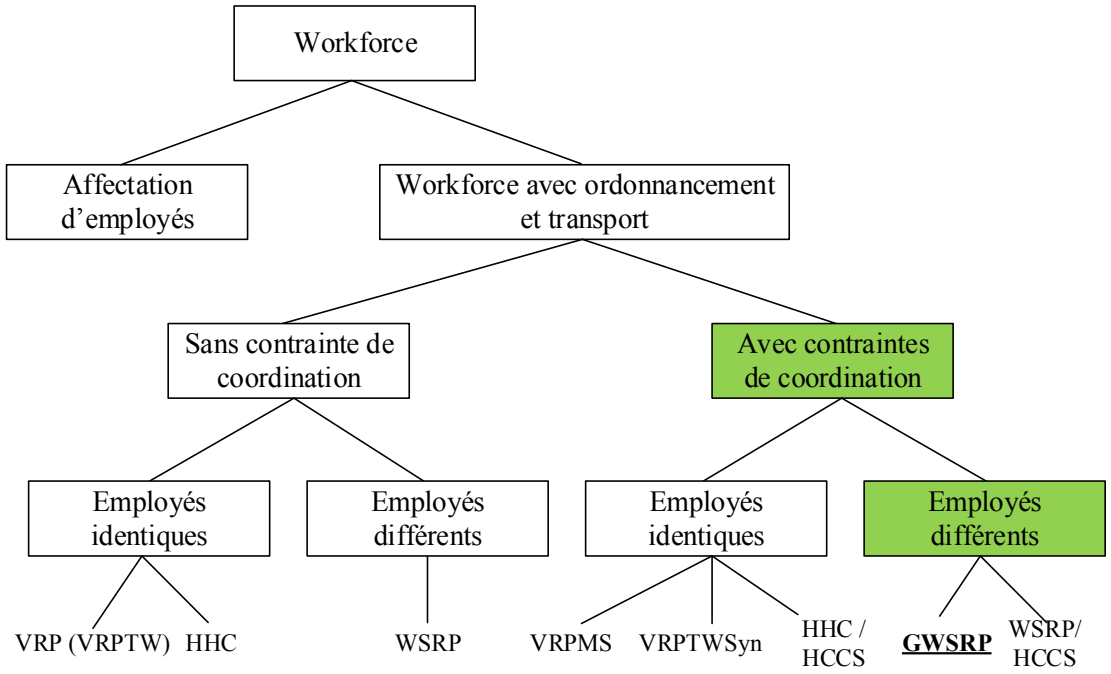


Figure 1 Classification du GWSRP

Le Tableau 2 résume les caractéristiques des principaux problèmes de tournées de véhicules avec des contraintes de coordination.

Le VRPTWSyn est le Vehicle Routing and Scheduling Problem with Time Windows and Synchronized visits introduit par (Bredström and Rönnqvist, 2008). HCCS est le Home Care Crew Scheduling Problem qui est une généralisation du uncapacitated and multiple-depot VRPTW, introduit par (Rasmussen et al., 2012). VRPMS est le VRP with Multiple Synchronization constraints de (Rousseau et al., 2013) et (Hojabri et al., 2018).

Le GWSRP est le problème le plus général, car il prend entre autres en compte à la fois des temps de service (ou processing times), des fenêtres de temps pour les employés et les visites, des coûts de réalisations en fonction de l'employé et des zones géographiques, et enfin des employés qui ont des caractéristiques différentes.

Une différence majeure avec le VRPTWSyn est que dans ce problème, les employés sont tous identiques : même temps de trajet, même coût, etc. Un autre point clef du GWSRP est la prise en compte de la qualité de service (QoS) pour les clients et pour les employés dans la fonction objectif, comme cela a déjà été souligné à maintes reprises pour le WSRP dans le chapitre 3.

Pour le HHC, la fonction objectif est une somme hiérarchisée de la distance, du coût de réalisation qui est considéré comme un critère de qualité de service (QoS) et du nombre de visites non réalisées. D'après (Rasmussen et al., 2012), le coût de réalisation d'une visite dépend de l'employé qui la réalise, et permet de modéliser la satisfaction du client. Un coût faible signifie que l'employé est fortement apprécié et, au contraire, un coût élevé signifie que l'employé n'est pas désiré. Pour les GWSRP, il y a deux coûts distincts un pour la réalisation, et un pour la satisfaction, les critères de la distance et de réalisation sont sommés pour définir un compromis entre les deux.

Le VRPMS concerne le transport et l'installation de matériel à domicile. Les véhicules transportant le matériel ont une capacité limitée, tandis que les véhicules des employés qui assurent l'installation n'ont pas de capacité car il y a un seul employé par véhicule (Hojabri et al., 2018). Pour le GWSRP, le transport concerne uniquement les employés et pas de biens, chaque employé a un véhicule, et donc aucune contrainte sur leur capacité n'est prise en compte. De plus, le VRPMS ne prend pas en compte un critère lié à la qualité de service (pour les clients ou les employés), et les employés sont tous identiques, avec des capacités de transport limitées.

Tableau 2 Résumé des caractéristiques des problèmes de la littérature

	GWSRP	**VRPTWSyn**	**HCCS**	**VRPMS**
Nombre d'employés limité	X	X	X	-
Temps de service	X	X	X	X
Fenêtre de temps pour les visites	X	X	X	X
Fenêtre de temps pour les employés	X	X	X	-
Compatibilité employé/localisation	X	-	-	-
Coût de réalisation	X	-	X	-
Compatibilité employé/visites	X	X	-	-
Zone géographique	X	-	-	-
Employés affectés à une tournée	X	-	X	-
Qualité de service pour le client	**X**	**X**	**X**	**-**
Qualité de service pour les employés	**X**	**-**	**-**	**-**
Employés non-identiques	**X**	**-**	**X**	**-**
Priorité des visites	-	-	X	-
Capacité des véhicules	-	-	-	X
Fonction Objectif				
Objectif 1	Distance + processing time	Distance / Équilibrage/ QoS des clients	Distance	Distance
Objectif 2	QoS des clients		QoS des clients	
Objectif 3	**QoS des employés**		Somme pondérée des visites non réalisées	
Objectif 4	Nb visites non réalisées			

La fonction du GWSRP est identique à celle du WSRP et est donc :

$$f = \lambda_1 \sum_{i \in V \cup I_w \cup L_w} \sum_{j \in V \cup I_w \cup L_w} (T_{i,j} + p_i^w) x_{i,j}^w + \lambda_2 \sum_{i \in V} \left(3 - \sum_{j \in V \cup L_w} \rho_i^w x_{i,j}^w \right)$$

$$+ \lambda_3 \sum_{i \in V} (\psi_i^w + \theta_i^w) + \lambda_4 \sum_{i \in V} y_i$$

2. Définition des contraintes de coordination

2.1. Représentation des tournées dans un graphe disjonctif

Le GWSRP est modélisé par un graphe dans lequel les visites, ainsi que les dépôts de départ et les dépôts de retour, sont des nœuds. Un arc représente la route reliant deux visites. La tournée de l'employé w est représentée dans un graphe par un chemin partant du dépôt de départ I_w et finissant au dépôt de retour L_w. Ce chemin peut comprendre une succession de visites i effectuées par l'employé. Chacune des visites est caractérisée par une date st_i qui correspond à la date de début de service par l'employé. L'arc reliant deux nœuds i et u – deux visites – du graphe est pondéré par la distance $T_{i,u}$ entre ces deux visites, plus la durée de service Pt_i de la visite i.

La fenêtre de temps $[E_i ; L_i]$ d'une visite i est modélisée dans le graphe par deux arcs : un time-lag maximal pour la borne supérieure et un time-lag minimal pour la borne inférieure (Figure 2). Le time-lag maximal va de la visite i au dépôt de départ de l'employé et est pondéré par la valeur L_i. Le time-lag minimal est issu du dépôt de départ et orienté vers la visite i, il a pour valeur E_i. Ces deux time-lags imposent que la date de début st_i de la visite i soit supérieure à E_i et inférieure à L_i. Le time-lag maximal empêche que la date st_i « s'éloigne » de la date 0 qui est la date de départ du dépôt.

Sur le schéma de la Figure 2, la visite i possède une fenêtre de temps $[20 ; 150]$ et la visite i ne doit pas démarrer avec un décalage supérieur à 100 unités de temps après le début de la tournée (time-lag maximal).

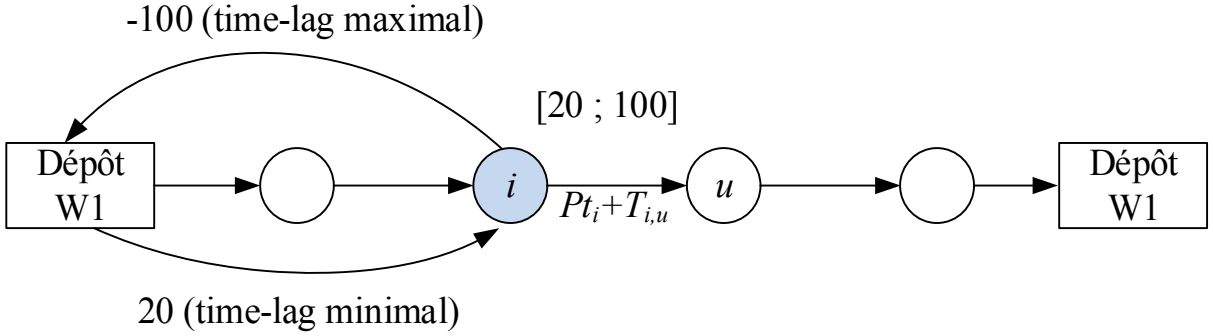


Figure 2 Modélisation d'une tournée

Ainsi, si la date de début de la tournée est fixée à 0 (Figure 3) la date maximale d'exécution de i est de valeur 100 ($st = 100$).

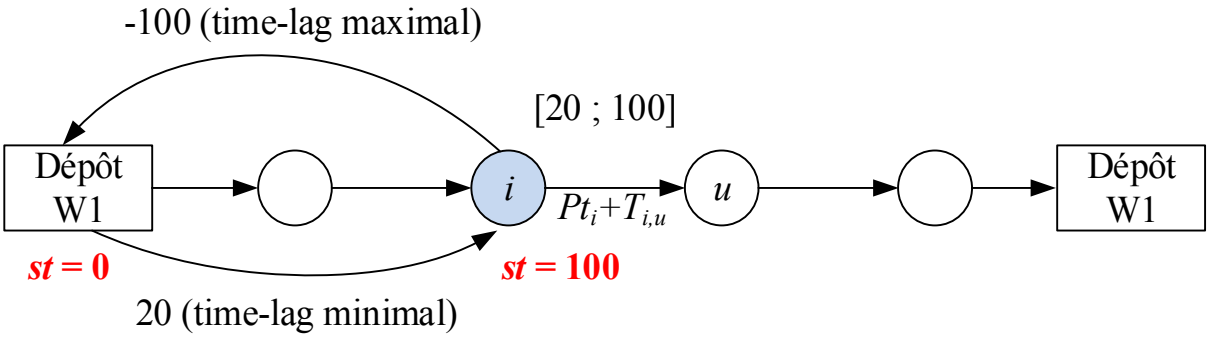


Figure 3 Date de début au plus tard de la visite

Une solution peut être définie avec une date de début de la visite i valant 120 ($st = 120$) et décaler la date de début de la tournée à la date $st = 20$. Le décalage de 20 unités de temps de la date de début de la tournée est réalisé par un simple calcul de plus long chemin comme le montre la Figure 4. Sur cette figure, la date $st = 120$ de la visite i se propage au dépôt sous la forme de 120-100, ce qui donne 20 comme date de début du sommet 0.

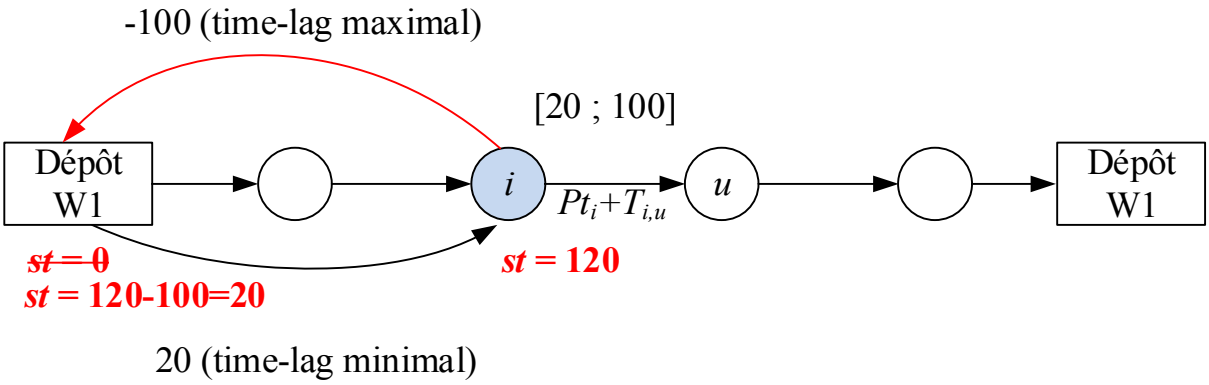


Figure 4 Modification de la date de début de la tournée en fonction de la date de la visite

Un algorithme de plus court chemin de type Bellman-Ford permet de calculer les dates de début des visites au plus tôt (voir chapitre 1). Le chapitre 3 présente un algorithme pour calculer le coût minimal de la tournée, car ce dernier dépend de la date de début des visites afin de maximiser la qualité de service des employés lorsque les tournées sont indépendantes les unes des autres. Pour le GWSRP, les contraintes de coordination impliquent des dépendances entre les différentes tournées, et l'algorithme du chapitre 3 ne peut pas s'appliquer directement.

La représentation de l'ensemble des tournées peut s'apparenter à un problème de type Job-shop Scheduling, où une gamme est équivalente à une tournée, et où les opérations passant sur les machines correspondent aux visites. La valeur des arcs est la somme du temps de service de la visite plus la distance entre les deux visites. La Figure 5 illustre les deux représentations possibles d'un ensemble de tournées. La partie basse de la figure est une modélisation type « problème d'ordonnancement ». De nombreux articles issus de la communauté d'ordonnancement prennent en compte des contraintes temporelles entre les opérations (Roy and Sussmann, 1964; Caumond et al., 2008; Artigues et al., 2011). Il existe donc des outils et modélisations spécifiques, qu'il est possible d'adapter aux problèmes de production et de transport intégrés.

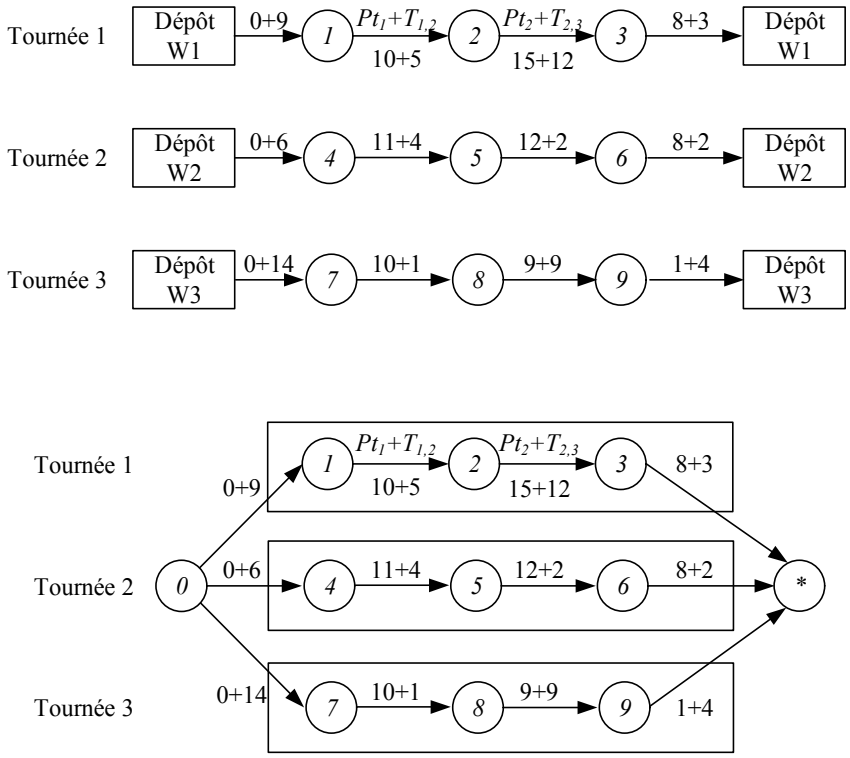


Figure 5 Tournées de véhicules et graphe conjonctif

2.2. Contraintes de coordination

Cette section introduit les neuf types de contraintes prises en compte dans le GWSRP. Les contraintes sont issues des problèmes classiques d'ordonnancement et des problèmes de tournées de véhicules.

2.2.1. Visites en disjonction

Si un patient nécessite la visite de plusieurs personnels de santé, mais que l'ordre de passage n'a pas d'importance, il s'agit alors de visites en disjonction. Des contraintes de visites en disjonction surviennent par exemple, lorsque le patient a besoin de voir une infirmière, un kinésithérapeute et un médecin dans la journée. L'ordre de passage de ces trois personnels de santé n'est pas contraignant, mais pour que ceux-ci puissent réaliser correctement leur travail, il est nécessaire qu'ils passent chacun à leur tour. Cette contrainte est également très fréquente dans les problèmes d'ordonnancement (type Job-shop ou Flow-shop) où une machine ne peut réaliser qu'une opération à la fois.

De manière plus formelle, soit un ensemble de visites i, j, k, qui ne peuvent pas s'exécuter simultanément. La contrainte disjonctive se traduit linéairement, pour les visites i et j, par :

$$st_i + Pt_i - M(1 - b_{ij}) \leq st_j$$

$$st_j + Pt_j - Mb_{ij} \leq st_i$$

$$b_{ij} + b_{ji} = 1$$

Avec $b_{ij} = 1$ si la visite i est avant la visite j, et $b_{ij} = 0$ sinon. M est un entier (suffisamment grand) et, st_i et st_j sont les dates respectives de début des visites i et j. Cette contrainte est modélisée dans un graphe par des arêtes disjonctives (Figure 6) qui sont à orienter pour obtenir une solution. Un tel graphe est semblable au graphe disjonctif du Job-shop Scheduling Problem (Roy and Sussmann, 1964). La partie supérieure de Figure 6 présente un diagramme de Gantt où les trois visites sont ordonnancées sans se chevaucher.

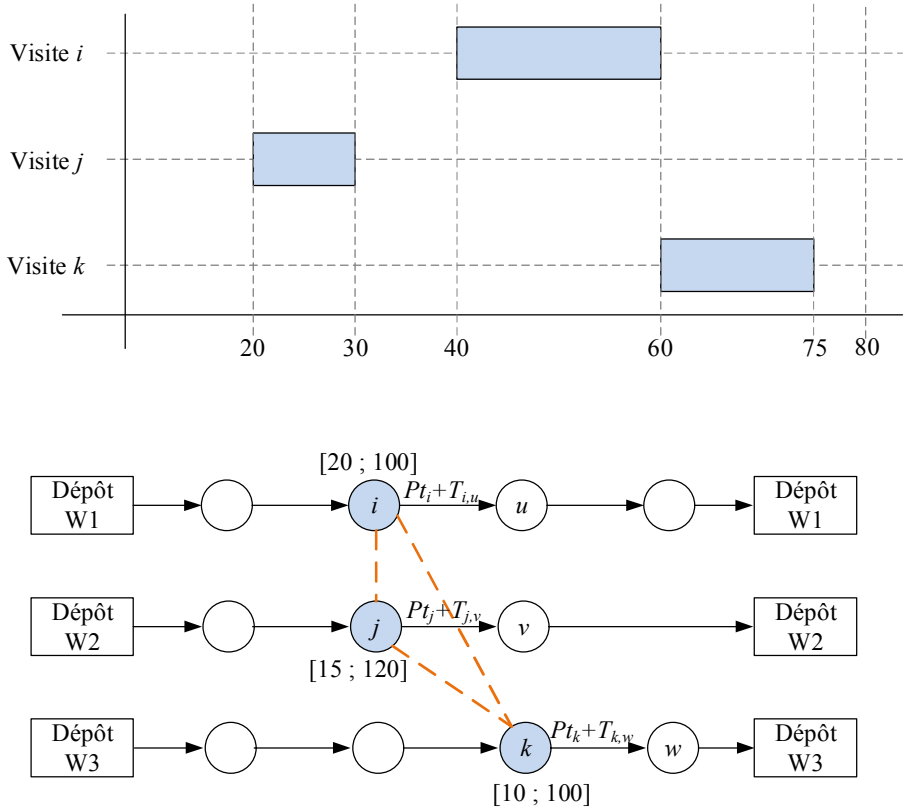


Figure 6 Contrainte disjonctive

Il est important de noter que cette contrainte n'est pas transitive. Si elle concerne trois visites, alors il faut écrire toutes les combinaisons des contraintes deux à deux.

2.2.2. Visites synchronisées ou visites simultanées

Une contrainte de synchronisation (ou visites simultanées) impose que les deux visites i et j débutent en même temps. Cette contrainte de coordination est souvent présente dans la littérature sous le nom de « contrainte de synchronisation ». Elle permet de modéliser des cas où deux personnes doivent être présentes en même temps pour une livraison (Hojabri et al., 2018) ; où deux cadres de santé sont nécessaires pour la réalisation d'un soin à domicile (Rasmussen et al., 2012) ; où un véhicule doit être synchronisé avec un drone pour des livraisons (Mbiadou Saleu et al., 2018). Cette contrainte se formalise par :

$$st_i = st_j$$

st_i et st_j sont les dates respectives de début des visites i et j. Cette contrainte se modélise dans le graphe par deux arcs : l'un de la visite i à la visite j, et l'autre de la visite j à la visite i, tous les deux de poids nul (Figure 7). La partie supérieure de la Figure 7 est un diagramme de Gantt où les deux visites (i et j) débutent en même temps. Leurs durées respectives ne sont pas nécessairement identiques.

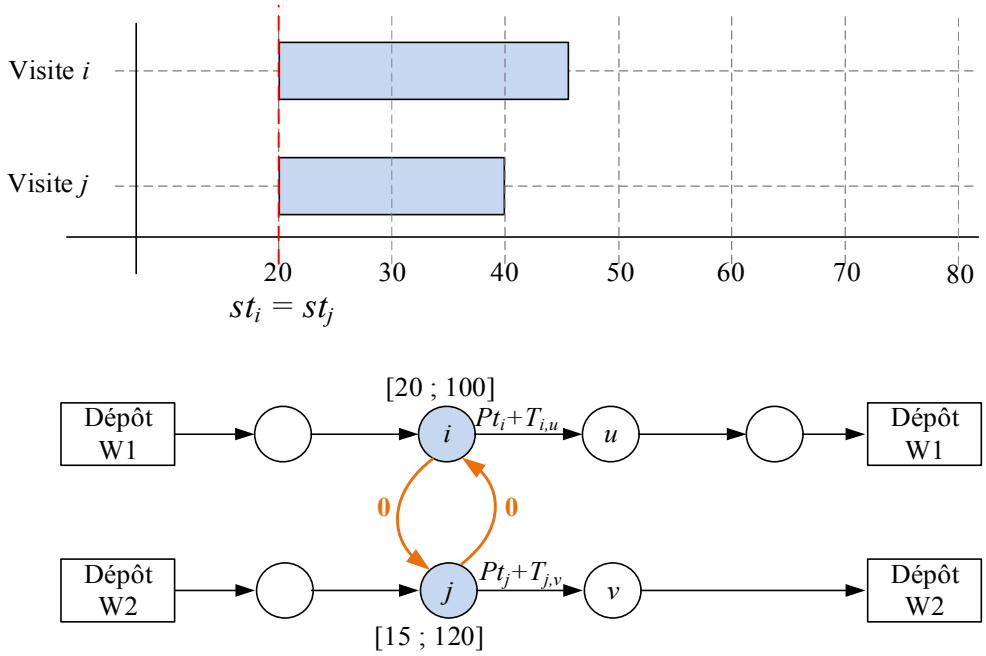


Figure 7 Contrainte de simultanéité

2.2.3. Visites ayant une durée d'exécution en commun

Une contrainte de coordination sur des visites ayant une durée d'exécution en commun impose que plusieurs visites doivent avoir une durée T_c d'exécution minimale en commun. Par exemple, deux aides-soignants doivent être présents pendant quelques instants chez un patient pour faire un soin. Ou alors pour l'installation de nouveau matériel, un technicien livre le matériel et un spécialiste le configure ensuite. Cette contrainte se formalise par :

$$max(st_i, st_j) - min(st_i + Pt_i, st_j + Pt_j) \geq T_c$$

Ou par le couple d'équations :

$$st_i + (T_c - Pt_j) \leq st_j$$

$$st_j + (T_c - Pt_i) \leq st_i$$

La contrainte se modélise dans le graphe par deux arcs : l'un partant de i et arrivant en j de poids $T_c - Pt_j$, et l'autre allant dans le sens inverse (de j vers i) de poids $T_c - Pt_i$ (Figure 8). T_c est la durée d'exécution en commun, et Pt_i la durée de la visite i. Il est important de noter que $T_c \leq Pt_i$ et $T_c \leq Pt_j$. Par conséquent, $T_c - Pt_i \leq 0$ et $T_c - Pt_j \leq 0$. Les deux arcs représentent en réalité deux time-lags maximaux. La partie supérieure de la Figure 8 présente un diagramme de Gantt avec deux visites et une durée T_c en commun. Comme il n'est pas trivial de trouver la valeur des arcs dans le graphe, un exemple est présenté ensuite.

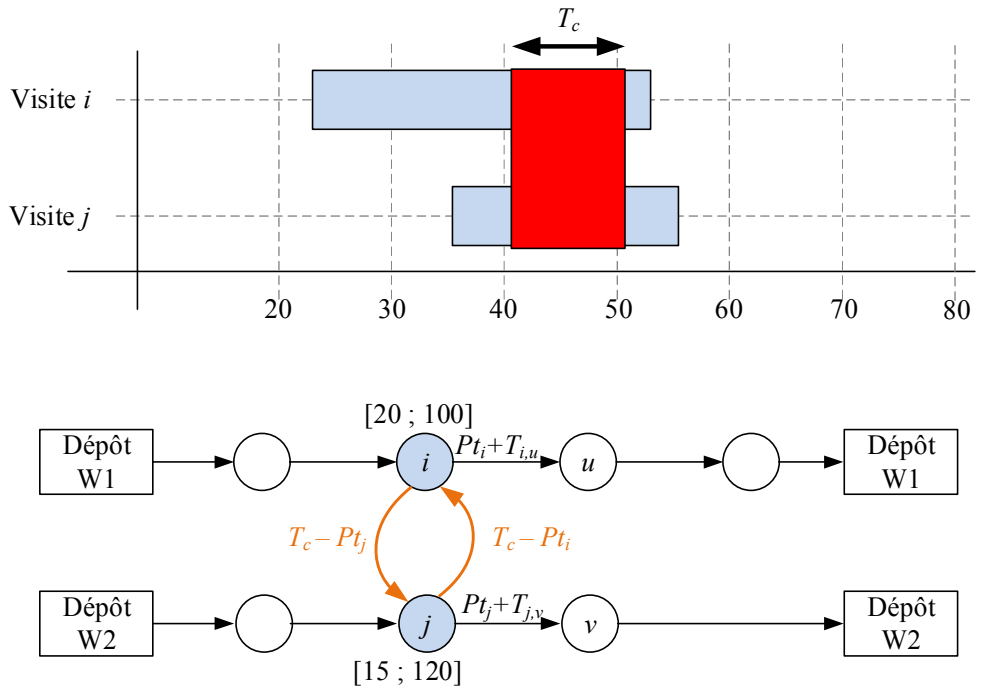


Figure 8 Contrainte de visites avec une durée d'exécution en commun

Soient $Pt_i = 30$, $Pt_j = 20$ et $T_c = 10$. Soit $st_i = 30$ la date de début de la visite i connue (sans perte de généralité, le cas inverse peut être étudié). La date de début au plus tôt ES_j de j est donc 20 (Figure 9). Si j débute plus tôt que la date 20, alors la visite n'a pas une durée de 10 unités de temps en commun.

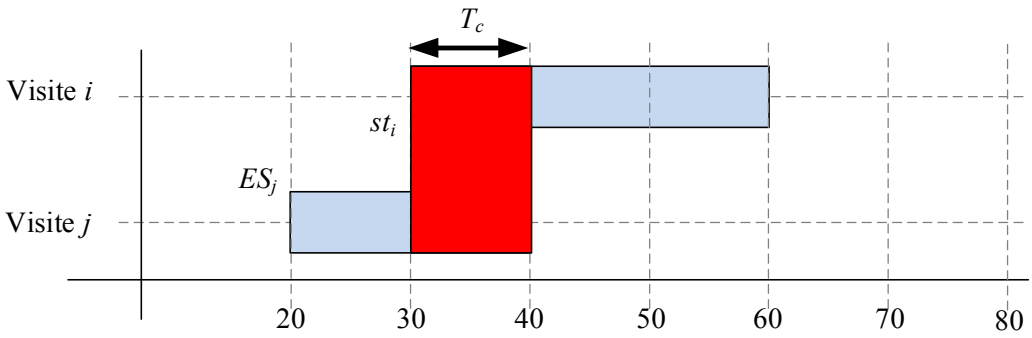


Figure 9 Date au plus tôt de j

La visite j doit finir au plus tôt 10 unités de temps après le début de la visite i. La formule qui permet d'obtenir la date au plus tôt ES_j de j en fonction de st_i est $ES_j = st_i + T_c - Pt_j = 30 + 10 - 20 = 20$. Cette formule se traduit par un arc allant de i vers j et de valeur $T_c - Pt_j$ (voir la Figure 10).

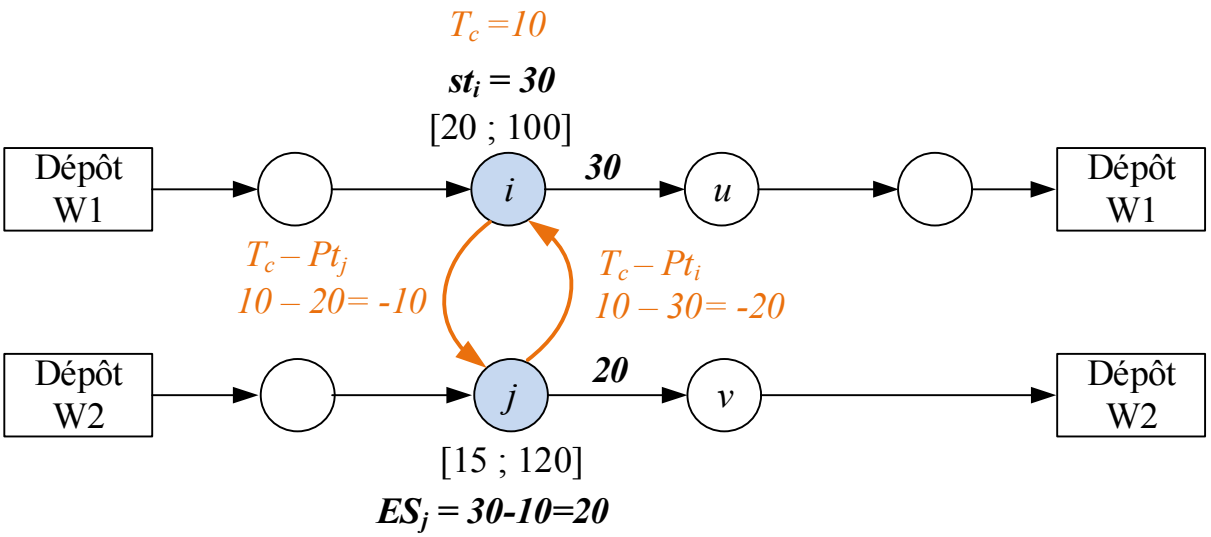


Figure 10 Date au plus tôt de j

La date au plus tard LS_j de j, toujours en fonction de st_i, est $LS_j = 50$. Si la visite j débute plus tard, alors la contrainte de coordination n'est pas respectée (cf. Figure 11). La visite j doit commencer au plus tard 10 unités de temps avant la fin de la visite i. La date LS_j se calcule avec la formule : $LS_j = st_i + Pt_i - T_c = 30 + 30 - 10 = 50$. Cette contrainte se traduit par un arc allant de j vers i, et de poids $-(Pt_i - T_c)$ (voir la Figure 10). Elle impose une mise à jour de la date de début de i si la visite j commence après la date 50.

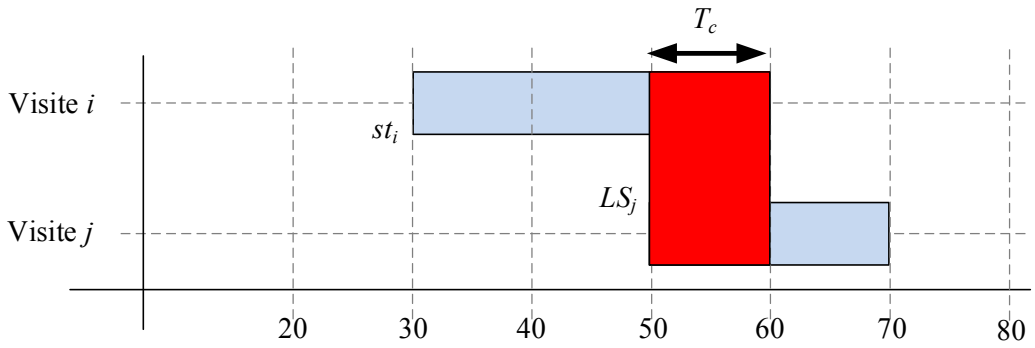


Figure 11 Date au plus tard de j

2.2.4. Visite ayant une durée d'exécution pendant un intervalle particulier

La visite i à une fenêtre de temps de type $[E_i ; L_i]$, qui représente une contrainte sur sa date de début st_i. La visite i doit en plus s'exécuter dans l'intervalle particulier $[CE_i ; CL_i]$. Cette contrainte se traduit linéairement par :

$$(Cl - Pt_i) \leq st_i$$

$$0 \geq st_i - CE$$

Cette contrainte peut être vue comme une réécriture de la fenêtre de temps de la visite : $st_i \in [CL - Pt_i ; CE]$ (voir la Figure 12).

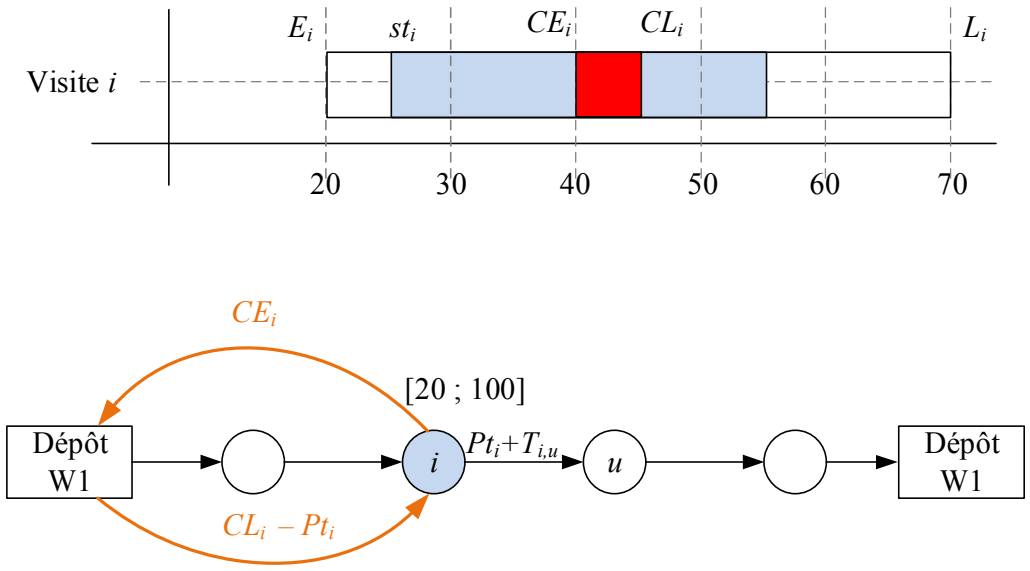


Figure 12 Contrainte d'exécution de la visite dans un intervalle particulier

Soient $[CE_i ; CL_i] = [40 ; 45]$ et $Pt_i = 20$, alors la visite i doit débuter au plus tôt à $ES_i = 25$, sinon, la contrainte n'est pas respectée (partie haute de la Figure 13). La date LS_j au plus tard de la visite est 40 (partie basse de la Figure 13). Les valeurs de ES_i et LS_j sont obtenues par $ES_i = CL_i - Pt_i = 45 - 20 = 25$ et $LS_i = CE_i = 40$.

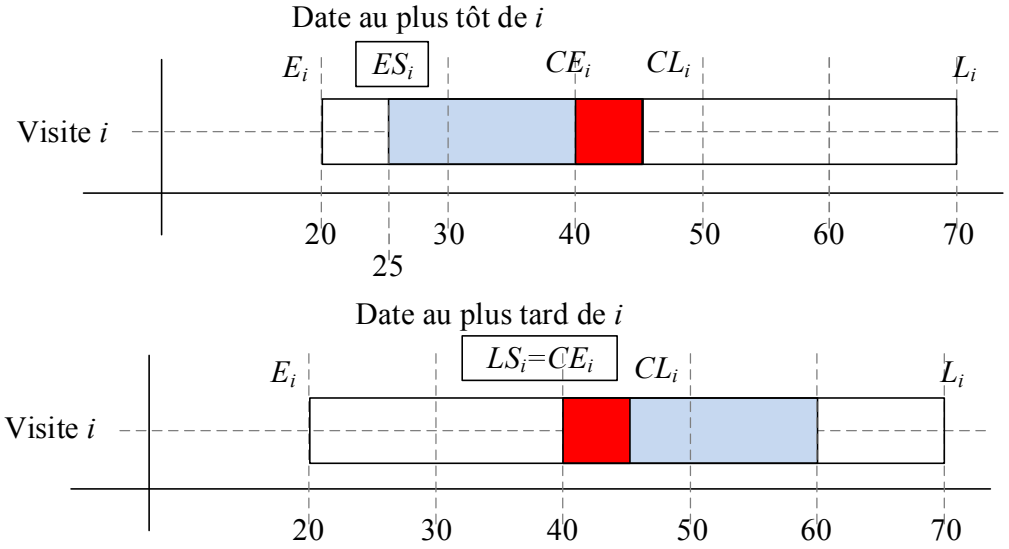


Figure 13 Dates au plus tôt et au plus tard de i

2.2.5. Visites conjonctives

Deux visites conjonctives i et j sont liées par une contrainte conjonctive qui impose un ordre spécifique (i avant j) ce qui, d'un point de vue temporel, impose que la seconde visite (j) commence après la fin de la première (i). Cette contrainte est régulièrement considérée dans les problèmes d'ordonnancement sous le nom de « contrainte de précédence ».

La date ES_j au plus tôt de la visite j est la date de fin de la visite i : $ES_j = st_i + Pt_i$. La date de début st_j de la visite j doit donc être supérieure à $st_j \geq st_i + Pt_i$ (voir la Figure 14).

$$st_j \geq st_i + Pt_i$$

La Figure 14 présente un diagramme de Gantt avec une contrainte conjonctive (partie supérieure) et sa modélisation dans un graphe (partie inférieure de la figure). La visite j ne peut pas débuter avant la fin de la visite i.

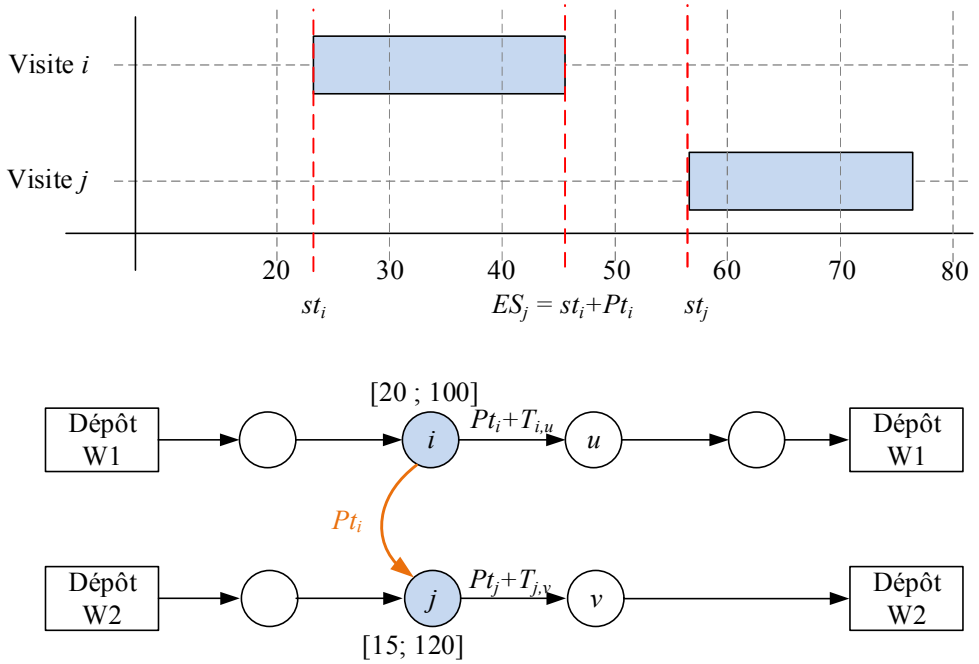


Figure 14 Contrainte conjonctive

2.2.6. Time-lag minimal conjonctif

Un time-lag minimal conjonctif entre deux visites i et j impose l'ordre des visites et un délai entre celles-ci : la visite j commence après une certaine date qui dépend de la date de début de i et de la valeur du time-lag minimal (Drexl, 2012). Cette contrainte permet de modéliser des situations où une certaine durée minimale doit être observée entre deux visites. C'est le cas, par exemple, de la prise d'un médicament qui doit être réalisée deux fois dans la journée, avec une durée minimale à respecter entre les deux prises.

Les time-lags minimaux conjonctifs sont régulièrement étudiés dans le cadre de problèmes d'ordonnancement (Mitten, 1959; Brucker and Knust, 1999; Caumond et al., 2008; Artigues et al., 2011) et de tournées de véhicules (Dohn et al., 2011; Rasmussen et al., 2012; Mankowska et al., 2014).

Il est important de noter qu'un time-lag minimal $TL^-_{i,j}$ impose l'ordre i avant j et que sa valeur est positive. Par ailleurs, la contrainte des visites conjonctives est un cas particulier de cette contrainte où $TL^-_{i,j} = Pt_i$. La contrainte s'écrit linéairement, pour i avant j :

$$st_i + TL^-_{i,j} \leq st_j$$

La contrainte se modélise dans le graphe par un arc de la visite i vers la visite j de valeur $TL^{-}_{i,j}$ (Figure 15). La partie supérieure de la Figure 15 est le diagramme de Gantt, la visite j ne peut pas débuter plus tôt que la date 50 à cause du time-lag minimal. Toutefois, rien n'impose que la visite j début directement à la date 50 : elle peut commencer plus tard, comme c'est le cas sur le diagramme de Gantt de la Figure 15.

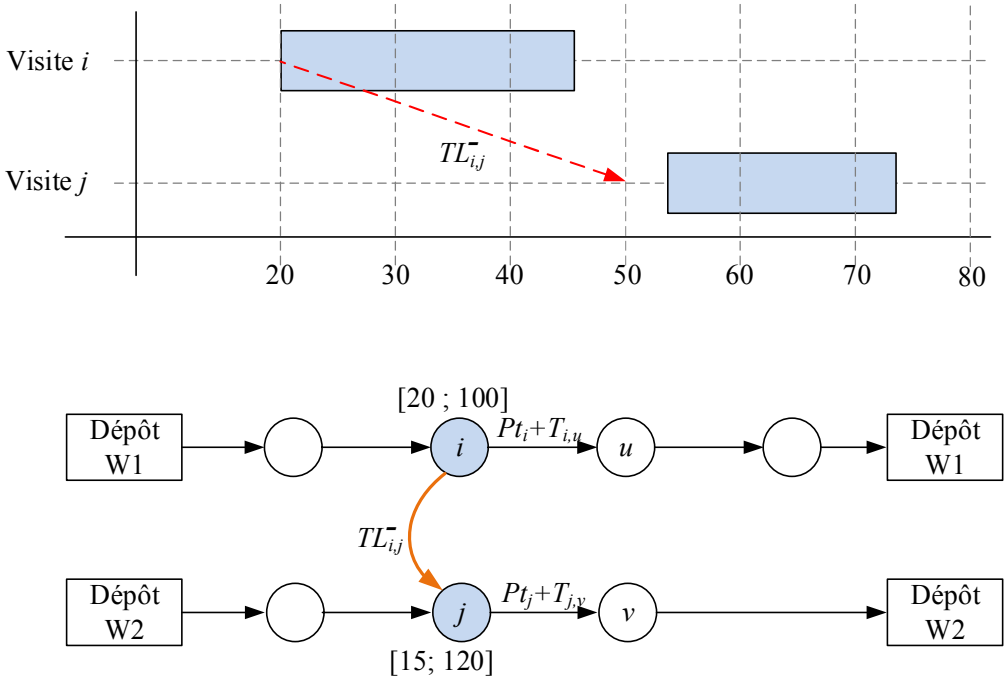


Figure 15 Contrainte de time-lag minimal conjonctif

2.2.7. Time-lag minimal disjonctif

Cette contrainte de time-lag minimal disjonctif est une extension de la contrainte de coordination de type time-lag minimal conjonctif. Pour la contrainte de time-lag minimal disjonctif, l'ordre des visites n'est pas imposé (contrairement au time-lag minimal conjonctif). La valeur du time-lag dépend de l'ordre des deux visites et peut être différente. La contrainte s'écrit linéairement par :

$$st_i + TL^{-}_{i,j} - M(1 - b_{ij}) \leq st_j$$

$$st_j + TL^{-}_{j,i} - M(1 - b_{ji}) \leq st_i$$

$$b_{ij} + b_{ji} = 1$$

Avec $b_{ij} = 1$ si i est avant j, et $b_{ij} = 0$ sinon. $TL^{-}_{i,j}$ est la valeur du time-lag de i vers j, et $TL^{-}_{j,i}$ est la valeur du time-lag de j vers i. Cette contrainte est une généralisation de la contrainte de coordination de visite en disjonction, et dans ce cas $TL^{-}_{i,j} = Pt_i$. Elle se modélise dans le graphe par une arête disjonctive qu'il faut orienter pour obtenir une solution.

2.2.8. Time-lag maximal conjonctif

Un time-lag maximal conjonctif entre deux visites i et j impose qu'un délai maximal entre les deux visites soit respecté (Mitten, 1959), et impose aussi l'ordre des deux visites. Cette contrainte est courante dans les problèmes d'ordonnancement (Caumond et al., 2008; Artigues et al., 2011) ou de tournées de véhicules de type Dial-a-Ride (Cordeau and Laporte, 2003). Elle permet également de modéliser des cas où deux soins ne doivent pas être séparés de plus d'une durée maximale (Rasmussen et al., 2012), ou bien des cas où la surveillance d'un patient (ou d'une pièce) doit avoir lieu dans un certain délai maximal entre deux visites, par exemple. La contrainte s'écrit linéairement, pour i avant j :

$$st_i - TL^+_{i,j} \geq st_j$$

$$st_i \leq st_j$$

$TL^+_{i,j}$ est la valeur (positive) du time-lag maximal de i vers j. La visite j doit débuter après la visite i. La Figure 16 présente un diagramme de Gantt où la visite j commence avant la limite donnée par $TL^+_{i,j}$. La contrainte se modélise dans un graphe par deux arcs, l'un de i vers j et de valeur 0, l'autre de j vers i et de valeur $-TL^+_{i,j}$.

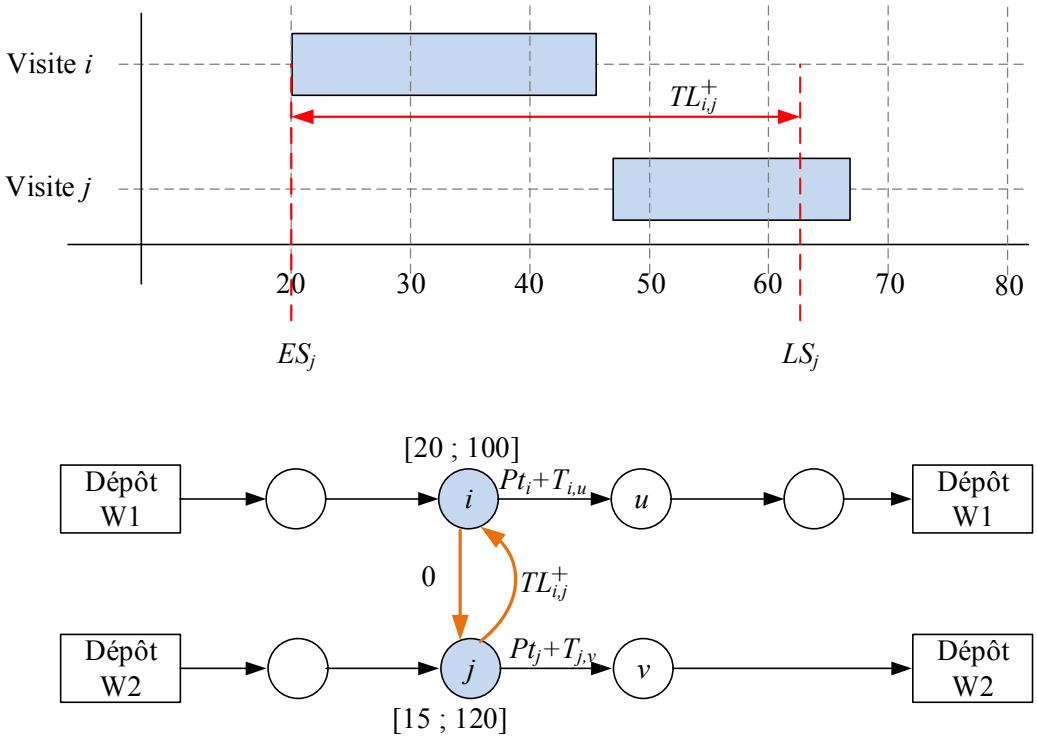
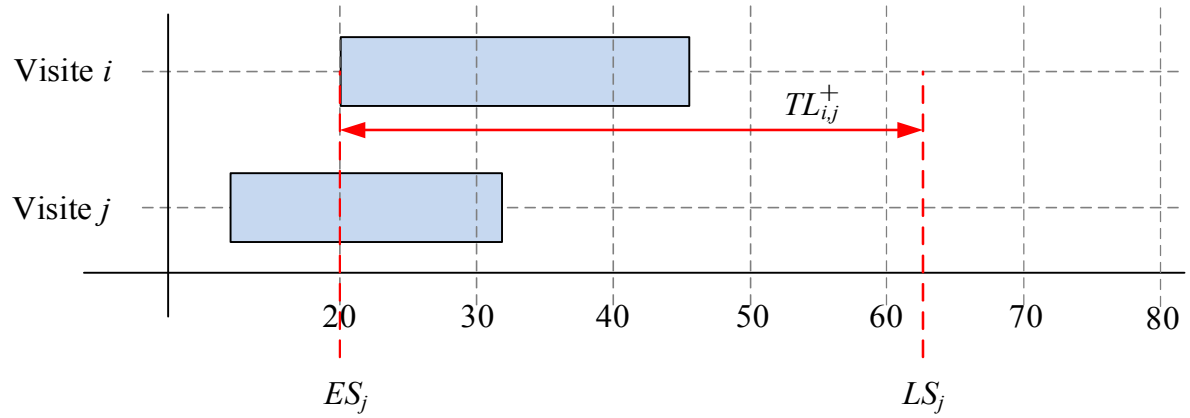


Figure 16 Contrainte de time-lag maximal conjonctif

L'arc de valeur 0 est obligatoire, car il impose l'ordre relatif entre i et j et empêche ainsi de planifier j avant i. Le cas où j est planifiée avant i satisfait automatiquement la contrainte de time-lag maximal comme le montre la Figure 17.



La contrainte de time-lag maximal est
trivialement respectée car j est planifiée avant i

Figure 17 Contrainte de time-lag sans ordre relatif

2.2.9. Time-lag maximal disjonctif

La contrainte de time-lag maximal disjonctif est une contrainte de time-lag maximal dépendant de l'ordre d'exécution des visites. Cet ordre n'est pas imposé. Suivant l'ordre de passage des visites, la valeur du time-lag varie. Cette contrainte s'écrit linéairement :

$$st_i - TL^{+}_{i,j} + M(1 - b_{ij}) \geq st_j$$

$$st_i - M(1 - b_{ij}) \leq st_j$$

$$st_j - TL^{+}_{j,i} + M(1 - b_{ji}) \geq st_i$$

$$st_j - M(1 - b_{ji}) \leq st_i$$

$$b_{ij} + b_{ji} = 1$$

$b_{ij} = 1$ si i est avant la visite j et $b_{ij} = 0$ sinon. $TL^{+}_{i,j}$ est la valeur du time-lag de i vers j, et $TL^{+}_{j,i}$ est la valeur du time-lag de j vers i.

2.2.10. Combinaison de contraintes

En plus des neuf contraintes de coordination présentées précédemment, des combinaisons de contraintes sont possibles. Ces combinaisons ne sont pas exprimées explicitement dans les instances spécifiques du GWSRP. Elles sont néanmoins présentées dans cette section car elles peuvent être traitées de manière implicite.

2.2.10.1. Time-lag minimal + Time-lag maximal conjonctifs

La contrainte de Time-lag minimal + Time-lag maximal conjonctifs est introduite par (Rasmussen et al., 2012) et est la concaténation des contraintes de time-lag minimal conjonctif et de time-lag maximal conjonctif. L'ordre des visites est imposé dans cette contrainte. La contrainte s'écrit, pour i avant j :

$$st_i + TL^-_{i,j} \leq st_j$$

$$st_i - TL^+_{i,j} \geq st_j$$

Avec $TL^-_{i,j}$ la valeur du time-lag minimal et $TL^+_{i,j}$ la valeur du time-lag maximal.

2.2.10.2. Time-lag minimal + Time-lag maximal disjonctifs

La contrainte Time-lag minimal + Time-lag maximal disjonctifs est similaire à la contrainte Time-lag minimal + Time-lag maximal conjonctifs, mais l'ordre des visites n'est pas imposé, et la valeur des différents time-lags dépend de cet ordre. La contrainte s'écrit linéairement :

$$st_i + TL^-_{i,j} - M(1 - b_{ij}) \leq st_j$$

$$st_i - TL^+_{i,j} + M\big(1 - b_{ij}\big) \geq st_j$$

$$st_j + TL^-_{j,i} - M(1 - b_{ji}) \leq st_i$$

$$st_j - TL^+_{j,i} + M\big(1 - b_{ji}\big) \geq st_i$$

$$b_{ij} + b_{ji} = 1$$

$TL^-_{i,j}$ et $TL^-_{j,i}$ sont les valeurs des time-lags minimaux. $TL^+_{i,j}$ et $TL^+_{j,i}$ sont les valeurs du time-lags maximaux. $b_{ij} = 1$ si i avant j, $b_{ij} = 0$ sinon (et inversement avec b_{ji}).

2.3. Contraintes de coordination en PPC

Il est important de voir que les contraintes de coordination peuvent se réécrire avec un paradigme PPC afin d'aider les solveurs. Le Tableau 3 introduit une écriture PPC des contraintes de coordination et leur formalisation en PLNE.

Tableau 3 Contraintes de coordination en formulations PLNE et PPC

	Contrainte	PLNE	PPC
1	Visites en disjonction	$st_i + Pt_i - M(1 - b_{ij}) \leq st_j$ $st_j + Pt_j - Mb_{ij} \leq st_i$	Cumulative(st, 1, 1)
2	Visites synchronisées	$st_i = st_j$	$st_i = st_j$
3	Visites ayant une durée d'exécution en commun	$st_i + (T_c - Pt_j) \leq st_j$ $st_j + (T_c - Pt_i) \leq st_i$ $b_{ij} + b_{ji} = 1$	$max(st_i, st_j) -$ $min(st_i + Pt_i, st_j + Pt_j) \geq T_c$
4	Visite avec un intervalle particulier	$(Cl - Pt_i) \leq st_i$ $0 \geq st_i - CE$	$st_i \in [CL - Pt_i;\ CE]$
5	Visites conjonctives	$st_i + Pt_i \leq st_j$	$st_i + Pt_i \leq st_j$
6	Time-lag minimal conjonctif	$st_i + TL^-_{i,j} \leq st_j$	$st_i + TL^-_{i,j} \leq st_j$
7	Time-lag minimal disjonctif	$st_i + TL^-_{i,j} - M(1 - b_{ij}) \leq st_j$ $st_j + TL^-_{j,i} - M(1 - b_{ji}) \leq st_i$ $b_{ij} + b_{ji} = 1$	$st_i + TL^-_{i,j} \leq st_j$ $\vee$ $st_j + TL^-_{j,i} \leq st_i$
8	Time-lag maximal conjonctif	$st_i - TL^+_{i,j} \geq st_j$ $st_i \leq st_j$	$st_i - TL^+_{i,j} \geq st_j$ $st_i \leq st_j$
9	Time-lag maximal disjonctif	$st_i - TL^+_{i,j} + M(1 - b_{ij}) \geq st_j$ $st_i - M(1 - b_{ij}) \leq st_j$ $st_j - TL^+_{j,i} + M(1 - b_{ji}) \geq st_i$ $st_j - M(1 - b_{ji}) \leq st_i$ $b_{ij} + b_{ji} = 1$	$(st_i - TL^+_{i,j} \geq st_j \wedge\ st_i \leq st_j)$ $\vee$ $(st_j - TL^+_{j,i} \geq st_i \wedge st_j \leq st_i)$

La contrainte de coordination des visites en disjonction (seconde ligne du Tableau 3) utilise la contrainte globale Cumulative. Une contrainte Cumulative (Aggoun and Beldiceanu, 1993; Simonis et al., 1995) est une contrainte d'ordonnancement sous contrainte de ressources. La contrainte Cumulative est semblable à la contrainte DiffN présentée dans le chapitre 3. Les deux contraintes ont pour objectif de positionner un ensemble de tâches sur un diagramme de Gantt (en général, la terminologie issue du domaine de la PPC est « tâche », mais celle-ci peut également être une « activité », ou une « opération » ou une « visite »). Une tâche est définie par sa durée et par sa consommation en ressources. Les contraintes DiffN et Cumulative imposent que les tâches soient en disjonction.

Lorsqu'une tâche est planifiée, elle consomme une quantité de ressources pendant un temps donné, cette ressource est rendue à la fin de l'exécution de la tâche. Durant la durée d'exécution de la tâche, la quantité de ressources ne peut pas être consommée par une autre tâche. La Figure 18 illustre la planification d'une tâche $v1$ pendant 20 unités de temps. Cette tâche consomme trois unités de ressources.

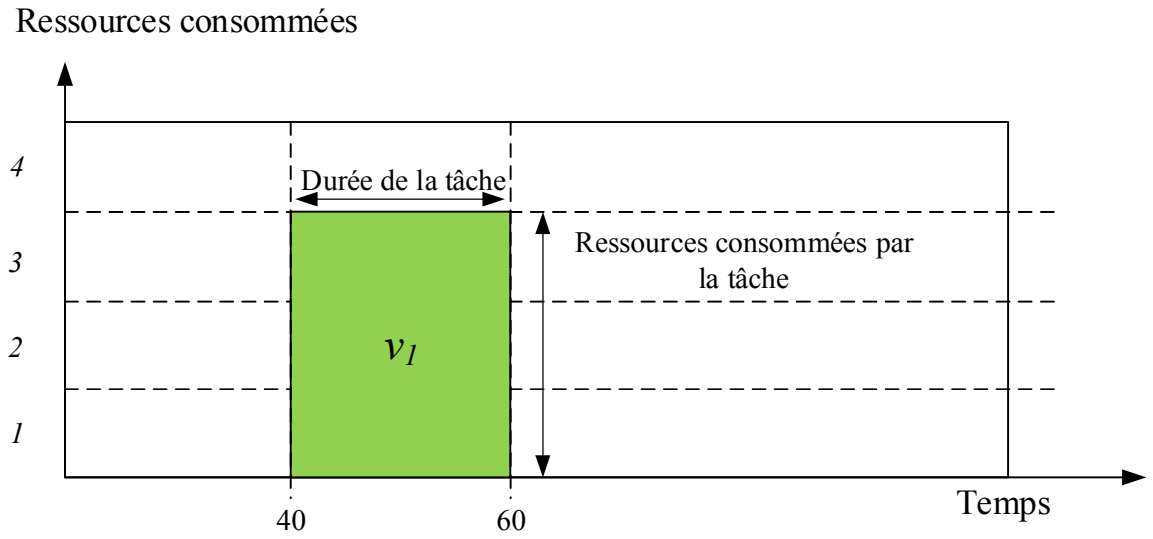


Figure 18 Consommation de ressources par une tâche

Deux tâches ne peuvent pas se superposer (ou se chevaucher) sur le diagramme de Gantt, car la superposition de deux tâches signifie qu'elles consomment en même temps les mêmes ressources. Par exemple, sur la Figure 19, les tâches v_1 et v_2 se chevauchent, ce qui est interdit par les contraintes DiffN et Cumulative.

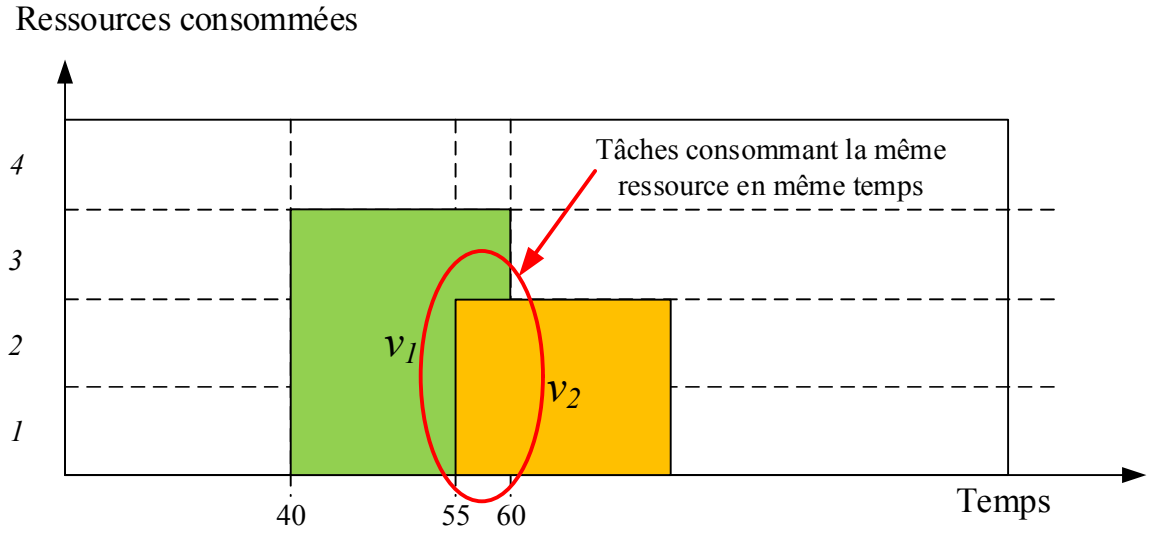


Figure 19 Tâches avec chevauchement

La différence entre une contrainte Cumulative et une contrainte DiffN est que la contrainte Cumulative est un ordonnancement à une dimension suivant l'axe des abscisses et ne prend pas en compte le placement des tâches sur l'axe des ordonnées (il n'y a pas de variable associée à cet axe). À l'inverse, la contrainte DiffN est un ordonnancement à deux dimensions (il y a des variables associées à l'axe des abscisses et à l'axe des ordonnées). La Figure 20 illustre la différence entre les deux types de contrainte. La partie haute de la figure représente une contrainte Cumulative, et la partie basse une contrainte DiffN. Pour la contrainte Cumulative, il n'y a pas de placement suivant l'axe des ordonnées. La tâche v_2 est planifiée de la date 0 à la date 50. Cette tâche est « découpée » en plusieurs parties modélisant la consommation de la ressource aux différents instants. Pour la contrainte DiffN, l'axe des

ordonnées est associé à une variable du problème. La tâche v_2 est caractérisée par un couple $(x\ ;\ y)$ qui vaut $(0\ ;3)$. La tâche v_2 ne peut pas être divisée.

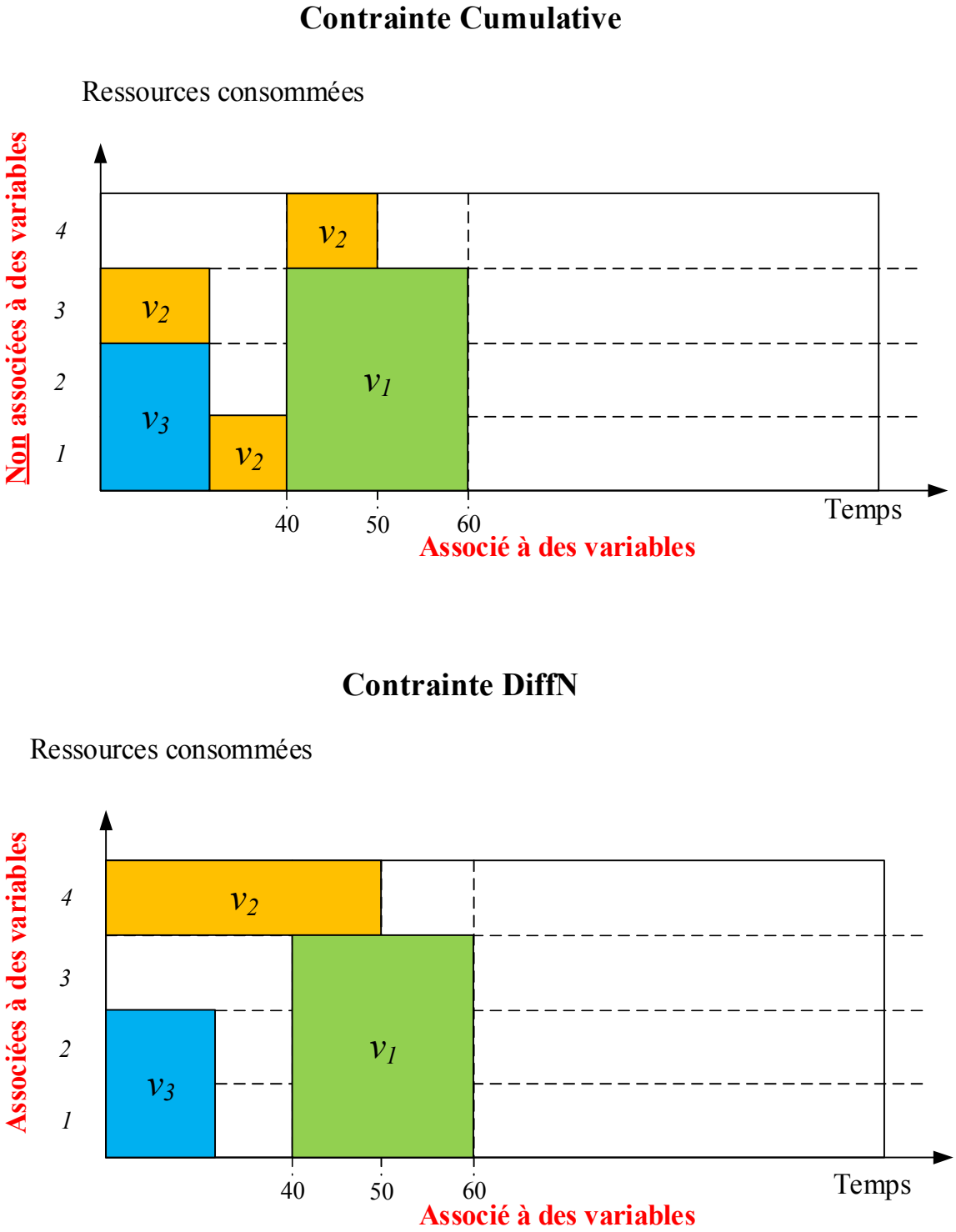


Figure 20 Contrainte Cumulative et contrainte DiffN

Pour le GWSRP, la contrainte de coordination numéro 1 concerne des visites en disjonction, et peut donc être modélisée par une contrainte DiffN ou Cumulative. Deux visites en disjonction ne peuvent pas être planifiées à la même date, et donc se chevaucher sur un diagramme de Gantt. Ce cas est équivalent à une contrainte Cumulative avec une ressource,

en quantité unitaire, et où les visites ont des consommations unitaires de la ressource. Aucune variable n'est associée à l'axe des ordonnées, donc une contrainte Cumulative est suffisante par rapport à une contrainte DiffN. Dans le cas du GWSRP, la ressource peut être vue comme un employé (fictif) qui ne peut pas faire les différentes visites en même temps. La Figure 21 illustre l'utilisation d'une contrainte Cumulative pour modéliser les visites en disjonction.

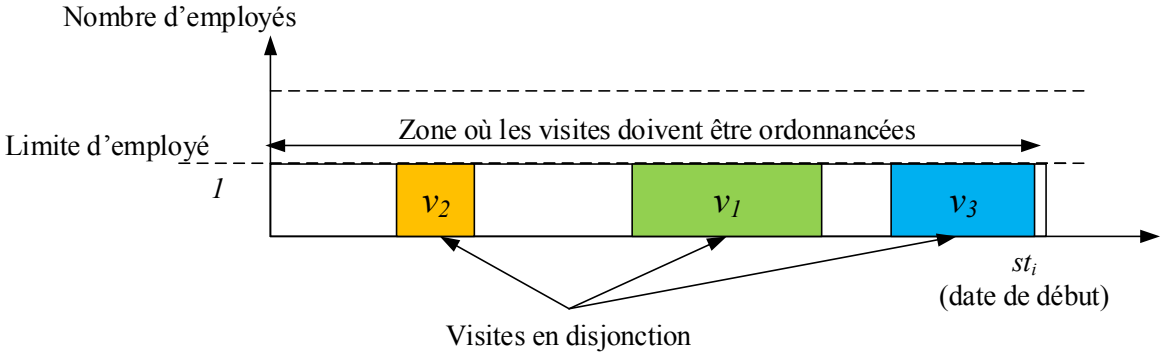


Figure 21 Contrainte Cumulative pour visites en disjonction

L'avantage d'utiliser une contrainte Cumulative est, d'une part, de limiter le nombre de contraintes, et d'autre part, d'offrir au solveur une vision globale du problème afin de mieux propager les informations. Les contraintes globales sont à privilégier afin d'obtenir des modèles PPC efficaces (Rossi et al., 2006; Bourreau et al., 2020).

2.4. Contraintes de coordination dans la littérature

Le Generalised Workforce Scheduling and Routing Problem (GWSRP) est une généralisation du WSRP et prend en compte les neuf contraintes de coordination suivantes : visites en disjonction, visites synchronisées, visites avec une durée d'exécution en commun, visites avec un intervalle de temps particulier, visites conjonctives, time-lag minimal conjonctif, time-lag minimal disjonctif, time-lag maximal conjonctif, time-lag maximal disjonctif.

Le Tableau 4 est un résumé des contraintes de coordination prises en compte dans la littérature. Le « D. » signifie disjonctif, le « S. » signifie synchronisation, et « C. » conjonctif. TL est l'abréviation de time-lag.

Les contraintes de synchronisation sont prises en compte par tous les articles recensés qui traitent de problèmes de tournées de véhicules avec coordination. (Dohn et al., 2011; Rasmussen et al., 2012; Mankowska et al., 2014) sont les trois seuls articles qui s'intéressent à des contraintes de coordination avec des time-lags minimaux et des time-lags maximaux. Mais ces problèmes sont éloignés du GWSRP car ils ne prennent pas en compte la qualité de service pour les employés (concernant la violation des fenêtres de temps des employés et leurs régions préférées).

Tableau 4 Contraintes de coordination prises en compte suivant les articles

Nom de la contrainte	**Visites D.**	**Visites S.**	**Durée commune**	**Intervalle**	**Contrainte C.**	**TL Min C.**	**TL Min D.**	**TL Max C.**	**TL Max D.**	**TL Min+ TL Max C.**	**TL Min+ TL Max D.**
Numéro de contrainte	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11.**
(Eveborn et al., 2006)		X									
(Bredström and Rönnqvist, 2007)		X									
(Bredström and Rönnqvist, 2008)		X									
(Dohn et al., 2011)		X	X			X		X		X	
(Rasmussen et al., 2012)		X	X			X		X		X	
(Afifi et al., 2013)		X									
(Rousseau et al., 2013)		X									
(Labadie et al., 2014)		X									
(Mankowska et al., 2014)		X				X		X			
(Afifi et al., 2016)		X									
(Haddadene et al., 2016)		X									
(Hojabri et al., 2018)		X									
(Liu et al., 2018)		X									
(Masmoudi and Cheikhrouhou, 2018)		X									
(Parragh and Doerner, 2018)		X									
GWSRP	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

3. Difficulté de l'évaluation d'un graphe du GWSRP

Les tournées du GWSRP peuvent se modéliser sous la forme d'un graphe, où chaque visite est un sommet du graphe et chaque ligne représente une tournée. De ce point de vue, le GWSRP qui relève de la communauté « routing » est très proche des problèmes de type ordonnancement de la communauté « ordonnancement ». Il existe de nombreuses fonctions d'évaluation à la fois dans le domaine de l'ordonnancement et dans le domaine des tournées permettant d'obtenir les dates de début des visites. Certaines sont présentées dans les chapitres précédents, notamment la fonction d'évaluation d'une tournée du WSRP.

Le chapitre 3 du manuscrit met en évidence la difficulté à calculer les dates de début d'une tournée de type WSRP afin de minimiser le nombre de violations des fenêtres de temps de l'employé affecté à la tournée. Un algorithme dynamique est proposé afin de trouver les dates de début des visites qui minimisent le coût de la tournée, car les algorithmes classiques d'évaluation de graphe ne prennent pas en compte le critère de qualité de service tel qu'il est défini par (Castillo-Salazar et al., 2016). Pour le WSRP, les tournées sont indépendantes les unes des autres, et l'algorithme dynamique présenté au chapitre 3 peut être appliqué itérativement aux différentes tournées. La Figure 22 illustre des tournées du WSRP, qui sont indépendantes les unes des autres.

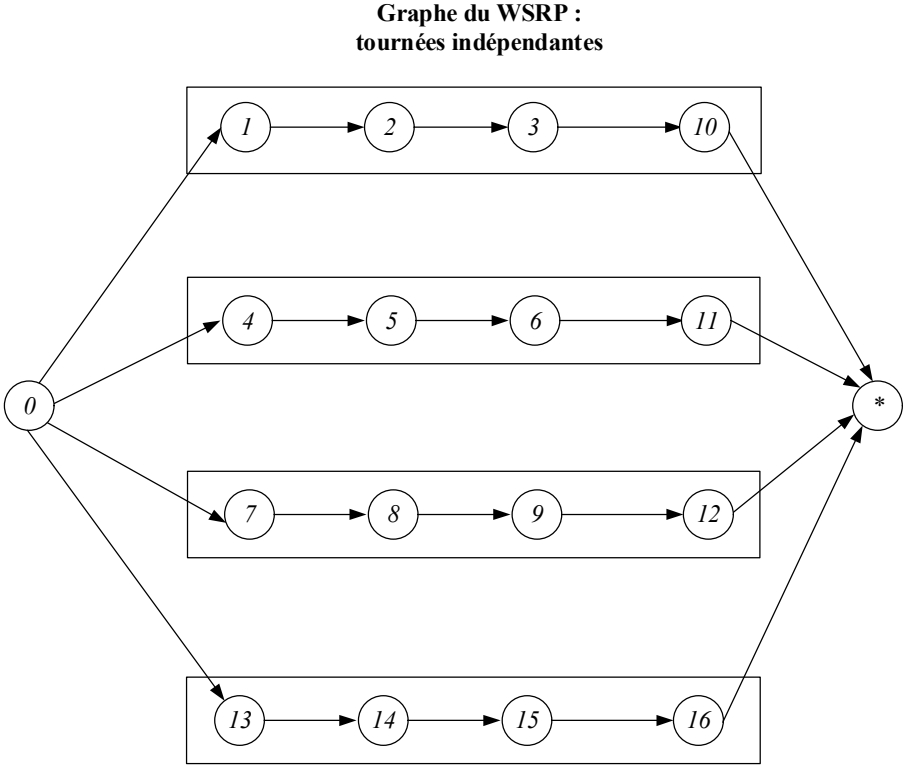


Figure 22 Tournées indépendantes les unes des autres pour le WSRP

Le critère de qualité de service, du WSRP, concernant l'employé est conservé pour le GWSRP. Ainsi chaque visite commençant ou finissant hors de la fenêtre de temps de l'employé est pénalisée. Toutefois, pour le GWSRP, les tournées sont dépendantes les unes des autres comme le montre la Figure 23. Les différentes contraintes de coordination sont

mises en avant dans différentes couleurs, ce qui permet de visualiser les interactions entre les différentes tournées.

L'évaluation d'une tournée dépend de l'évaluation des autres tournées. Minimiser une tournée indépendamment des autres tournées ne garantit l'obtention ni d'une solution respectant toutes les contraintes de coordination, ni de la solution globale de coût minimal. En d'autres termes, obtenir localement les dates de début optimales pour chaque tournée, ne permet pas d'obtenir globalement les dates de début optimales pour toutes les tournées. Il est donc important de calculer simultanément les dates de toutes les tournées.

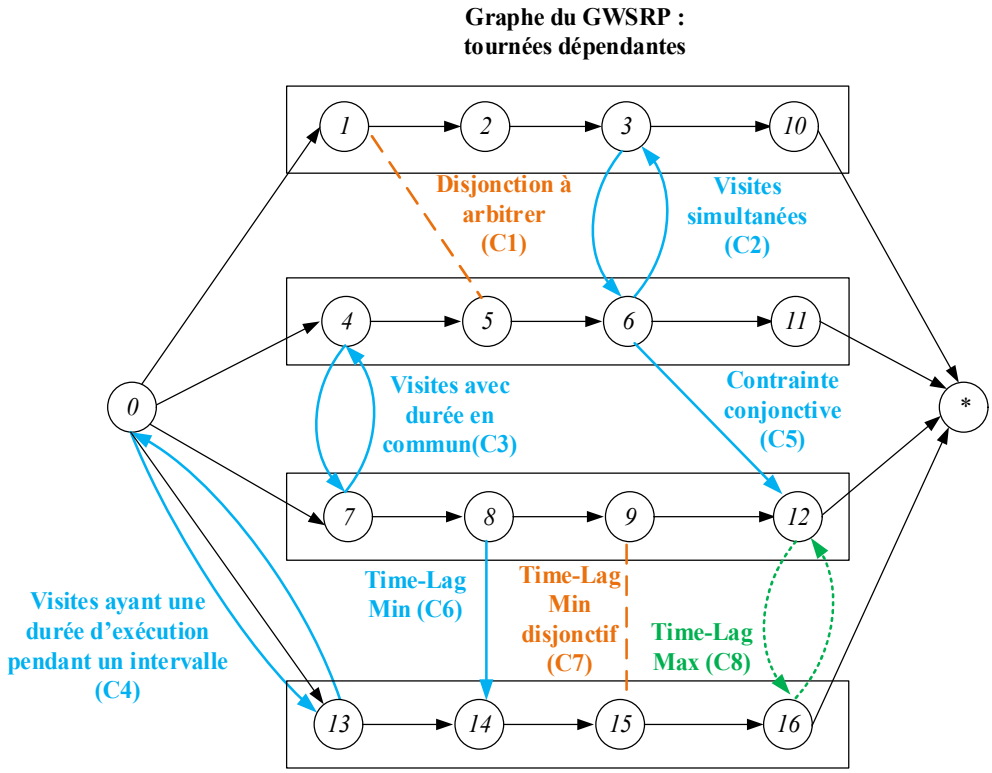


Figure 23 Tournées dépendantes les unes des autres pour le GWSRP

L'évaluation du graphe du GWSRP doit prendre en compte : la maximisation de la qualité de service des employés telle qu'elle est décrite pour le WSRP ; le respect des contraintes de coordination ; et l'arbitrage des contraintes disjonctives.

L'évaluation du graphe par une procédure dynamique semblable à celle introduite dans le chapitre 3 se heurte à quelques difficultés car le problème est très contraint à cause des fenêtres de temps d'une part, et à cause des contraintes de coordination d'autre part.

Les nombreuses contraintes de coordination peuvent introduire des cycles dans le graphe, il est nécessaire d'avoir une procédure qui permet d'éviter les cycles. La présence d'arêtes disjonctives nécessite que celles-ci soient arbitrées. Ces contraintes imposent soit un arc de i vers j, soit un arc de j vers i. Il est possible que certaines orientations aboutissent à des graphes cycliques. Cette difficulté est régulièrement traitée dans les problèmes d'ordonnancement de type Job-shop (voir le chapitre 1 et le chapitre 2), grâce à une représentation indirecte des gammes (ou des tournées dans le cas du GWSRP) sous forme de vecteur de Bierwirth. À notre connaissance, aucune représentation indirecte n'est présente dans la littérature lorsque des arcs de coût négatifs existent.

Par ailleurs, il est important de noter que la procédure dynamique introduite au chapitre 3, pour le WSRP, est basée sur un algorithme de propagation de labels. Dans cette procédure, un label est dépendant de la date de début de la visite, et la propagation d'un label peut générer la création de deux nouveaux labels. La date de début de la visite (et donc le label) dépend des pénalités de la fonction objectif. Pour le GWSRP, la date de début d'une visite contenue dans le label dépend, simultanément, des pénalités et des contraintes de coordination. Cette double dépendance conduit à générer un grand nombre de labels chaque fois qu'un label est propagé et ceci sans qu'une règle de dominance puisse permettre d'éliminer efficacement les "bons"' labels.

L'utilisation d'un graphe disjonctif et d'une telle fonction d'évaluation obligerait, de plus, à redéfinir les opérateurs de recherche locale classiques des problèmes d'ordonnancement ou de tournées de véhicules. Le problème est très contraint à cause des fenêtres de temps et des contraintes de coordination, les opérateurs de recherche locale auraient sûrement beaucoup de difficulté à modifier le graphe tout en le conservant acyclique et évaluable pour obtenir une solution.

Il faut noter que ces difficultés sont clairement définies dans (Caumond et al., 2008) pour le Job-shop avec time-lags maximal qui est un cas particulier du GWSRP. Dans cet article, un time-lag relie deux opérations appartenant à la même gamme. (Lacomme et al., 2011) et (Lacomme et al., 2012) étudient les cas où les time-lags sont génériques et peuvent relier des opérations appartenant à des gammes différentes. (Karoui et al., 2010) puis (Artigues et al., 2011) proposent des approches basées sur la Programmation Par Contraintes (PPC) pour résoudre les problèmes de Job-shop avec time-lags. Ces approches sont plus performantes que les approches heuristiques issues de (Caumond et al., 2008; Lacomme et al., 2011, 2012). L'article de (Artigues et al., 2011), dont la méthode de résolution est basée sur la PPC, est un élément moteur pour tester ce type d'approche sur le GWSRP qui est une généralisation du Job-shop avec time-lags.

Devant la difficulté d'une procédure dynamique pour évaluer le graphe du GWSRP, celui-ci est évalué par un solveur de Programmation Par Contraintes (PPC), basé sur le modèle introduit pour le WSRP au chapitre 3, et enrichi des contraintes de coordination présentées à la section 2.3 de ce chapitre. Généralement, la PPC fournit très rapidement une solution, et elle se montre très efficace dans les problèmes très contraints (Bockmayr and Hooker, 2005; Pour et al., 2018; Bourreau et al., 2020). De plus, la PPC permet également d'orienter les arcs disjonctifs et peut servir de recherche locale (Rousseau et al., 2013; Hojabri et al., 2018).

4. Méthode de résolution

La résolution du GWSRP est basée sur un schéma algorithmique nommé « Constraint-Programming based Decomposition Method (CPDM) » qui se compose de trois étapes successives : les deux premières concernent la création de tournées sans prendre de contraintes de coordination en compte, tandis que la troisième permet l'insertion de ces contraintes dans les tournées.

4.1. Schéma global de la CPDM

La Constraint-Programming based Decomposition Method (CPDM) est une approche tirant parti d'un schéma basé sur une Génération de Colonnes (GC) qui vise à créer des tournées en relâchant les contraintes de coordination ; suivit d'une phase itérative comparable à une recherche locale réalisée par un modèle de Programmation Par Contraintes (PPC). Cette dernière phase a pour objectif d'insérer les contraintes de coordination dans les tournées. La CPDM combine une GC et un modèle PPC (Bourreau et al., 2019a). En effet, la génération de colonnes est une approche connue comme étant efficace pour résoudre les problèmes d'ordonnancement et/ou de transport ; tandis que la PPC permet généralement de trouver très rapidement des solutions réalisables dans un problème très contraint (Backer et al., 2000; Bockmayr and Hooker, 2005; Pour et al., 2018; Bourreau et al., 2020).

La GC permet d'obtenir un ensemble de tournées indépendantes les unes des autres qui sont modélisées par un graphe. Le modèle PPC permet simultanément : d'insérer les Contraintes de Coordination (CC) dans le graphe ; de calculer les dates des visites minimisant le coût de la solution ; et de modifier légèrement les tournées afin soit d'améliorer la solution, soit de prendre en compte de nouvelles contraintes de coordination qui ne peuvent pas être insérées dans le graphe obtenu à la fin de la GC sans modification de celui-ci.

La CPDM (Constraint-Programming based Decomposition Method) se compose de trois étapes (Figure 24) :

- **Construction de tournées initiales** : cette première étape a pour objectif la construction d'un grand nombre de tournées sans prendre en compte les contraintes de coordination. Cette étape correspond à une génération de colonnes résolue à la racine (la solution peut être fractionnaire, aucun Branch-and-Price n'est effectué), qui est limitée par la durée *SCGlimitTime*. Cette étape retourne un ensemble TGC de colonnes assimilables à des tournées, chacune possédant un coût et étant affecté à un employé. Cet ensemble TGC doit contenir de nombreuses tournées ($|TGC| > |W|$, où W est l'ensemble des employés) afin de favoriser la suite de la CPDM. L'étape 1 est détaillée dans la section 4.2.
- **Sélection des tournées** : cette seconde étape de la CPDM a pour objectif de trouver une solution entière au problème du GWSRP sans contrainte de coordination (donc au problème de WSRP). Cette procédure prend en entrée un ensemble TGC de tournées (avec $|TGC| > |W|$) et retourne un sous-ensemble t de tournées avec $|t| = |W|$, où chaque employé possède une tournée (certaines tournées peuvent être vides). Cette phase est équivalente à la résolution d'un Set Partitioning Problem (SPP) en nombres entiers sur l'ensemble des tournées TGC issues de l'étape 1. La fonction objectif du SPP consiste à minimiser le coût de la solution et à maximiser le nombre de contraintes prises en compte afin d'obtenir une solution favorisant pour la dernière étape de la CPDM. En plus de retourner l'ensemble des tournées t, cette étape retourne donc l'ensemble $ChCC$ des contraintes de coordination qui sont respectées par les tournées contenues dans t, et l'ensemble $UnCC$ des contraintes de coordination qui ne sont pas respectées par les tournées issues de t. La résolution du SPP est limitée dans le temps par *SClimitTime*. L'étape 2 est détaillée dans la section 4.3.
- **Insertion des contraintes de coordination** : la troisième étape a pour objectif de trouver une solution respectant toutes les contraintes de coordination. Cette solution est construite itérativement à partir des tournées t obtenues par l'itération précédente. La troisième étape est réalisée par un solveur PPC, dont le temps d'exécution est limité. Cette procédure est donc appelée plusieurs fois (*iter_max*), ce qui permet d'explorer différentes parties de l'arbre de recherche. De plus, cette phase de la CPDM fait appel à la méthode ICPA

(*Iterative Constraint-Programming based Algorithm*) qui essaie d'insérer incrémentalement les contraintes de coordination dans les tournées. En effet, introduire toutes les contraintes d'un seul coup ne permet pas de trouver de solution. L'étape 3 est détaillée dans la section 4.4.

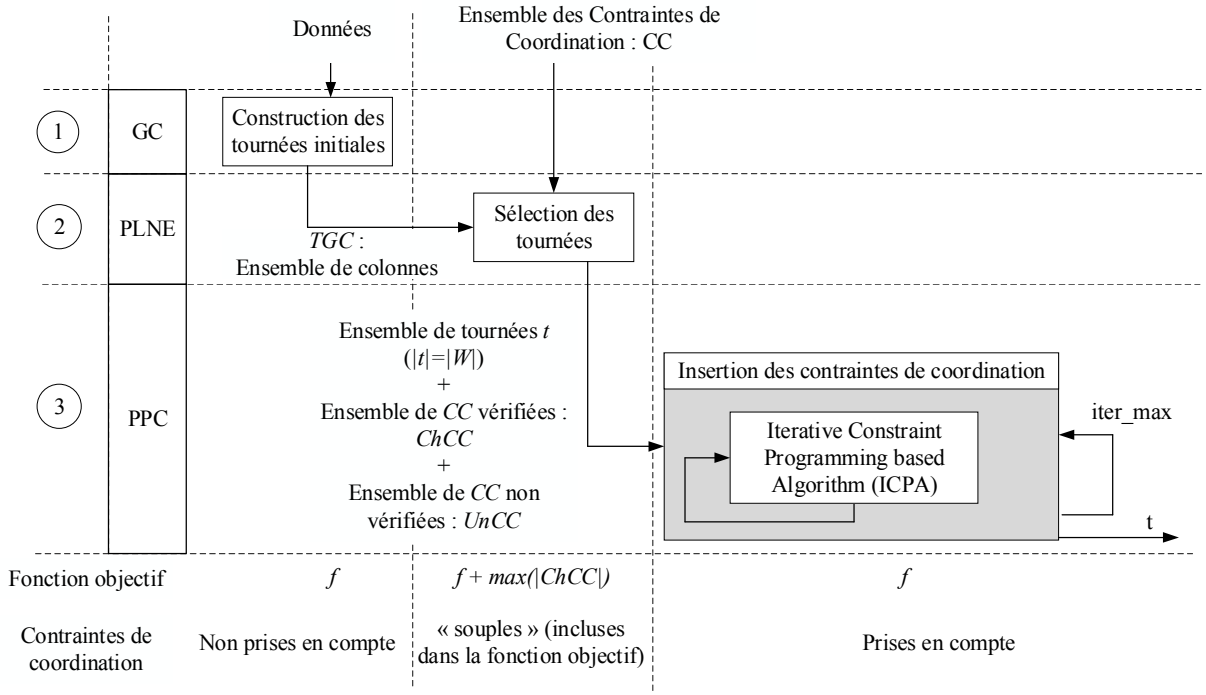


Figure 24 Schéma global de la CPDM

D'autres schémas itératifs ont été testés, mais ils n'ont pas abouti à des améliorations significatives des résultats. Ces schémas sont présentés en Annexes.

L'Algorithme 1 est l'algorithme de principe de l'approche CPDM. Celui-ci reçoit cinq arguments en entrée : CC est l'ensemble des contraintes de coordination (ligne 6) ; $Data$ est l'ensemble des données d'un problème de GWSRP sans les contraintes de coordination (ligne 5) ; *SCGlimitTime* est la durée limitée de l'étape 1 (ligne 7) ; *SClimitTime* est la durée maximale de l'étape 2 (ligne 8) ; *iter_max* est le nombre maximal d'itérations de l'étape 3 (ligne 9). CPDM retourne l'ensemble t des tournées de la solution (ligne 2) et le coût $Cost$ de cette solution (ligne 3).

La première étape de la CPDM consiste à construire des tournées initiales (ligne 15) qui ne prennent pas en compte les contraintes de coordination. La seconde étape est la sélection des tournées (ligne 16). La troisième étape, qui correspond à l'insertion des contraintes de coordination (CC), est itérée *iter_max* fois (ligne 17 à 19). Chaque étape est présentée en détail dans les sections suivantes.

Algorithme 1 : Constraint-Programming based Decomposition Method

Sortie
t : ensemble des tournées
Cost : coût de la solution
Entrée
Data : données des instances
CC : ensemble des contraintes de coordination
SCGlimitTime : temps limite de l'étape 1 (GC)
SClimitTime : temps limite de l'étape 2
iter_max : nombre maximal d'itérations pour l'étape 3
Variable
$ChCC$: contraintes de coordination vérifiées par les tournées t
$UnCC$: contraintes de coordination non vérifiées par les tournées t
TGC : ensemble des colonnes issues de la GC
Début
| TCG = Contruction_tournees_initiales(Data, SCGlimitTime) // Étape 1
| (t, $ChCC$, $UnCC$) = Selection_des_tournees(CC, TCG, SClimitTime) // Étape 2
| **Pour** $i = 1, \ldots,$ Max_Iter **Faire** // Étape 3
| | (cost, t, $ChCC$, $UnCC$)= Insertion_des_CC(t, $ChCC$, $UnCC$)
| **FinPour**
| **Retourner (t, Cost)**
Fin

4.2. Construction des tournées initiales

La première étape de la CPDM est la construction de tournées initiales qui ne prennent pas en considération les contraintes de coordination. Le problème traité est donc équivalent au WSRP. Les tournées sont engendrées par la Génération de Colonnes (GC), présentée dans le chapitre 3, et brièvement rappelée ici.

Une colonne représente une tournée pour un employé donné. Le problème maître est équivalent à un Set Partitionning Problem, où une visite ne peut être sélectionnée qu'une unique fois (contrainte 1), et où, parmi toutes les tournées appartenant à l'employé w, une seule peut être sélectionnée (contrainte 2). Le problème maître issu du chapitre 3 est rappelé ci-dessous.

$$f_{master} = Min \sum_{k \in \Omega} CT_k \, y_k$$

$$s.t.$$

$$\sum_{k \in \Omega} \Delta_k^i y_k = 1 \qquad \forall i \in V \qquad (1)$$

$$\sum_{k \in \Omega | E_k = w} \Delta_k^i \, y_k \leq 1 \qquad \forall w \in W \qquad (2)$$

V est l'ensemble des visites ; W l'ensemble des employés ; Ω l'ensemble des colonnes ; CT_k le coût de la colonne k ; E_k l'employé réalisant la tournée de la colonne k ; Δ_k^i une

valeur binaire valant 1 si la visite i est réalisée par la tournée de la colonne k, et 0 sinon. Enfin, y_k est une variable binaire valant 1 si la colonne est sélectionnée, et 0 sinon.

Le problème esclave, pour un employé w donné, correspond à un Elementary Shortest Path Problem with Resource Constraints (ESPPRC), incluant la qualité de service pour les employés. Cette dernière étant dépendante des dates de début des visites, le problème esclave est résolu par la procédure de programmation dynamique qui est introduite dans le chapitre 3.

Dans la formulation du problème maître donnée au chapitre 3, le problème maître ne prend pas en compte les dates de début des visites. En effet, une colonne modélise une tournée dont il est garanti que les dates de passage sur les visites sont compatibles avec les fenêtres de temps. Le problème maître est équivalent à un Set Partitionning Problem. Pour prendre en considération les contraintes de coordination dans la GC, il faudrait que le problème maître prenne en compte des dates de début des visites, et pour les calculer, des contraintes supplémentaires devraient être insérées dans le problème maître. La résolution du problème maître nécessiterait alors plus d'efforts de calculs. Par conséquent, les variables duales seraient elles aussi dépendantes du temps, et lors de la résolution du problème esclave, le coût d'une tournée dépendrait, entre autres, du nombre de violations de la fenêtre de temps de l'employé et des dates de début de chaque visite. La résolution du problème esclave très couteuse en temps rend difficile l'utilisation pratique de ce type d'approche. Le point de vue proposé par la CPDM est profondément différent, dans le sens où il s'agit d'une résolution composée de trois étapes, chacune résolvant une « sous-partie » d'un problème global.

La première étape de la CPDM (qui correspond à une génération de colonnes) permet d'obtenir une solution fractionnaire qui pourrait donner une solution entière avec une approche de type Branch-and-bound, par exemple. Ce type d'approche est relativement couteux et présente pour notre problème un intérêt limité dans le sens où il ne s'agit pas d'obtenir une solution optimale du problème relâché, mais uniquement une solution. Il n'y a pas d'élément permettant d'affirmer qu'une « solution optimale » du problème relâché est facilement « modifiable » à l'étape 3 de la CPDM qui concerne l'ajout des contraintes de coordination. Il est même raisonnable de penser qu'une très bonne solution du problème relâchée contient des tournées où les visites sont probablement ordonnancées à des dates proches les unes des autres. Une solution avec ces caractéristiques risque alors d'être fortement impactée par l'ajout des contraintes de coordination et ne pas permettre une insertion des contraintes uniquement par légères modifications des tournées.

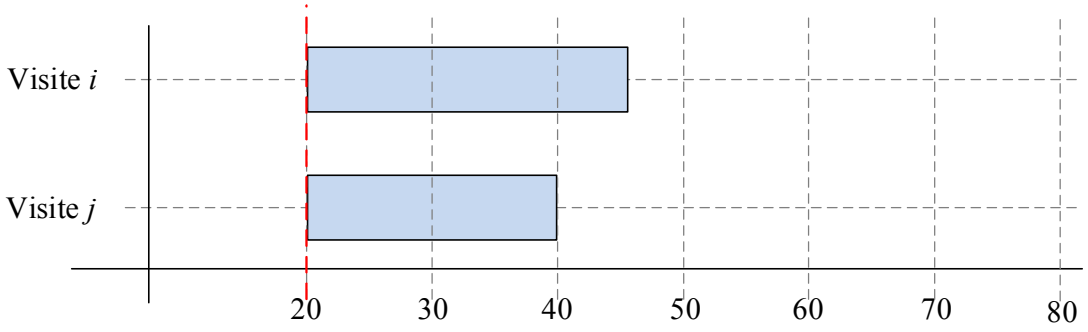
Ces considérations nous poussent à définir une seconde étape qui est « sous-optimale » d'un point de vue théorique, mais suffisante pour le schéma d'optimisation global. La prochaine étape de la CPDM permet de fournir une solution en nombres entiers qui sert de donnée d'entrée à l'étape 3. Cette dernière ajoute les contraintes de coordination en éventuellement modifiant « légèrement » les tournées, c'est-à-dire, en autorisant des visites à être déplacées d'une tournée à une autre ou en modifiant l'ordre des visites au sein d'une même tournée. La notion de modification de tournées est définie par la suite.

À la fin de la première phase de la CPDM, un ensemble TGC de colonnes est retournée. Cet ensemble doit contenir un grand nombre de colonnes afin de favoriser la recherche d'une solution de bonne qualité lors des prochaines étapes de la CPDM. En plus d'être rapide, la résolution du problème esclave par programmation dynamique permet d'obtenir plusieurs colonnes par résolution, et est donc intéressante pour l'approche utilisée.

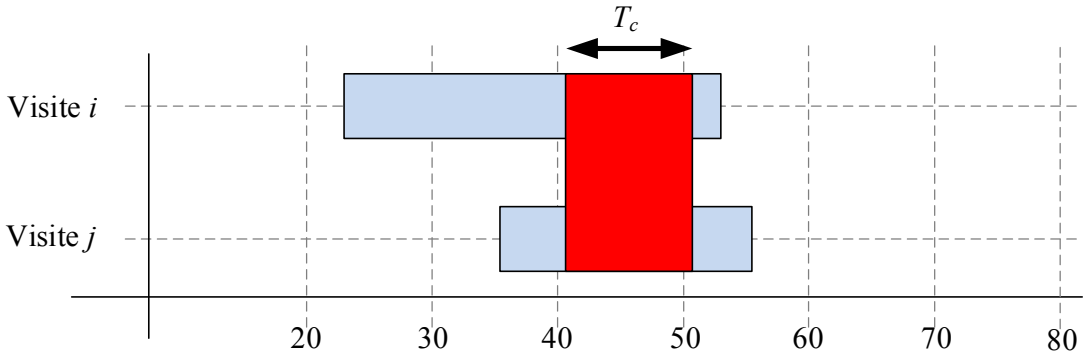
Il est important de garder à l'esprit que l'ensemble TGC de colonnes transmis à la prochaine étape de la CPDM équivaut à un ensemble de tournées réalisables (et affectées à un employé), où aucune contrainte de coordination n'est prise en compte.

Bien que la génération de colonnes ne prenne pas en compte les contraintes de coordination, quelques contraintes peuvent être facilement insérées dans la résolution du problème esclave :

- Pour les contraintes de visites synchronisées (voir la section 2.2.2) ou visites ayant une durée d'exécution en commun (voir la section 2.2.3), les deux visites i et j, concernées par une même contrainte, ne peuvent pas appartenir à la même tournée (Figure 25). Toute colonne comportant deux visites reliées par une contrainte de synchronisation ou de durée d'exécution en commun ne peut pas faire partie de la solution recherchée et ne nécessite donc pas d'être générée.



Visites avec une contrainte de synchronisation (les visites commencent en même temps)



Visites ayant une durée T_c d'exécution en commun

Figure 25 Visites ne pouvant pas être dans la même tournée

- Pour les visites ayant une durée d'exécution au cours d'un intervalle particulier (voir la section 2.2.4 ou la Figure 26 ci-dessous), il a été montré que cette contrainte équivaut à la réécriture de la fenêtre de temps de la visite. Ce type de contrainte peut donc être facilement pris en compte durant la résolution des sous-problèmes.

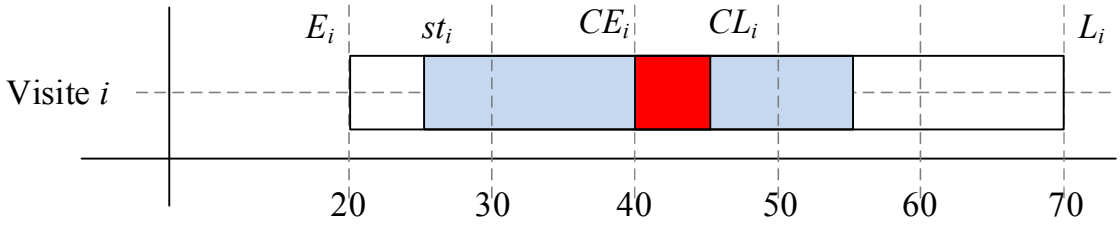
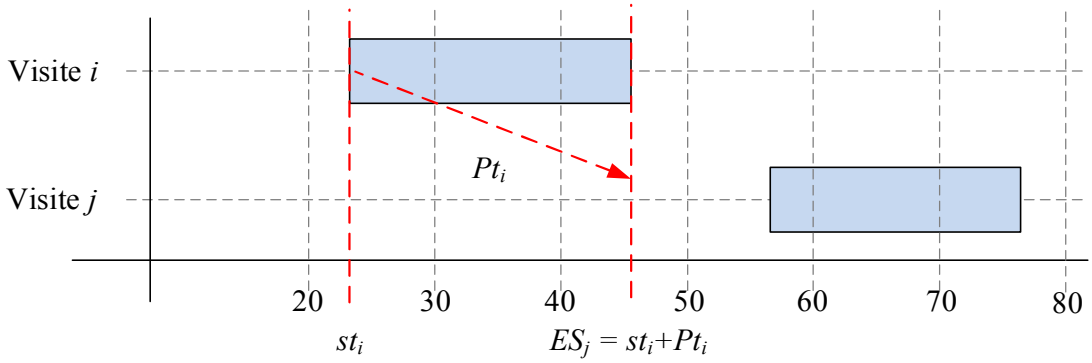
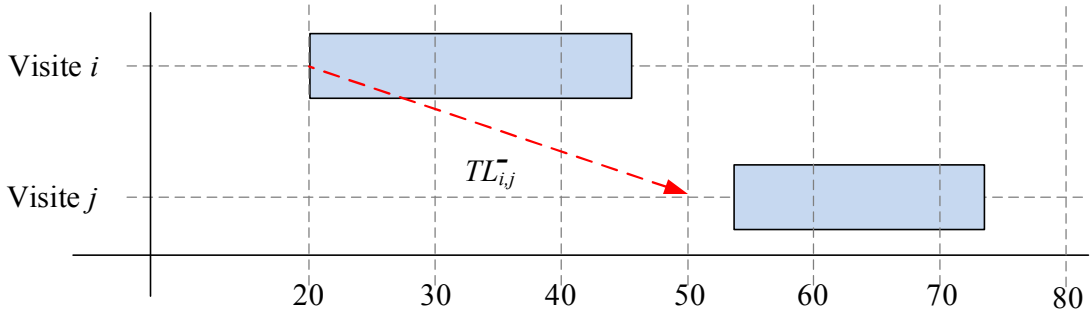


Figure 26 Fenêtre de temps de la visite pouvant être modifiée

- Pour les contraintes avec un ordre relatif entre les visites i et j, c'est-à-dire les contraintes conjonctives (section 2.2.5), les contraintes de time-lag minimal conjonctif (2.2.6) et les contraintes de time-lag maximal conjonctif (2.2.8), il est facile d'imposer au sous-problème de respecter cet ordre si les deux visites sont dans la même tournée (cf.Figure 27). Si une seule des deux visites est présente dans la tournée, alors il n'y a pas besoin de prendre en compte l'ordre relatif.



Visites conjonctives : i ne peut pas être avant j



Time-lag minimal : i ne peut pas être avant j

Figure 27 Fenêtre de temps de la visite pouvant être modifiée

En conclusion, prendre en compte ces quelques considérations permet de laisser le problème maître indépendant du temps, et de ne pas trop complexifier la résolution du problème esclave. Ces contraintes sont indépendantes des dates de début des visites lorsqu'elles sont considérées pour une unique tournée.

4.3. Solution en nombres entiers et sélection des tournées

La génération de colonnes est résolue au nœud racine et ne garantit donc pas de procurer une solution en nombres entiers. En effet, la contrainte d'intégrité est relâchée dans la GC. La GC fournit un ensemble TGC de colonnes, où les employés possèdent plusieurs tournées ($|TGC| > |W|$). La seconde étape de la CPDM permet d'obtenir une première solution par résolution du Set Partitioning Problem (qui est également le problème maître de la GC) en nombres entiers. La Figure 28 présente les trois étapes de la CPDM, et cette section se concentre sur la seconde étape, c'est-à-dire la sélection des tournées parmi celles modélisées par les colonnes à la fin de la GC.

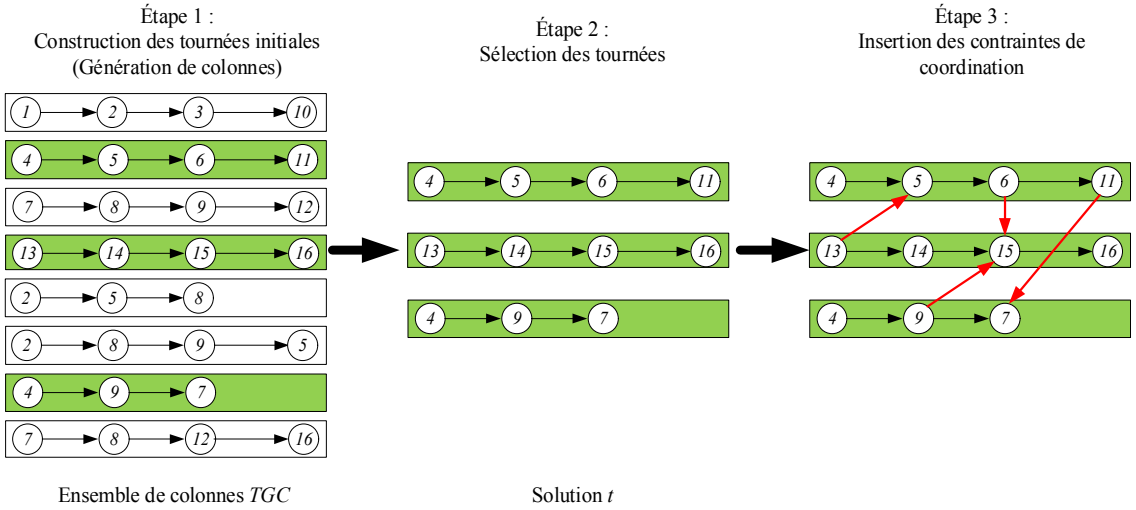


Figure 28 Les trois étapes de la CPDM

Il est évident que la solution de coût minimal, lors de la seconde étape, n'est pas nécessairement celle qui permet d'insérer facilement toutes les contraintes de coordination lors de la troisième (et dernière) étape de la CPDM. En effet, une solution localement optimale ne l'est pas forcément globalement. Toutefois, le choix de la solution retournée à la fin de cette étape peut avoir un grand impact sur la suite de la résolution. Parmi toutes les solutions possibles, une solution qui prend déjà en compte des contraintes de coordination semble être plus intéressante.

C'est dans ce but que la fonction objectif f^{spp} du Set Partitioning Problem (SPP) est hiérarchique et qu'elle maximise en premier le nombre de contraintes de coordination respectées, puis minimise en second le coût de la solution. Une partie des contraintes de coordination est donc respectée à la fin de cette étape. La fonction objectif est alors la suivante :

$$f^{spp} = \Lambda_1 \max(f^v) + \Lambda_2 \min(f)$$

Avec $\Lambda_1 = 1000\Lambda_2$; et

$$f^v = \sum_{c \in CC} VC^c$$

CC est l'ensemble des contraintes de coordination ; VC^c est une variable binaire qui vaut 0 si la contrainte de coordination $c \in CC$ est respectée, et qui vaut 1 sinon.

Lors de la résolution du Set Partitioning Problem (SPP), les tournées contenues dans l'ensemble TGC ne sont pas modifiées, et seules les dates de début des visites sont calculées. L'objectif est de trouver le sous-ensemble de colonnes minimisant le nombre de contraintes de coordination non prises en compte. Les contraintes de coordination prises en compte peuvent appartenir à n'importe laquelle des neuf catégories de contraintes.

Le modèle en PLNE, résolu lors de la seconde étape de la CPDM, est donc :

$$\sum_{k \in TGC} \Delta_k^i \, y_k = 1 \qquad \forall j \in A \qquad (C1)$$

$$\sum_{k \in TGC | E_k = w} y_k \leq 1 \qquad \forall w \in W \qquad (C2)$$

$$st_j + Pt_j + T_{j,j+1} - (y_k + 1)M \leq st_{j+1} \qquad \forall k \in TGC, \forall j \in k \qquad (C3)$$

$$VC^c = \begin{cases} = 0 \text{ si } c \text{ est vérifiée} \\ = 1 \text{ sinon} \end{cases} \qquad \forall c \in CC \qquad (C4)$$

Les deux premières contraintes correspondent au Set Partitioning Problem. La contrainte $(C3)$ calcule les dates de début des visites de la tournée k si celle-ci est sélectionnée. Par abus de langage, j est une visite de la tournée k et $j+1$ est la visite suivante. La contrainte $(C4)$ permet de vérifier si les contraintes sont respectées.

À la fin de cette étape, la solution trouvée est constituée :

- D'un sous-ensemble de tournées t (ou solution, par abus de langage), avec $|t| = |W|$, c'est-à-dire, une tournée pour chaque employé (une tournée peut être vide).
- D'un sous-ensemble $ChCC$ de contraintes de coordination respectées.
- D'un sous-ensemble $UnCC$ de contraintes de coordination qui n'ont pas été prises en compte.

La résolution du PLNE peut être très longue lorsque de nombreuses colonnes sont présentes, mais aussi à cause du grand nombre de variables binaires, elle est donc limitée par une durée *SClimitTime*.

4.4. Insertion des contraintes de coordination

L'objectif de la dernière étape de la CPDM est, d'une part, d'insérer toutes les contraintes de coordination entre les visites, et d'autre part, de trouver si possible une solution de meilleure qualité. La solution obtenue lors de l'étape précédente est le point de départ de cette dernière étape.

L'insertion des contraintes de coordination dans les tournées est similaire à un opérateur de recherche locale qui se concentre sur de légères modifications des tournées afin d'apporter des améliorations en un temps de calcul court. Cette troisième étape est basée sur un modèle de Programmation Par Contraintes (PPC).

L'utilisation d'un modèle et, par conséquent, d'un solveur PPC, démarrant à partir d'une solution, permet de gérer de manière intégrée les deux mécanismes de la recherche locale : la phase d'intensification et la phase de diversification. L'utilisation du modèle PPC permet l'exploration de voisinages (qui peuvent être éloignés), sans imposer une dégradation de la solution pour sortir des optimaux locaux (Backer et al., 2000; Hojabri et al., 2018). Ces mécanismes sont notamment utilisés par (Bourreau et al., 2019b) pour le Job-shop Scheduling Problem.

Le solveur PPC part d'une solution initiale t, c'est-à-dire que le solveur déroule son arbre de recherche et, pour chaque variable, il affecte la valeur de celle-ci correspondant à la solution t. Une solution t est un ensemble de tournées où chaque visite est définie par son affectation à un employé et par son successeur dans la tournée.

Par exemple, si la visite i est affectée à l'employé w dans la solution t, alors lorsque le solveur choisit de brancher sur a_i, il affecte la valeur $a_i = w$. Si la visite i est suivie de la visite j dans les tournées de t, alors lorsque le solveur branche sur s_i, il choisit $s_i = j$. Plus tard, ces choix sont remis en cause lorsque le solveur remonte dans l'arbre. La solution t correspond à la solution située le plus à gauche dans l'arbre de recherche (voir la Figure 29). L'objectif est de trouver une première solution sans réaliser de backtrack dans l'arbre de recherche.

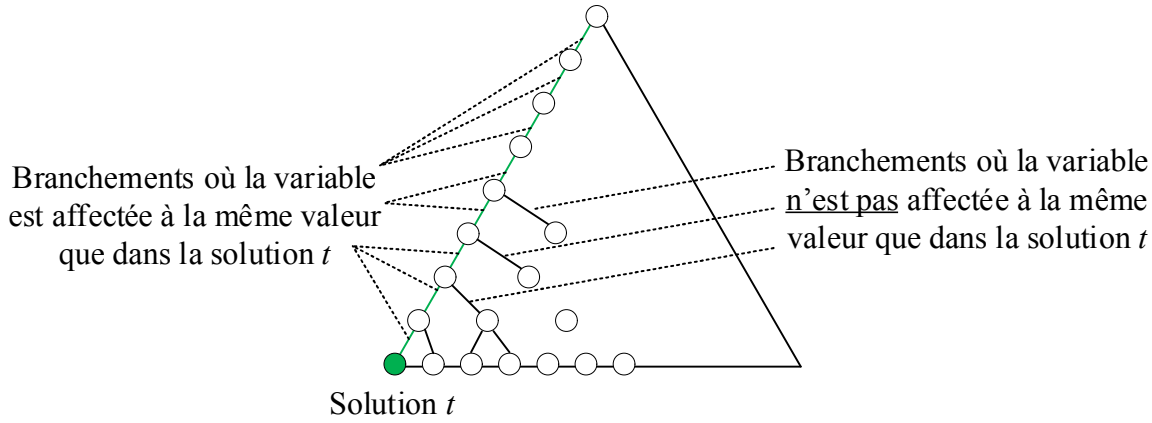


Figure 29 Démarrer d'une solution en PPC

Le parcours de l'arbre de recherche par le solveur correspond à un parcours en profondeur, où les branchements remis en question en premier se situent en bas à gauche de l'arbre (voir la Figure 30). Une fois cette partie de l'arbre explorée, le solveur backtrack et remonte dans l'arbre. Par conséquent les variables sur lesquelles ont été réalisés les premiers branchements ne sont pas remises en question rapidement, tandis que les dernières variables sont remises en cause et plusieurs valeurs possibles sont testées.

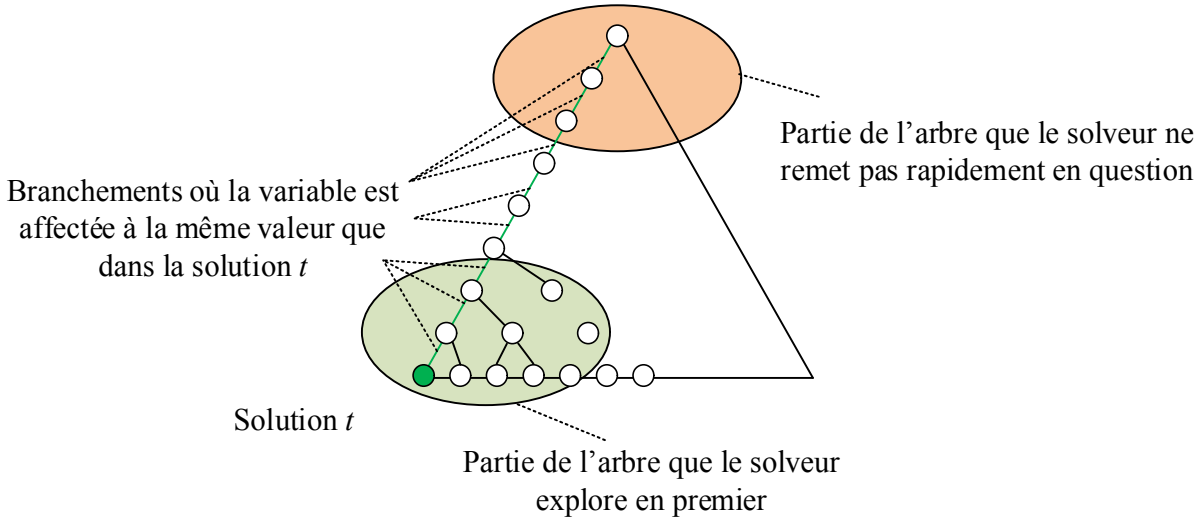


Figure 30 Parcours de l'arbre par le solveur

L'ordre du parcours de l'arbre de recherche par le PPC est présenté en détail dans la Figure 31 et correspond à un parcours en profondeur. Les nœuds de l'arbre sont numérotés en fonction de leur position durant la recherche arborescente.

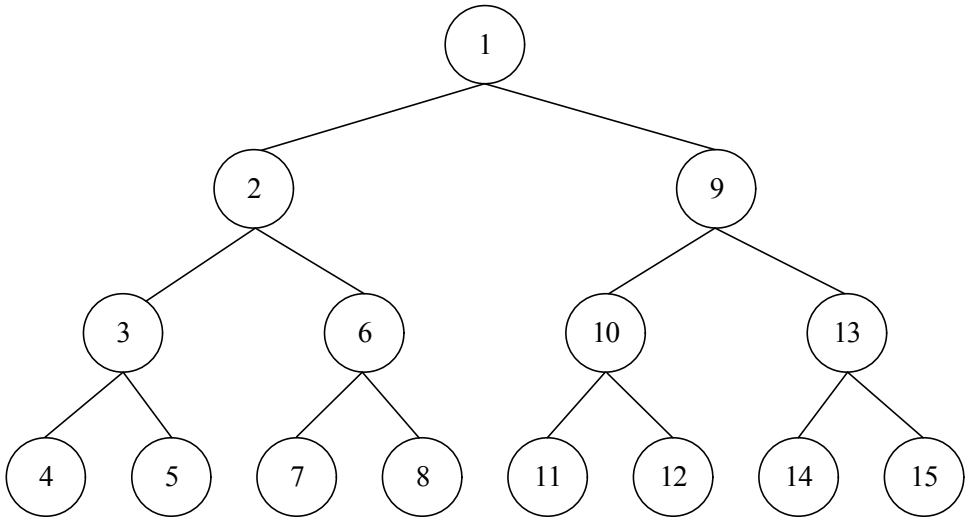


Figure 31 Ordre de parcours des nœuds dans l'arbre de recherche

Au vu du parcours de l'arbre de recherche par le solveur, il semble raisonnable de penser que le solveur est rapidement capable de trouver une solution meilleure (solution en bleu sur la Figure 32) en parcourant uniquement le bas de l'arbre avec peu de bracktracks et de branchements. Le solveur trouve donc une solution optimale localement.

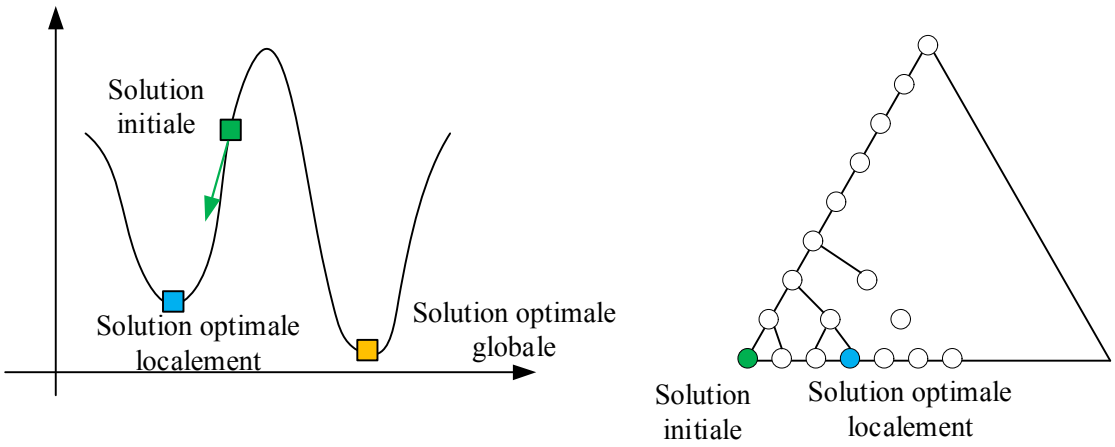


Figure 32 Solution optimale localement par PPC

Une des forces de la PPC est d'élaguer des branches de l'arbre de recherche lorsqu'une solution est trouvée. Cet avantage permet d'obtenir des solutions qui sont meilleures, et d'explorer un voisinage plus lointain. Toutefois, il est nécessaire que le modèle PPC « se rend compte » qu'il est dans sur une solution optimale localement, ce qui suppose de remonter très haut dans les branchements. Le modèle PPC doit donc être suffisamment efficace. Ce phénomène permet de sortir d'une solution optimale localement sans avoir besoin de dégrader la solution comme c'est fréquemment le cas dans les approches de type heuristique (voir la Figure 33).

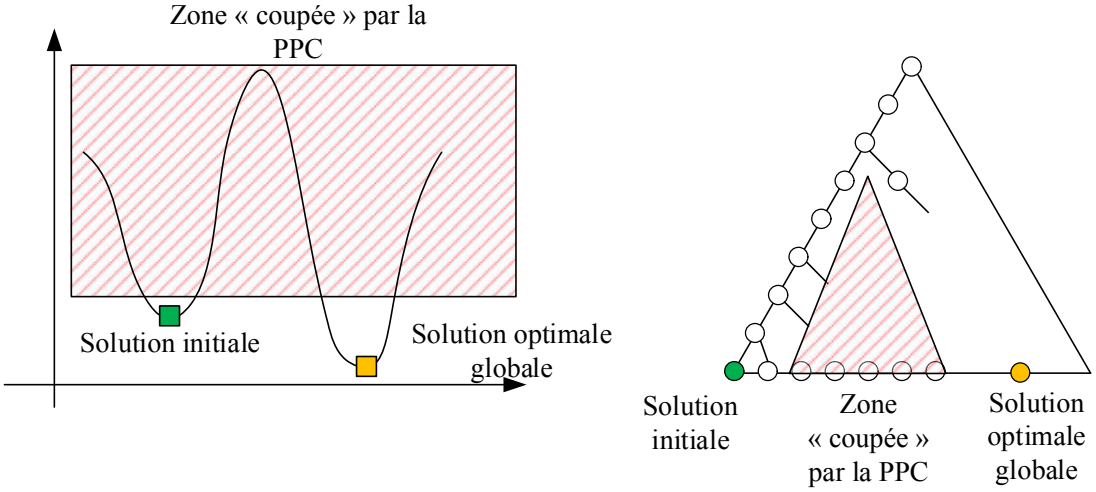


Figure 33 Solution optimale globale par PPC

L'ordre de branchement des variables est très important. Des variables branchées proches de la racine de l'arbre ne sont remises en cause que lorsque toutes les possibilités des variables situées en dessous ont été testées. Une variable en haut de l'arbre a donc un fort impact sur les branchements des autres variables et sur la solution. La Figure 34 illustre les variables situées en haut de l'arbre et celles en bas (partie gauche de la figure), ainsi que la taille de l'espace de recherche qui doit être parcouru avant de remettre en cause une variable. Sur la partie droite de la figure, il est visuellement facile de constater que la taille du sous-arbre induit par le nœud rouge est beaucoup plus grande que le sous-arbre induit par le nœud

vert. Le sous-arbre vert est même inclus dans le sous-arbre rouge (ce qui n'est pas tout à fait le cas sur la figure afin de conserver une certaine clarté).

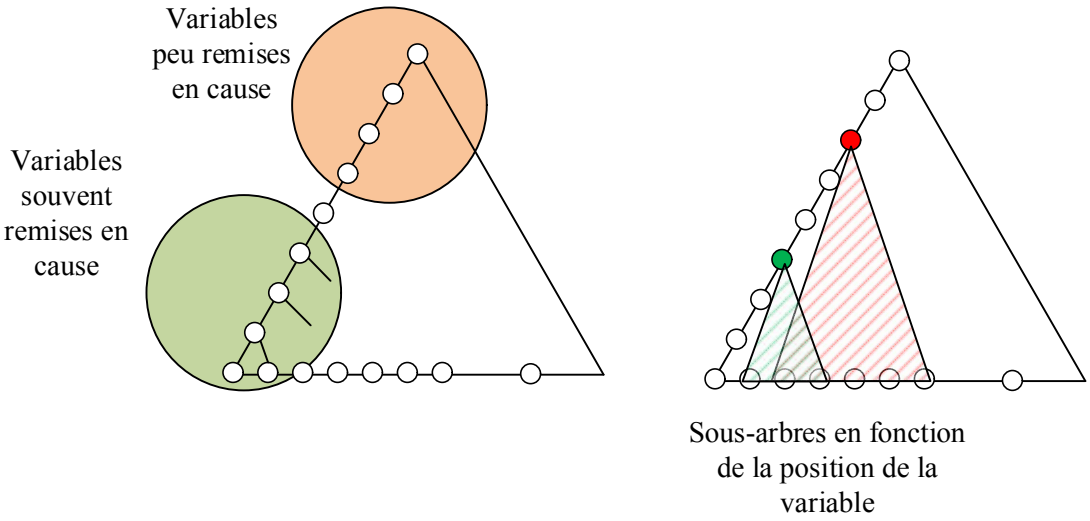


Figure 34 Remise en cause des variables suivant leur position

Certaines variables ont plus d'influences que d'autres sur la taille de l'arbre de recherche et sur le parcours de celui-ci. La Figure 35 présente un graphe des contraintes, où chaque variable est représentée par un nœud, et une contrainte reliant deux variables par une arête. Certains nœuds ont un degré très faible (les nœuds en orange), tandis que d'autres ont au contraire un degré plus fort (les nœuds verts). Lors des premiers branchements dans l'arbre de recherche, si les variables sélectionnées (ou branchées) correspondent aux nœuds en vert, alors elles permettent de propager des informations vers de nombreuses autres variables, et ainsi réduire la taille de l'arbre. Au contraire, la sélection de variables avec un faible degré lors des premiers branchements ne permet généralement pas beaucoup de propagations.

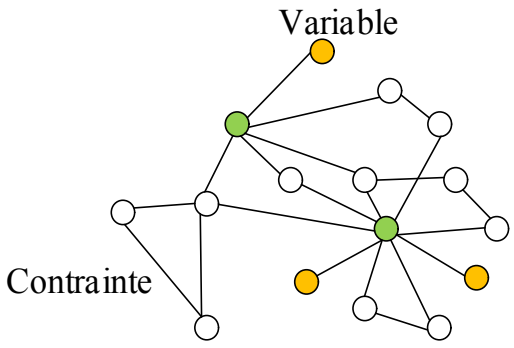


Figure 35 Graphe des contraintes

L'ordre de branchement des variables est donc en ce sens primordial, surtout qu'une variable sélectionnée lors d'un branchement en haut de l'arbre n'est pas remise en cause rapidement. Un mauvais branchement risque d'empêcher l'obtention de bonnes solutions. Généralement, les solveurs promeuvent une stratégie de branchement *DomOverWDeg*, où les variables sélectionnées en premier sont les variables les plus « importantes » d'après le solveur. Pour chaque variable i, le solveur calcule $DomOverWDeg_i = \frac{Dom_i}{W_i Deg_i}$, où Dom_i est

la taille du domaine, Deg_i est le degré du nœud dans le graphe des contraintes, et W_i un poids calculé par le solveur pouvant être vu comme de l'apprentissage. La variable i avec la valeur $DomOverWDeg_i$ la plus petite est sélectionnée en priorité, car elle a un petit domaine (peu de valeurs possibles pour la variable), et beaucoup de contraintes associées. Cette variable est considérée comme « importante » ou « intéressante » pour le solveur.

La stratégie *DomOverWDeg* a notamment été testée pour la CPDM, mais sans résultat convaincant lorsque la troisième étape est itérée plusieurs fois. Cette stratégie déterministe sélectionne toujours les variables dans le même ordre. Si la résolution PPC est limitée dans le temps (comme c'est le cas ici), alors les variables en haut de l'arbre ne sont jamais remises en question et une grande partie de l'arbre n'est pas explorée. Laisser plus de temps à la résolution PPC n'a pas permis à celle-ci de fournir de meilleurs résultats.

La meilleure stratégie trouvée est de donner un ordre de branchement Bra, qui est aléatoire et nouveau à chaque itération. Cette stratégie s'apparente à une approche de type multi-starts, principe classiquement utilisé dans le domaine des métaheuristiques.

4.4.1. Reconstruction des tournées par insertion

L'objectif de la phase de reconstruction des tournées par insertion est de trouver une solution respectant toutes les contraintes de coordination, en modifiant légèrement les tournées obtenues lors de l'étape 2. Ici, le point capital est que les modifications envisagées sont limitées en temps et/ou en distance par rapport à la solution initiale. Cette approche définit clairement un processus proche de ceux utilisés dans les métaheuristiques. Cette limitation sur les modifications autorisées des tournées peut conduire à des situations ou une contrainte de coordination ne peut pas être ajoutée, parce qu'il est interdit d'envisager des solutions trop éloignées de la solution initiale.

Cette phase est itérée *max_iter* fois et fait appel à la fonction Iterative Constraint-Programming based Algorithm (ICPA). L'objectif de l'ICPA est d'insérer itérativement les contraintes de coordination dans les tournées : par expérience, l'insertion simultanée de toutes les contraintes ne permet pas d'obtenir une solution respectant toutes les contraintes en un temps raisonnable, et/ou respectant la distance maximale définie entre la nouvelle solution et la solution initiale. Une itération de l'ICPA consiste à trouver une solution respectant un sous-ensemble $LocalCC \subseteq CC$ de contraintes de coordination (CC est l'ensemble de toutes les contraintes de coordination).

L'ICPA prend en entrée des tournées t, les contraintes de coordination $ChCC$ vérifiées par les tournées t, les contraintes de coordination $UnCC$ non respectées par les tournées t, et un ordre $OrCC$ d'insertion des contraintes de coordination. L'ICPA retourne en sortie un nouvel ensemble de tournées t', un nouvel ensemble de contraintes de coordination $ChCC'$ vérifiées, et un nouvel ensemble de contraintes de coordination non respectée $UnCC'$, de telle sorte que $\forall c \in ChCC$, et $c \in ChCC'$ (une contrainte respectée en entrée doit rester vérifiée en sortie). La fonction ICPA est présentée en détail dans la section 4.4.2.

La procédure globale de reconstruction des tournées par insertion des contraintes de coordination est introduite par l'Algorithme 2. À chaque itération (lignes 13-19), la procédure ICPA est appelée (ligne 14) afin de trouver une solution vérifiant toutes les contraintes de coordination (ou un maximum de contraintes). L'ordre d'insertion des contraintes de coordination dans la procédure ICPA est défini par $OrCC$. Cet ordre est généré de façon

aléatoire, car il n'a pas d'influence importante sur la solution finale (les résultats expérimentaux sur différents ordres sont présentés à la section 5.3.1). Une nouvelle solution issue de l'ICPA est dans le pire des cas la même, et dans le meilleur des cas une meilleure solution (c'est-à-dire qu'il y a plus de contraintes de coordination prises en compte ou une fonction objectif de coût plus faible). Une contrainte de coordination respectée en entrée ne peut pas devenir non vérifiée par l'ICPA.

Algorithme 2 : Reconstuction des tournées par insertion des contraintes de coordination

1. **Sortie**
2. t : ensemble des tournées
3. Cost : coût de la solution
4. **Entrée**
5. t : ensemble des tournées
6. $UnCC$: ensemble des contraintes de coordination non respectées
7. $ChCC$: ensemble des contraintes de coordination respectées
8. iter_max : nombre maximal d'itérations
9. **Variables**
10. $OrCC$: ordre d'insertion des contraintes de coordination
11. **Début**
12. | $OrCC$ = creer_ordre() // création aléatoire d'un ordre des CC
13. | **Pour** $i = 1, \ldots,$ iter_max **Faire**
14. | | (t', $ChCC'$, $UnCC'$, Cost') = **ICPA**(t, $ChCC$, $UnCC$, Cost, $OrCC$) // fonction ICPA
15. | | $t = t'$
16. | | $ChCC = ChCC'$
17. | | $UnCC = UnCC'$
18. | | Cost = Cost'
19. | **FinPour**
20. | **Retourner (t, Cost)**
21. **Fin**

4.4.2. L'Iterative Constraint-Programming based Algorithm

L'Iterative Constraint-Programming based Algorithm (ICPA) prend en entrée une solution définie par un ensemble de tournées t, un ensemble de contraintes de coordination respectées $ChCC$, un ensemble de contraintes de coordination non vérifiées $UnCC$, et un ordre $OrCC$ d'insertion des contraintes de coordination appartenant à $UnCC$.

L'objectif de l'ICPA est de trouver une solution meilleure que celle passée en entrée (donc définie par l'ensemble des tournées t). Une solution S' est dite meilleure qu'une solution S si elle respecte un plus grand nombre de contraintes de coordination $|ChCC(S')| > |ChCC(S)|$ ou si la valeur de la fonction objectif est plus petite $f(S') < f(S) \wedge |ChCC(S')| = |ChCC(S)|$. L'ICPA est, dans ce sens, une approche itérative semblable à une recherche locale.

L'Algorithme 3 est l'algorithme de principe de l'ICPA. Une itération de l'ICPA (lignes 22 à 31) correspond à la recherche d'une solution respectant un sous-ensemble $LocalCC$ de contraintes de coordination. La recherche d'une nouvelle solution s'effectue à partir d'une solution initiale définie par les tournées t. L'ensemble $LocalCC$ est initialisé par l'ensemble $ChCC$ des contraintes respectées (ligne 17). À chaque itération, une contrainte de

coordination c non vérifiée ($c \in UnCC$) est ajoutée à l'ensemble $LocalCC$ (ligne 24), puis le problème est résolu par PPC avec la fonction *solve_CP*. Si une nouvelle solution est obtenue, alors celle-ci respecte toutes les contraintes de coordination de l'ensemble $LocalCC$. La nouvelle solution remplace l'ancienne (ligne 29) et devient le nouveau point de départ lors des futures résolutions. La contrainte c est ajoutée à l'ensemble $ChCC$ (ligne 27) et est supprimée de l'ensemble $UnCC$ (ligne 28). L'ordre Bra de branchement des variables dans le solveur PPC est aléatoire et généré à chaque itération (ligne 12). Si toutes les contraintes de coordination sont respectées ($UnCC = \emptyset$), alors la procédure ICPA consiste uniquement en une recherche locale pour améliorer la solution (lignes 16 à 20).

Algorithme 3 : Iterative Constraint-Programming based Algorithm (ICPA)

1. **Sortie**
2. t : ensemble des tournées
3. Cost : coût de la solution
4. $UnCC$: ensemble des contraintes de coordination non respectées
5. $ChCC$: ensemble des contraintes de coordination respectées
6. **Entrée**
7. t : ensemble des tournées
8. $UnCC$: ensemble des contraintes de coordination non respectées
9. $ChCC$: ensemble des contraintes de coordination respectées
10. $OrCC$: ordre d'insertion des contraintes de coordination
11. **Variables**
12. S : ordre de branchement dans l'arbre de PPC
13. $LocalCC$: contraintes de coordination prises en compte lors d'une itération
14. new_sol : booléen qui vaut vrai si une nouvelle solution est trouvée
15. **Début**
16. | Bra = random_order_of_var_in_CP()
17. | $LocalCC = ChCC$
18. | (t', Cost, new_sol) = solve_CP(t, $LocalCC$, Bra)
19. | **Si** new_sol **Alors**
20. | | $t = t'$
21. | **FinSi**
22. | **Pour** $\forall c \in UnCC$ d'après $OrCC$ **Faire**
23. | | Bra = **random_order_of_var_in_CP**()
24. | | $LocalCC = ChCC \cup \{c\}$
25. | | (t', Cost, new_sol) = **solve_CP**(t, $LocalCC$, Bra)
26. | | **Si** new_sol **Alors**
27. | | | $ChCC = ChCC \cup \{c\}$
28. | | | $UnCC = UnCC \setminus \{c\}$
29. | | | $t = t'$
30. | | **FinSi**
31. | **FinPour**
32. | **Retourner (t, Cost, $ChCC$, $UnCC$)**
33. **Fin**

La procédure ICPA retourne une nouvelle solution t, ainsi qu'un nouvel ensemble $ChCC$ de contraintes de coordination prises en compte et un nouvel ensemble $UnCC$ de contraintes de coordination non respectées. Le modèle PPC utilisé à chaque itération est présenté dans la section suivante.

4.4.3. Le modèle de Programmation Par Contraintes

Le modèle de Programmation Par Contraintes (PPC) utilisé par l'ICPA est celui introduit au chapitre 3 et enrichi des contraintes de coordination (voir section 2.3). Le solveur PPC ne peut pas évaluer, en un temps raisonnable toutes les façons possibles de modifier les tournées, car le nombre de possibilités est très grand. Il est donc nécessaire d'introduire une limite de temps (*SPClimitTime*) afin d'obtenir des temps d'exécution acceptables. Une distance entre les tournées initiales et la solution (seules de légères modifications des trajets sont autorisées) est également envisagée pour obtenir une intensification autour de la solution initiale.

Les tournées (voir chapitre 3) sont notamment définies par :
- a_i, l'affectation de la visite i (à un employé).
- s_i, le successeur de la visite i.

La distance entre les tournées initiales et la solution courante du solveur PPC est définie par rapport aux variables a_i et s_i. Soient :
- A_i, l'affectation initiale de la visite i à un employé.
- S_i, le successeur initial de la visite i.

Deux nouvelles variables binaires sont définies :
- ba_i, un booléen qui vaut 1 si $a_i \neq A_i$ et 0 sinon.
- bs_i, un booléen qui vaut 1 si $s_i \neq s_i$ et 0 sinon.

Les écarts entre les vecteurs A_i et a_i d'une part, et entre S_i et s_i d'autre part, sont définis par une distance de Hamming. δ est l'écart maximal entre le vecteur A_i et le vecteur a_i et γ est l'écart maximal entre les vecteurs S_i et s_i. Les contraintes (PPC 1) à (PPC 6) sont ajoutées au modèle PPC du chapitre 3 pour mesurer et limiter la distance.

$$ba_i \in \{0\,;1\} \qquad \forall i \in V \qquad \text{(PPC 1)}$$

$$bs_i \in \{0\,;1\} \qquad \forall i \in V \qquad \text{(PPC 2)}$$

$$ba_i = (a_i \neq A_i) \qquad \forall i \in V \qquad \text{(PPC 3)}$$

$$bs_i = (s_i \neq S_i) \qquad \forall i \in V \qquad \text{(PPC 4)}$$

$$\sum_{\forall i \in V} ba_i \leq \delta \qquad \text{(PPC 5)}$$

$$\sum_{\forall i \in V} bs_i \leq \gamma \qquad \text{(PPC 6)}$$

La limite de temps *SPClimitTime* est définie de telle sorte que la solution trouvée soit de bonne qualité et que la probabilité pour le solveur de trouver une solution meilleure après cette date soit faible. Avec cette approche limitée en temps, la sélection des variables dans l'arbre de branchement devient un point clef : les variables sélectionnées au début de la recherche ne sont pas reconsidérées dans le temps imparti.

L'efficacité d'une approche de ce type, basée sur un temps d'exécution très court, repose sur une définition correcte des stratégies de sélection de variables dans l'arbre de branchement. En effet, les backtracks effectués par le solveur, ne peuvent avoir lieu que très localement, et les décisions de branchement effectuées en « haut » de l'arbre ne sont pas remises en cause.

À chaque itération de l'ICPA, un nouvel ordre Bra de branchement aléatoire est défini. Lors des itérations paires, les premières variables dans l'arbre de branchement sont les variables a_i, puis les variables s_i. Lors des itérations impaires, ce sont les variables s_i les premières, puis les variables a_i. De plus, l'ordre des variables a_i et s_i est aléatoire. L'ordre de sélection de ces variables permet d'explorer un plus large espace des solutions. Certaines expériences numériques et discussions au sujet de cet ordre sont présentées dans la section 5.3.2.

5. Études numériques

À notre connaissance, il n'existe pas de jeu d'instances dédiées au GWSRP. Les études numériques portent sur deux jeux d'instances : celui de (Bredström and Rönnqvist, 2008) pour le VRPTWSyn (Vehicle Routing and Scheduling Problem with Time Windows and Synchronized visits), qui peut être résolu par la CPDM ; et un nouveau jeu introduit spécifiquement pour le GWSRP. Des études numériques portent également sur les paramètres du modèle PPC lors de la troisième étape de la CPDM.

Les expériences sont conduites sous Windows 7, sur un ordinateur Intel Core i7-4790 3.60 GHz, avec 16.0 Go de RAM, ce qui équivaut à 2671 MFlops selon Dongarra (http://www.roylongbottom.org.uk/linpackresults.htm). Le solveur PPC utilisé est Choco 4.10.0. La version de CPLEX est la 12.7.0. La génération de colonnes est implémentée en C++.

Le Tableau 5 introduit les paramètres utilisés par la CPDM suivant les instances. Les valeurs de ces paramètres ont été ajustées empiriquement. Les instances GGLT définissent un nouveau jeu d'instances spécialement introduit pour le GWSRP (et présenté à la section 5.2). L'instance 13 est très nettement plus grande que les autres et les paramètres ont dû être ajustés.

Brièvement, *SCGlimitTime* est la limite de temps de la génération de colonnes ; *Labels/nœud* est le nombre de labels maximal par nœud lors de la résolution dynamique ; et *Nb colonnes/Resolution* est le nombre de colonnes remontées au problème maître lors de la résolution d'un sous-problème (voir chapitre 3) ; *NbThreadsCPLEX* est le nombre maximal de threads que peut utiliser CPLEX pour la résolution du problème maître. Pour la seconde étape de la CPDM, *NbThreadsCPLEX* est le nombre limite de threads que peut utiliser CPLEX ; *SClimitTime* est la durée (en secondes) maximale lors de la résolution du Set Partitioning Problem. Pour la troisième étape de la CPDM, *iter_max* est le nombre d'itérations de l'insertion des contraintes ; *TimeLimit/Iter* est le temps maximal laissé à Choco lors de la résolution PPC à chaque itération de l'ICPA ; δ et γ définissent les distances maximales entre la solution courante et la solution initiale (voir section 4.4.3). Pour la résolution directe par CPLEX, *NbThreadsCPLEX* est le nombre threads utilisé, et *TimeLimit* est la durée de calcul maximal autorisée.

Tableau 5 Paramétrage de la CPDM

	Instances		(Bredström and Rönnqvist, 2008)	GGLT 1 à 12	GGLT 13
CPDM	Construction des tournées initiales (Génération de colonnes)	*SCGlimitTime*	3600 s	3600 s	
		Labels/nœud	250	250	50
		Nb colonnes/ Resolution	20	20	20
		NbThreadsCPLEX	1	1	
	Sélection des tournées	*NbThreadsCPLEX*	1	1	
		SClimitTime	150 s	150 s	1260 s
	Insertion des contraintes de coordination	*iter_max*	15	15	
		TimeLimit/Iter	10 s	10 s	40 s
		δ	$\|V\|/2$	/	
		γ	$\|V\|/2$	/	
CPLEX		*NbThreadsCPLEX*	/	1	
		TimeLimit	/	10800 s	

Les instances de (Bredström and Rönnqvist, 2008) ainsi que les nouvelles instances GGLT, et les solutions obtenues sont téléchargeables à l'adresse suivante : http:\\www.isima.fr\~lacomme\GWSRP\ et http://fc.isima.fr/~gondran/researches.html.

5.1. Tests sur les instances de (Bredström and Rönnqvist, 2008)

Les instances de (Bredström and Rönnqvist, 2008) pour le VRPTWSyn sont au centre de plusieurs articles récents, et des méthodes dédiées sont expérimentées sur celles-ci. Le VRPTWSyn combine un problème d'affectation, un problème d'ordonnancement et un problème de tournées de véhicules, ce qui fait du VRPTWSyn un des problèmes de la littérature les plus proches du GWSRP.

L'étude de cette section est une comparaison entre cinq approches dédiées au VRPTWSyn et la CPDM. Cette dernière n'est pas une approche spécifique à ce problème, mais elle permet de trouver des solutions de bonne qualité avec quelques mérites par rapport aux méthodes dédicacées.

Le VRPTWSyn (VRP with Time Windows and Synchronized visits) peut être traité par un modèle ou une approche résolvant le GWSRP. Les trois différences majeures entre ces deux problèmes sont : la fonction objectif du VRPTWSyn, qui ne prend en compte que la distance ; les fenêtres de temps des employés qui n'ont pas le droit d'être violées dans le VRPTWSyn ; et enfin le VRPTWSyn n'inclut qu'un seul type de contraintes de coordination (les contraintes de synchronisation).

Plus formellement la fonction objectif du VRPTWSyn est :

$$f^{VRPTWSyn} = \sum_{i \in V \cup I_w \cup L_w} \sum_{j \in V \cup I_w \cup L_w} (T_{i,j}) x_{i,j}^{w}$$

Pour rappel, la fonction objectif du GWSRP est :

$$f = \lambda_1 \sum_{i \in V \cup I_w \cup L_w} \sum_{j \in V \cup I_w \cup L_w} \left(T_{i,j} + p_i^w \right) x_{i,j}^w + \lambda_2 \sum_{i \in \mathrm{V}} \left(3 - \sum_{\mathrm{j} \in V \cup L_w} \rho_i^w \, x_{i,j}^w \right)$$

$$+ \lambda_3 \sum_{i \in \mathrm{V}} \left(\psi_i^w + \theta_i^w \right) + \lambda_4 \sum_{i \in \mathrm{V}} y_i$$

PC_i^w est le coût de réalisation de la visite i par l'employé w. Pour le VRPTWSyn, $PC_i^w = 0, \forall i \in V, \forall w \in W$. ρ_i^w est la qualité de service concernant la visite (et par extension le client). Lorsque l'employé w réalise la visite i, $\rho_i^w = 3$ $(\forall i \in V, \forall w \in W)$ pour le VRPTWSyn. PA_i^w est une variable binaire valant 1 si w apprécie la région où est située i, et valant 0 sinon. Pour le VRPTWSyn, $PA_i^w = 1 \ \forall i \in V, \forall w \in W$. Le GWSRP permet de modéliser le cas où certains employés ne peuvent pas être affectés à une visite, $CoTr_i^w = 1$ si la visite est autorisée, et 0 sinon. Cette considération n'est pas prise en compte dans le VRPTWSyn et donc $CoTr_i^w = 1, \forall i \in V, \forall w \in W$.

(Bredström and Rönnqvist, 2008) proposent 10 instances numérotées de 1 à 10. La fenêtre de temps des visites est un paramètre de ces instances, et trois cas sont fréquemment étudiés : Short, Medium et Large. Les instances sont nommées par le numéro auquel est ajoutée une lettre une lettre (S, M ou L) en fonction de la fenêtre de temps des visites. Au total, le jeu de données se compose de 30 instances.

La fonction objectif du VRPTWSyn est $f^{VRPTWSyn} = \sum_{i \in V \cup I_w \cup L_w} \sum_{j \in V \cup I_w \cup L_w} \left(T_{i,j} \right) x_{i,j}^w$, comme il l'a été signalé précédemment. Néanmoins, les valeurs des solutions dans les publications correspondent à f^{bred}. Cette valeur est calculée par la formule suivante : $f^{bred} = \frac{9 f^{VRPTWSyn}}{H}$, d'après (Bredström and Rönnqvist, 2008), où H est un horizon de planification propre à chaque instance. Par souci de clarté, les résultats présents dans ce chapitre correspondent à $f^{VRPTWSyn}$.

Le Tableau 6 présente les caractéristiques des instances de VRPTWSyn de (Bredström and Rönnqvist, 2008). Pour chacune des 10 instances, sont donnés le nombre d'employés, le nombre de clients, le nombre de visites, le nombre de contraintes de synchronisation et l'horizon H. Le nombre de clients est plus faible que le nombre de visites, car certains clients requièrent deux visites et celles-ci doivent être synchronisées.

Tableau 6 Caractéristiques des instances de VRPTWSyn

N° instance	Nb employés	Nb clients	Nb visites	Nb contraintes de synchronisation	Horizon H
01	4	18	20	2	695
02	4	18	20	2	730
03	4	18	20	2	702
04	4	18	20	2	554
05	4	18	20	2	717
06	10	45	50	5	672
07	10	45	50	5	651
08	10	45	50	5	704
09	16	72	80	8	718
10	16	72	80	8	643

(Bredström and Rönnqvist, 2007) proposent deux Branch-and-Price (BP1 et BP2) ; (Bredström and Rönnqvist, 2008) présentent les résultats obtenus par par une heuristique H ; la résolution par PLNE et la méthode SA-ILS sont proposées par (Afifi et al., 2016) ; et la méthode ALNS par (Liu et al., 2018).

Le Tableau 7 introduit les caractéristiques et les conditions expérimentales des articles cités précédemment. Pour chaque publication, sont donnés le nombre d'exécutions de chaque méthode, les caractéristiques de l'ordinateur, l'OS, le langage et les MFlops. Connaître les MFlops des différents ordinateurs permet de calculer un ratio à appliquer aux temps de calcul afin d'obtenir une comparaison juste et équitable des temps de calcul.

L'information « Intel Xeon 2.67GHz », fournie dans les articles de (Bredström and Rönnqvist, 2007, 2008) et (Afifi et al., 2016), n'est pas suffisante pour évaluer les MFlops des ordinateurs utilisés par les auteurs. Il faut aussi noter que (Liu et al., 2018) exécutent plusieurs fois leur approche, puisque celle-ci comporte une part d'aléatoire. Cela dit, le nombre d'exécutions n'est pas indiqué. Les indications partielles ne permettent donc pas d'avoir le ratio à appliquer aux temps de calcul.

Tableau 7 Caractéristiques des méthodes publiées

	(Bredström and Rönnqvist, 2007)		(Bredström and Rönnqvist, 2008)	(Afifi et al., 2016)	(Liu et al., 2018)	
Nom de la méthode	**BP1**	**BP2**	**H**	**SA-ILS**	**ALNS**	**CPDM**
Nb d'exécutions	1		1	10	Non communiqué	1
Ordinateur	Intel Xeon 2.67 GHz		Intel Xeon 2.67 GHz	Intel Xeon 2.67 GHz	Intel E5-2670	Intel Core i7-4790 3.60 GHz
OS				LINUX	LINUX	Windows
Langage	C++		C++	C++	C++	C++/Java
MFlops	/		/	/	2570	2671
Ratio	/		/	/	0.96	1

Le Tableau 8 introduit les résultats des différentes méthodes pour chaque instance. La seconde colonne du tableau est la meilleure solution S connue, et lorsque celle-ci est en gras avec un astérisque, cela signifie qu'elle a été prouvée optimale. La valeur de S correspond à la valeur de la solution $f^{VRPTWSyn}$.

Tableau 8 CPDM sur les instances de VRPTWSyn

	Best	(Bredström and Rönnqvist, 2007)						(Bredström and Rönnqvist, 2008)			(Afifi et al., 2016)						(Liu et al., 2018)						
	S	BP1 1 run			BP2 1 run			H 20 runs			PLNE			SA-ILS 10 runs			ALNS >1 run (not given)			**CPDM 1 run**			
		S	*T*	*TT*	*S*	*T*	*TT*	*S*	*T*	*TT*	*S*	*T*	*TT*	*S*	*T*	*TT*	*S*	$\bar{T}$	*TT*	*S*	*T*	*TT*	Gap
01S	**274***	274	/	1.96	274	/	1.12	274	/	120.03	274	/	3.43	274	0.02	/	274	0.06	/	**274**	0.6	0.6	**0.0**
02S	**346***	346	/	3.28	346	/	0.56	346	/	120.07	346	/	0.22	346	0.02	/	346	0.08	/	**346**	0.6	0.6	**0.0**
03S	**283***	283	/	14.17	283	/	3.84	283	/	120.26	283	/	1.79	283	0.02	/	283	0.09	/	**283**	0.7	0.7	**0.0**
04S	**378***	378	/	14.02	378	/	1.54	378	/	120.16	378	/	30.9	378	0.02	/	378	0.08	/	**378**	2.6	2.6	**0.0**
05S	**313***	313	/	2.84	313	/	2.90	313	/	120.13	313	/	6.99	313	0.03	/	313	0.09	/	**313**	2.2	4.0	**0.0**
06S	**608***	608	/	3600	608	/	197	608	/	600.94	608	/	3600	608	13.97	/	608	1.67	/	**608**	7.4	7.4	**0.0**
07S	**607***	607	/	14.72	607	/	169	/	/	603.97	/	/	3600	607	18.34	/	607	1.71	/	608	235.2	251.0	0.2
08S	**746***	746	/	931	746	/	850	/	/	657.03	/	/	3600	746	25.13	/	746	2.11	/	750	102.2	264.1	0.5
09S	952	/	/	3600	974	/	3600	/	/	626.26	/	/	3600	952	150.52	/	953	22.50	/	1086	315.0	320.0	14.1
10S	610	/	/	3600	652	/	3600	/	/	604.46	/	/	3600	614	16.1	/	610	20.10	/	696	339.1	344.1	14.1
01M	**274***	274	/	221.93	274	/	3.69	274	/	120.25	274	/	14.48	274	0.02	/	274	0.06	/	**274**	4.6	5.1	**0.0**
02M	**290***	290	/	8.12	290	/	3.20	290	/	120.11	290	/	25.97	290	0.03	/	290	0.09	/	**290**	0.9	0.9	**0.0**
03M	**260***	260	/	17.57	260	/	4.31	260	/	120.19	260	/	21.24	260	0.03	/	260	0.09	/	**260**	0.7	0.7	**0.0**
04M	**349***	349	/	27.53	349	/	2.55	349	/	120.15	349	/	1380	349	0.05	/	349	0.08	/	**349**	2.6	2.6	**0.0**
05M	**281***	281	/	57.04	281	/	9.10	281	/	120.09	281	/	6.2	281	0.03	/	281	0.08	/	**281**	2.6	2.6	**0.0**
06M	575	576	/	3600	575	/	3600	868	/	609.58	608	/	3600	575	26.68	/	575	1.63	/	581	36.0	181.7	1.0
07M	541	555	/	3600	547	/	3600	/	/	648.02	916	/	3600	541	18.34	/	541	1.94	/	575	200.9	270.5	6.3
08M	**668***	668	/	3600	668	/	3490	/	/	632.61	/	/	3600	668	15.01	/	668	1.85	/	689	265.0	264.9	3.1
09M	871	937	/	3600	881	/	3600	/	/	612.19	/	/	3600	871	292.17	/	871	23.03	/	952	328.7	333.7	9.3
10M	544	610	/	3600	579	/	3600	/	/	705.2	/	/	3600	544	52.75	/	545	16.20	/	607	311.6	316.6	11.6
01L	**262***	262	/	107.41	262	/	11.91	262	/	120.48	262	/	76.71	262	0.03	/	262	0.06	/	**262**	1.2	1.2	**0.0**
02L	**277***	277	/	2.72	277	/	7.41	277	/	120.95	277	/	183.11	277	0.03	/	277	0.09	/	**277**	1.6	1.6	**0.0**
03L	**257***	257	/	42.78	257	/	1.44	257	/	120.60	257	/	96.47	257	0.02	/	257	0.09	/	**257**	1.4	1.4	**0.0**
04L	**316***	316	/	9.74	316	/	7.69	326	/	120.04	316	/	3600	316	0.09	/	316	0.09	/	**316**	140.4	140.4	**0.0**
05L	**266***	266	/	9.11	266	/	5.15	266	/	120.85	266	/	225.23	266	0.03	/	266	0.09	/	**266**	1.3	1.3	**0.0**
06L	**533***	533	/	3279	533	/	3600	/	/	624.06	/	/	3600	533	15.86	/	533	1.47	/	534	65.0	214.9	0.2
07L	498	498	/	3600	498	/	3600	/	/	645.33	916	/	3600	498	15.92	/	498	1.66	/	540	253.4	258.4	8.4
08L	626	674	/	3600	634	/	3600	/	/	618.63	/	/	3600	626	24.51	/	627	1.92	/	640	96.6	246.5	2.2
09L	832	886	/	3600	869	/	3600	/	/	607.36	/	/	3600	837	207.17	/	832	21.54	/	914	287.4	292.4	9.9
10L	526	/	/	3600	/	/	3600	/	/	631.39	/	/	3600	554	51.89	/	526	17.00	/	579	322.6	340.6	10.1

* Solution prouvée optimale

La colonne T indique le temps (en secondes) pour trouver la meilleure solution. Les colonnes TT indiquent le temps total de résolution (en secondes). Les temps sont normalisés quand cela est possible grâce aux informations du Tableau 7. Les trois dernières colonnes concernent les résultats obtenus par la CPDM. Gap est le gap en pourcentage entre la solution obtenue par la CPDM et la meilleure solution connue.

Il est important de noter que les méthodes BP1, BP2 et CPDM ne sont exécutées qu'une seule fois, tandis que la méthode H est exécutées 20 fois, le SA-ILS 10 fois, et l'ALNS plusieurs fois sans que le nombre exact ne soit mentionné. Pour les méthodes BP1, BP2 et H, seul le temps total de calcul est fourni, tandis que pour la méthode SA-ILS, le temps donné est le temps pour trouver la meilleure solution. Pour l'ALNS, le temps indiqué est le temps moyen d'une exécution normalisée. Il est difficile, dans ces conditions, de proposer une étude comparative équitable.

Le Tableau 8 montre que l'approche CPDM proposée fournit des solutions de bonne qualité pour la majorité des instances, excepté pour les instances 9 et 10. Ces instances sont celles qui comportent le plus de clients et d'employés. Pour les instances 1 à 5 (quel que soit l'ensemble S, M ou L), la CPDM est capable de trouver la solution optimale en quelques secondes. Par exemple, pour l'instance 01S, la solution optimale $S = 274$ est obtenue en 0.6 seconde.

Le Tableau 9 est un résumé des différentes méthodes publiées pour les instances de (Bredström and Rönnqvist, 2008). Il met en lumière le nombre d'instances résolues par chaque méthode (seconde colonne). Par exemple, la méthode BP1 résout 27 instances, la BP2 résout 29 instances et H résout 17 instances, tandis que les méthodes SA-ILS, ALNS et CPDM résolvent les 30 instances.

Les méthodes SA-ILS et ALNS ont un temps de résolution pour trouver la meilleure solution plus faible que la CPDM, mais ne fournissent pas le temps total d'exécution. L'AVG Gap est le gap moyen avec les meilleures solutions connues. La CPDM a un gap de 3.0%, proche de celui de H mais avec un temps de calcul 2.75 fois plus faible (136 secondes en moyenne pour la CPDM, contre 374 secondes en moyenne pour la méthode H). L'ALNS a un gap moyen de 0.0% en un temps de calcul très faible, d'environ 4 secondes en moyenne par réplication. Cependant, il s'agit d'une méthode dédiée à ces instances et au problème du VRPTWSyn, contrairement à la CPDM qui est une méthode dédiée au GWSRP. En conclusion, la méthode CPDM obtient des résultats de bonne qualité. Toutefois, la meilleure méthode est l'ALNS de (Liu et al., 2018).

Tableau 9 Résumé de la CPDM sur les instances de VRPTWSyn

	Nombre d'instances résolues	**Avg. *T***	**Avg. *TT***	**Avg Gap (%)**
(Bredström and Rönnqvist, 2007) – BP1	27		1598	1.4
(Bredström and Rönnqvist, 2007) – BP2	29		1479	0.8
(Bredström and Rönnqvist, 2008) – H	17		374	3.2
(Afifi et al., 2016) – SA-ILS	30	31		0.2
(Liu et al., 2018) – ALNS	30	4*		0.0
CPDM	30	111	136	3.0

5.2. Les instances GGLT

À notre connaissance, aucune instance spécifiquement dédiée au GWSRP n'existe. C'est la raison pour laquelle, un ensemble de 13 instances a été généré. Les instances sont identiques à celles introduites pour le WSRP (chapitre 3), et elles sont enrichies de contraintes de coordination. Le Tableau 10 présente ces instances, en rappelant le nombre de visites et le nombre d'employés de chaque instance. Le nombre de contraintes de chaque catégorie est également indiqué. Par exemple, l'instance 1 possède 3 contraintes de type C1 (contraintes de disjonction), 2 contraintes de types C2 (contraintes de synchronisation), 1 contrainte de type C3 (contraintes avec deux visites ayant une durée d'exécution en commun), 2 contraintes de type C4 (contraintes de visite ayant une durée d'exécution pendant un intervalle particulier), 3 contraintes de type C5 (visites conjonctives), 3 contraintes de type C6 (time-lag minimal conjonctif), 1 contrainte de type C7 (time-lag minimal disjonctif), 2 contraintes de type C8 (time-lag maximal conjonctif), et enfin 1 contrainte de type C9 (time-lag maximal disjonctif). Le nombre total de contraintes de coordination est donné dans la dernière colonne (TTC).

Il est important de noter que le nombre de contraintes C1 (contraintes disjonctives) n'est pas suffisant pour appréhender la difficulté de l'instance, car plusieurs visites (et non pas uniquement deux visites) peuvent être concernées par une même contrainte C1. Par exemple, si les visites v_1, v_2, v_3 et v_4 appartiennent à une même contrainte disjonctive, alors celle-ci est comptabilisée comme une contrainte dans le Tableau 10 (colonne C1). Or le nombre réel de contraintes disjonctives deux à deux est de 6. La colonne « C1 : nb disjonctions » du Tableau 10 indique le nombre de contraintes disjonctives deux à deux.

Les instances ont été créées de manière à ce qu'une solution, obtenue en relâchant les contraintes de coordination et en minimisant la fonction objectif, fournisse une borne inférieure de mauvaise qualité et soit fortement différente de la solution optimale prenant en considération les contraintes de coordination. Les instances sont disponibles sur la page web : http:\\www.isima.fr\~lacomme\GWSRP\.

Tableau 10 Caractéristiques des instances GGLT pour le GWSRP

Instance	**Nombre de visites**	**Nombre d'employés**	**Contraintes de coordination**										
			C1	C1 : nb disjonctions	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9	**TC C**
1	32	5	3	19	3	1	2	3	3	1	2	1	**19**
2	31	6	4	8	1	3	0	3	1	2	0	1	**15**
3	38	7	6	28	1	0	0	1	1	0	0	0	**9**
4	28	8	3	45	0	0	0	0	0	0	0	0	**3**
5	13	2	1	6	1	0	0	1	1	1	1	0	**6**
6	28	4	2	21	1	0	0	0	2	2	0	3	**10**
7	9	36	3	32	2	0	0	0	0	0	0	0	**5**
8	71	34	7	123	3	5	0	3	7	8	1	5	**39**
9	30	10	3	67	2	1	0	2	1	2	1	2	**14**
10	62	16	7	163	2	0	0	1	1	2	0	2	**15**
11	57	17	4	244	0	0	0	0	0	0	0	0	**4**
12	61	21	2	20	2	1	0	1	2	2	1	2	**13**
13	177	55	6	106	3	2	0	1	2	2	3	4	**23**

Le Tableau 11 présente les résultats obtenus pour les 13 instances lors de leur résolution par PLNE et par la CPDM. S est la valeur de la meilleure solution trouvée ; *Gap CPLEX* est le gap en pourcentage obtenu par CPLEX entre la meilleure solution et la borne inférieure ; TT est le temps total d'exécution de CPLEX ou de la CPDM en secondes ; T^* est le temps nécessaire à la CPDM pour fournir la meilleure solution ; et $Gap(S)$ est le gap en pourcentage entre la meilleure solution CPLEX et la meilleure solution CPDM. La dernière ligne du tableau est la moyenne des valeurs. Le temps CPLEX est limité à 10800 secondes, tandis que la première phase de la CPDM est limitée à 3600 secondes, la seconde à 150 secondes, et la troisième à 15 fois 10 secondes (voir Tableau 5). Les meilleures solutions connues sont en gras, et les solutions optimales sont suivies d'un astérisque.

L'approche CPDM proposée est en mesure de trouver la solution pour les instances 5 et 6 ($Gap(S) = 0\%$), et une solution très proche de celles optimales pour les instances 1, 2, 3, 4, 7 et 9 ($Gap(S) <= 1\%$). Pour l'instance 2, le $Gap(S) = 0.7\%$. CPLEX requiert 489.4 secondes, tandis que la CPDM obtient sa meilleure solution en 31.6 secondes (soit 15 fois plus rapidement) et nécessite 109.6 secondes au total (soit 4 fois moins de temps que la résolution directe par PLNE). Pour l'instance 4, CPLEX requiert 6372.8 secondes, tandis que la CPDM trouve une solution avec un gap $Gap(S) = 0.3\%$ en 28.4 secondes. Des remarques similaires sont valables pour toutes les instances, excepté pour l'instance 1, où CPLEX a un temps d'exécution inférieur à la CPDM. Des recherches ont été menées pour savoir quelles caractéristiques possède cette instance, de manière à ce que CPLEX soit meilleur que la CPDM. Mais aucune caractéristique particulière n'est clairement apparue. Seule l'instance 11 résiste à la CPDM, et le gap est relativement important ($Gap(S) = 14.7\%$). Aucune raison apparente n'a été découverte en ce qui concerne la difficulté qu'a la CPDM pour résoudre cette instance.

Pour les instances 8 et 10, le gap $Gap(S)$ est inférieur à 6% (respectivement 2.4% et 5.4%). Pour les instances 12 et 13, le gap est négatif signifiant que la CPDM a trouvé une solution meilleure que CPLEX. Par exemple pour l'instance 13, la solution CPLEX a un coût $S = 7576.3$, tandis que la solution de la CPDM a un coût $S = 6875.1$, soit un gap $Gap(S) = -9.3\%$. De plus le temps de résolution de ces instances par la CPDM est plus court que par CPLEX pour l'instance 12, la CPDM met 276.3 secondes, tandis que CPLEX met 10804.1 secondes.

En moyenne (dernière ligne du Tableau 11) pour les 13 instances, la CPDM trouve une solution de coût 1.1% plus élevé que CPLEX, mais nécessite environ 10 fois moins de temps de calcul (422.4 secondes contre 4096.2 secondes). Plusieurs expériences ont été faites pour réduire le gap entre la CPDM et CPLEX, mais aucune amélioration significative n'a été obtenue.

Tableau 11 Solutions des instances GGLT

Instance	**CPLEX**			**CPDM**			
	S	*Gap CPLEX %*	TT (s)	S	T^* (s)	TT (s)	Gap(S)%
1	**703.3***	0.0	174.9	705.0	220.5	238.5	0.2
2	**207.2***	0.0	489.4	208.6	31.6	109.6	0.7
3	**886.5***	0.0	1636.8	889.7	162.4	162.4	0.4
4	**516.8***	0.0	6372.8	518.5	28.4	106.4	0.3
5	**106.6***	0.0	55.2	**106.6***	1.5	1.5	0.0
6	**1033.5***	0.0	800.1	**1033.5***	166.5	166.5	0.0
7	**558.1***	0.0	8061.4	558.3	155.9	203.9	<0.05
8	**645.2**	0.3	10811.3	660.4	704.1	746.1	2.4
9	**135.7***	0.0	842.5	136.2	49.1	109.1	0.4
10	**490.4***	0.0	1110.3	517.0	813.5	867.5	5.4
11	**410.9***	0.0	978.8	471.2	455.5	521.5	14.7
12	414.5	2.6	10804.1	**412.2**	210.3	276.3	-0.6
13	7576.3	37.5	11113.0	**6875.1**	1958.4	1982.4	-9.3
AVG			**4096.2**		**381.4**	**422.4**	**1.1**

Le Tableau 12 présente le détail des solutions obtenues pour chaque critère $C1$, $C2$, $C3$ et $C4$ de la fonction objectif du GWSRP, par résolution PLNE et par la CPDM. Les coefficients associés à chaque critère sont $\lambda_1 = 0.1$, $\lambda_2 = 10.0$, $\lambda_3 = 100.0$ et $\lambda_4 = 1\,000.0$.

Le Tableau 12 met en évidence que la CPDM est capable de trouver les mêmes valeurs que la solution de CPLEX pour les critères $C3$ et $C4$ (excepté pour l'instance 13). Il s'agit des deux critères qui ont le poids le plus fort, et qui correspondent à la qualité de service des employés et au nombre de visites non effectuées. Pour l'instance 13, la CPDM trouve une solution avec une meilleure valeur que CPLEX concernant le critère $C3$ (24 contre 29).

Pour le critère $C2$, la solution trouvée par la CPDM a la même valeur que la solution fournie par CPLEX pour les instances 1, 4, 5, 6, 7 et 8. Mais pour ces dernières, le critère 1 est supérieur en ce qui concerne les solutions de la CPDM à celui des solutions de CPLEX, par exemple pour l'instance 1, le critère $C2$ vaut 522 pour la solution par la CPDM et 462 pour la solution de CPLEX.

L'instance 2 est intéressante car la solution de la CPDM a une valeur de critère C1 (508) inférieure à la solution de CPLEX (524). Mais le critère $C2$ de la solution CPDM est plus élevé que celui de la solution CPLEX (15.78 pour la CPDM contre 15.48 pour CPLEX). Une situation inverse se produit pour l'instance 9 où la solution CPDM a un critère $C2$ plus faible que la solution CPLEX (respectivement 10.19 et 10.29). Cependant, le critère $C1$ est plus élevé (respectivement 343 et 328).

Pour l'instance 12, la CPDM est capable de trouver une solution avec des critères tous égaux ou meilleurs que ceux de la solution de CPLEX.

Tableau 12 Détail des solutions GGLT

	CPLEX					**CPDM**				
Instance	C1 (λ_1)	C2 (λ_2)	C3 (λ_3)	C4 (λ_4)	Total	C1 (λ_1)	C2 (λ_2)	C3 (λ_3)	C4 (λ_4)	Total
1	462	55.71	1	0	703.3	479	55.71	1	0	705.0
2	524	15.48	0	0	207.2	508	15.78	0	0	208.6
3	715	41.5	4	0	886.5	737	41.6	4	0	889.7
4	263	29.05	2	0	516.8	280	29.05	2	0	518.5
5	163	9.03	0	0	106.6	163	9.03	0	0	106.6
6	577	27.58	7	0	1033.5	577	27.58	7	0	1033.5
7	860	27.21	2	0	558.1	862	27.21	2	0	558.3
8	1278	31.74	2	0	645.2	1284	33.2	2	0	660.4
9	328	10.29	0	0	135.7	343	10.19	0	0	136.2
10	1097	28.07	1	0	490.4	1088	30.82	1	0	517.0
11	876	32.33	0	0	410.9	906	38.06	0	0	471.2
12	961	31.93	0	0	414.5	935	31.87	0	0	412.2
13	4227	425.36	29	0	7576.3	4855	428.96	21	0	6875.1

Le Tableau 13 présente les résultats fournis par CPLEX avec un temps d'exécution identique à celui utilisé par la méthode CPDM. La durée d'exécution de CPLEX est limitée par le temps total utilisé par la CPDM. Cela permet d'évaluer la vitesse de convergence des deux méthodes avec un temps maximal alloué identique.

GAP best solution est le gap (en pourcentage) entre la solution CPLEX et la meilleure solution connue. Gap(S)% est le gap (en pourcentage) entre la solution CPLEX et la solution CPDM. CPLEX ne réussit à résoudre entièrement qu'une unique instance (l'instance numéro 1). Toutefois, il réussit à trouver, pour l'instance 10, la meilleure solution, mais il n'aboutit pas à prouver son optimalité dans le temps imparti.

Un Gap(S) négatif signifie que CPLEX trouve une solution meilleure que la CPDM. C'est notamment le cas pour les instances 1, 4 et 9 où le gap est très faible ($Gap(S) > -0.5\%$). Pour les instances 10 et 11 ce gap est plus important (respectivement $-5{,}4\%$ et -8.8%). Un gap nul, ce qui est le cas de l'instance 7, signifie que la solution CPLEX et la solution CPDM sont identiques (ou presque, car dans le cas présent, la différence n'est pas significative).

Un gap Gap(S) positif signifie que la CPDM fournit de meilleures solutions que CPLEX, ce qui est le cas pour les instances 2, 3, 5, 6, 8, 12 et 13. Pour les instances 8, 12 et 13, ce gap est supérieur à 45%. Pour l'instance 8, la solution CPLEX, en 747.2 secondes, est 1203.6, tandis que la solution CPDM est 660.4. Pour l'instance 12, la solution CPLEX est 21008.5 en 277.6 secondes, et 412.2 pour la CPDM. Cette étude met en évidence l'intérêt de la CPDM pour les instances de grande taille, puisqu'elle a une vitesse de convergence supérieure à celle obtenue par une résolution linéaire.

Tableau 13 Résolution par CPLEX avec le même temps de calcul que la CPDM

Instance	**Meilleure solution connue**	**CPLEX**				**CPDM**			
	S^*	S	*Gap CPLEX %*	TT (s)	*GAP best solution %*	S	T^* (s)	TT (s)	Gap(S)%
1	**703.3***	703.3	0.0	153.1	**0.0**	705.0	220.5	238.5	-0.2
2	**207.2***	214.1	11.0	110.1	3.3	208.6	31.6	109.6	**2.6**
3	**886.5***	898.7	2.5	162.1	1.4	889.7	162.4	162.4	**1.0**
4	**516.8***	517.3	0.6	106.2	0.1	518.5	28.4	106.4	-0.2
5	**106.6***	108.8	4.8	2.0	2.1	106.6	1.5	1.5	**2.0**
6	**1033.5***	1043.5	10.6	167.1	1.0	1033.5	166.5	166.5	**1.0**
7	**558.1***	558.4	30.9	204.1	0.1	558.3	155.9	203.9	**0.0**
8	**645.2**	1203.6	47.1	747.2	86.5	660.4	704.1	746.1	**45.1**
9	**135.7***	135.8	5.6	109.1	0.1	136.2	49.1	109.1	-0.3
10	**490.4***	490.4	0.1	867.6	**0.0**	517.0	813.5	867.5	-5.4
11	**410.9***	433.2	6.1	522.6	5.4	471.2	455.5	521.5	-8.8
12	**412.2**	21008.5	98.1	277.6	4996.7	412.2	210.3	276.3	**98.0**
13	**6875.1**	48596.4	90.3	1986.5	606.8	6875.1	1958.4	1982.4	**85.9**
AVG									**17.0**

Le Tableau 14 introduit une comparaison entre les résolutions PLNE et PPC. Ainsi qu'il a déjà été souligné dans le chapitre 3 et dans de nombreuses publications (Bockmayr and Hooker, 2005; Pour et al., 2018; Bourreau et al., 2020), la PPC permet généralement de trouver rapidement une première solution de bonne qualité, mais a ensuite plus de difficultés à obtenir et prouver la solution optimale.

La PPC est limitée à 3600 secondes, et ne trouve en ce temps imparti qu'une seule solution optimale : en 446.2 secondes pour l'instance 5 alors que CPLEX requiert 55.2 secondes. Contrairement à la CPDM, la PPC n'utilise ici ni stratégie itérative, ni solution de départ. Les colonnes S sont les valeurs des meilleures solutions ; TT est le temps total en secondes ; T^* est le temps requis par la PPC pour trouver la meilleure solution ; S^1 est la valeur de la première solution trouvée par PPC ; et T^1 est le temps mis par la PPC pour trouver la première solution.

Pour les instances 8, 10 et 13, la PPC ne réussit pas à trouver une solution, même de mauvaise qualité. La stratégie de branchement serait sûrement à améliorer pour favoriser les branchements sur des visites contenant beaucoup de contraintes de coordination. Toutefois, il est important de noter que pour toutes les instances (exceptées les trois citées précédemment), le modèle PPC est capable de trouver une première solution en moins de 3.4 secondes, voire moins d'une demi-seconde pour les instances 1, 2, 3, 4, 5, 7 et 9. Ces résultats nous confortent dans l'idée d'utiliser la PPC dans la CPDM pour insérer les contraintes de coordination.

Tableau 14 Comparaison CPLEX et PPC

Instance	**CPLEX**			**PPC**					
	S	*Gap CPLEX %*	TT (s)	S	T^* (s)	TT (s)	Gap(S)%	S^1	T^1
1	**703.3**	0.0	174.9	710.1	3153.3	3600.0	1.0	919.5	0.5
2	**207.2**	0.0	489.4	371.3	1650.5	3600.0	79.2	946.8	0.1
3	**886.5**	0.0	1636.8	1197.5	2458.9	3600.0	35.1	2379.7	0.2
4	**516.8**	0.0	6372.8	533.1	2442.5	3600.0	3.2	539.0	0.1
5	**106.6**	0.0	55.2	**106.6**	352.3	446.2	0.0	113.9	0.0
6	**1033.5**	0.0	800.1	1047.5	158.2	3600.0	1.4	1720.7	2.0
7	**558.1**	0.0	8061.4	588.3		3600.0	5.4	1527.9	0.1
8	645.2	0.3	10811.3	/	/	/	/	/	/
9	**135.7**	0.0	842.5	162.6	2937.2	3600.0	19.8	1955.3	0.2
10	**490.4**	0.0	1110.3	/	/	/	/	/	/
11	**410.9**	0.0	978.8	1201.2	3599.0	3600.0	192.3	2081.0	1.1
12	414.5	2.6	10804.1	2979.7	1930.6	3600.0	618.9	3486.6	3.4
13	7576.3	37.5	11113.0	/	/	/	/	/	/

5.3. Impact de l'ordre d'insertion des contraintes de coordination et de l'ordre de branchement

La troisième étape de l'approche CPDM est dépendante de l'ordre d'insertion des contraintes de coordination et de l'ordre de branchement des variables dans l'arbre de recherche PPC. D'autres ordres peuvent conduire à d'autres solutions, potentiellement meilleures.

L'objectif de cette section est d'étudier l'impact de l'ordre d'insertion des contraintes de coordination, et celui de l'ordre de branchement. Ces études sont conduites sur l'étape 3 de la CPDM, et plus particulièrement sur la fonction ICPA, avec l'instance 10L, car à la fin de l'étape 2 de la CPDM, aucune des huit contraintes de coordination n'est respectée.

5.3.1. Ordre d'insertion des contraintes de coordination

Durant la troisième étape de la CPDM et de la fonction ICPA, les contraintes de coordination sont insérées itérativement dans le modèle PPC. Cette étude se focalise sur l'impact des ordres d'insertion des contraintes.

Pour cette étude, l'ordre de branchement des variables dans l'arbre de la PPC est toujours le même. Dix séquences d'insertion des contraintes de coordination sont aléatoirement générées : la fonction ICPA est donc appelée 10 fois, avec des ordres d'insertion différents (mais les tournées initiales sont toujours les mêmes).

Le Tableau 15 présente les résultats numériques de cette expérience sur l'instance 10L. Le nombre de contraintes non respectées au début de la fonction ICPA est de 8 (colonne « Nb contraintes non respectées au début de ICPA »). À la fin de chaque exécution, le nombre de contraintes de coordination non vérifiées est 0 (colonne « Nb contraintes respectées à la fin de l'ICPA »). Cela signifie qu'à chaque exécution, l'ICPA est capable de trouver une solution respectant toutes les contraintes de coordination. Par ailleurs, la solution de coût plus faible

est obtenue pour les exécutions 2, 4, 5, 6 et 8, et vaut $S = 654$. Pour toutes les autres exécutions, $S = 670$. L'écart type du coût est de 8.4.

Tableau 15 Conséquence de l'ordre d'insertion des contraintes de coordination

Exécution	Nb contraintes non respectées au début de ICPA	Nb contraintes non respectées à la fin de ICPA	Coût de la solution S
1	8	0	670
2	8	0	654
3	8	0	670
4	8	0	654
5	8	0	654
6	8	0	654
7	8	0	670
8	8	0	654
9	8	0	670
10	8	0	670
Écart type		**0**	**8.4**

En conclusion de cette expérience, l'ordre d'insertion des contraintes de coordination n'impacte ni la qualité des résultats et ni la capacité de la méthode ICPA à obtenir une solution respectant toutes les contraintes de coordination.

5.3.2. Ordre de branchement dans l'arbre de PPC

Cette seconde étude concerne l'impact de l'ordre des variables lors du branchement dans l'arbre de PPC pour l'instance 10L, durant la procédure ICPA. En effet, il a déjà été souligné que les variables branchées en haut de l'arbre sont moins rapidement remises en question que des variables situées en bas de l'arbre (voir la section 4.4). Pour cette étude, 10 exécutions sont réalisées. L'ordre d'insertion des contraintes de coordination est identique, mais l'ordre de branchement des variables dans l'arbre de PPC est généré aléatoirement à chaque exécution.

Le Tableau 16 présente le nombre de contraintes non respectées au début de l'ICPA (seconde colonne), le nombre de contraintes de coordination vérifiées à la fin de l'ICPA (troisième colonne) et le coût S de la solution (quatrième colonne). Le nombre de contraintes de coordination est toujours le même en début de procédure ICPA, puisque ce sont les mêmes tournées qui sont données en entrée. Les exécutions 2, 4, 5 et 6 ne fournissent pas une solution où toutes les contraintes de coordination sont respectées : la déviation standard est de 0.5. Le coût S de la solution varie entre 610 (exécution 4) et 687 (exécution 2) : l'écart type est de 30.4.

Tableau 16 Conséquence de l'ordre de branchement dans l'arbre du solveur PPC

Exécution	Nb contraintes non respectées au début de ICPA	Nb contraintes non respectées à la fin de ICPA	Coût de la solution S
1	8	0	655
2	8	1	687
3	8	0	624
4	8	1	610
5	8	1	694
6	8	1	632
7	8	0	672
8	8	0	611
9	8	0	648
10	8	0	669
Écart type		0.5	30.4

Cette expérience met en lumière l'importance de l'ordre de branchement des variables dans l'arbre de PPC pour obtenir des solutions de bonne qualité. Elle souligne également l'intérêt d'une approche itérative limitée dans le temps par rapport à une unique exécution.

6. Conclusion

Ce chapitre aborde le Generalised Workforce Scheduling and Routing Problem (GWSRP) en prenant explicitement en considération l'ordonnancement et le transport. Le GWSRP est une généralisation du WSRP, car il comprend (en plus des contraintes du WSRP) des contraintes de coordination issues des communautés de l'ordonnancement et des communautés de tournées de véhicules. Ces contraintes permettent de modéliser des applications réelles ou des employés doivent se coordonner afin de réaliser leurs visites. Il peut s'agir d'une synchronisation permettant d'effectuer une lourde charge, ou d'une disjonction afin de ne pas se gêner mutuellement. Ces contraintes peuvent aussi concerner des délais minimal et/ou maximal à respecter entre deux visites. Elles sont fréquentes dans les problèmes de maintenance et de services à domicile.

Le GWSRP possède la même fonction objectif que le WSRP. La qualité de service des employés impose également au GWSRP de fournir une solution qui n'est pas nécessairement semi-active. La différence entre ces deux problèmes réside dans les contraintes de coordination qui modifient la modélisation et la fonction d'évaluation. Pour le WSRP, les tournées des employés sont indépendantes les unes des autres, et une fonction d'évaluation maximisant la qualité de service des employés peut être utilisée itérativement sur chaque tournée. Pour le GWSRP, les tournées sont dépendantes les unes des autres à cause des contraintes de coordination. La fonction d'évaluation doit donc considérer toutes les tournées en même temps.

Une des contributions du chapitre est l'introduction de ce nouveau problème : le GWSRP et la formalisation des contraintes de coordination. Une autre contribution consiste en la proposition d'un schéma de résolution en trois étapes : la Constraint-Programming based Decomposition Method (CPDM). La première étape correspond à la résolution WSRP par une génération de colonnes (donc sans prise en compte des contraintes de coordination) pour fournir un ensemble de tournées initiales. La seconde étape de la CPDM correspond à la sélection d'un sous-ensemble de tournées maximisant le nombre de contraintes de

coordination respectées. La troisième étape est l'insertion itérative par un solveur PPC de toutes les contraintes de coordination dans les tournées.

La CPDM est testée sur les instances de la littérature de (Bredström and Rönnqvist, 2008) pour le VRPTWSyn (Vehicle Routing and Scheduling Problem with Time Windows and Synchronized visits), en adaptant légèrement la fonction objectif et les contraintes sur les dates de début des employés. La CPDM est capable de fournir une solution à toutes les instances en un temps de calcul raisonnable. Un nouveau jeu d'instances spécifique au GWSRP est introduit. Ces instances comprennent un grand nombre de contraintes de coordination. La CPDM est capable de trouver des solutions, en un temps de calcul 10 fois plus petit que CPLEX, avec un gap d'environ 1% par rapport aux solutions optimales obtenues par PLNE.

7. Bibliographie

Afifi, S., Dang, D.-C., Moukrim, A., Dang, D.-C., Moukrim, A., 2013. A Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows and Synchronization Constraints, in: International Conference on Learning and Intelligent Optimization. Springer, Berlin, Heidelberg, pp. 259–265. https://doi.org/10.1007/978-3-642-44973-4_27

Afifi, S., Duc-Cuong, D., Moukrim, A., 2016. Heuristic solutions for the vehicle routing problem with time windows and synchronized visits. Optimization Letters 10, 511–525.

Aggoun, A., Beldiceanu, N., 1993. Extending chip in order to solve complex scheduling and placement problems. Mathematical and Computer Modelling 17, 57–73. https://doi.org/10.1016/0895-7177(93)90068-A

Algethami, H., Landa-Silva, D., Martínez-Gavara, A., 2017. Selecting Genetic Operators to Maximise Preference Satisfaction in a Workforce Scheduling and Routing Problem, in: Proceedings of the 6th International Conference on Operations Research and Enterprise Systems. Presented at the ICORES, Porto, Portugal, pp. 416–423. https://doi.org/10.5220/0006203304160423

Artigues, C., Huguet, M.-J., Lopez, P., 2011. Generalized disjunctive constraint propagation for solving the job shop problem with time lags. Engineering Applications of Artificial Intelligence 24, 220–231. https://doi.org/10.1016/j.engappai.2010.07.008

Bachouch, R.B., Guinet, A., Hajri-Gabouj, S., 2011. A Decision-Making Tool for Home Health Care Nurses' Planning. Supply Chain Forum: An International Journal 12, 14–20. https://doi.org/10.1080/16258312.2011.11517250

Backer, B.D., Furnon, V., Shaw, P., Kilby, P., Prosser, P., 2000. Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics. Journal of Heuristics 6, 501–523.

Bertels, S., Fahle, T., 2006. A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. Computers & Operations Research 33, 2866–2890. https://doi.org/10.1016/j.cor.2005.01.015

Blais, M., Lapierre, S.D., Laporte, G., 2003. Solving a home-care districting problem in an urban setting. Journal of the Operational Research Society 54, 1141–1147. https://doi.org/10.1057/palgrave.jors.2601625

Bockmayr, A., Hooker, J.N., 2005. Constraint programming, in: Handbooks in Operations Research and Management Science. pp. 559–600.

Bourreau, É., Garaix, T., Gondran, M., Lacomme, P., Tchernev, N., 2019a. Problèmes de coordination de tournées dans le cadre du WSRP : résolution par PPC. Presented at the 20ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Le Havre, France.

Bourreau, É., Gondran, M., Lacomme, P., 2019b. A local search for the Job-Shop Scheduling Problem with Constraint Programming. Presented at the 30th European Conference on Operational Research (EURO), Dublin, Ireland.

Bourreau, É., Gondran, M., Lacomme, P., Vinot, M., 2020. Programmation Par Contraintes : démarches de modélisation pour des problèmes d'optimisation. Ellipses.

Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I., 2016. The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering 99, 300–313. https://doi.org/10.1016/j.cie.2015.12.007

Bredström, D., Rönnqvist, M., 2008. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. European Journal of Operational Research 191, 19–31. https://doi.org/10.1016/j.ejor.2007.07.033

Bredström, D., Rönnqvist, M., 2007. A Branch and Price Algorithm for the Combined Vehicle Routing and Scheduling Problem With Synchronization Constraints. NHH Dept. of Finance & Management Science Discussion Paper, (2007/7). https://doi.org/10.2139/ssrn.971726

Brucker, P., Knust, S., 1999. Complexity Results for Single-Machine Problems with Positive Finish-Start Time-Lags. Computing 63, 299–316. https://doi.org/10.1007/s006070050036

Castillo, I., Joro, T., Li, Y.Y., 2009. Workforce scheduling with multiple objectives. European Journal of Operational Research 196, 162–170. https://doi.org/10.1016/j.ejor.2008.02.038

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R., 2016. Workforce scheduling and routing problems: literature survey and computational study. Annals of Operations Research 239, 39–67.

Caumond, A., Lacomme, P., Tchernev, N., 2008. A memetic algorithm for the job-shop with time-lags. Computers & Operations Research 35, 2331–2356. https://doi.org/10.1016/j.cor.2006.11.007

Chahed, S., Marcon, E., Sahin, E., Feillet, D., Dallery, Y., 2009. Exploring new operational research opportunities within the Home Care context: the chemotherapy at home. Health Care Manag Sci 12, 179–191. https://doi.org/10.1007/s10729-009-9099-6

Chankov, S., Hütt, M.-T., Bendul, J., 2018. Influencing factors of synchronization in manufacturing systems. International Journal of Production Research 56, 4781–4801. https://doi.org/10.1080/00207543.2017.1400707

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579–594. https://doi.org/10.1016/S0191-2615(02)00045-0

Desaulniers, G., 2010. Branch-and-Price-and-Cut for the Split-Delivery Vehicle Routing Problem with Time Windows. Operations Research 58, 179–192. https://doi.org/10.1287/opre.1090.0713

Desaulniers, G., Desrosiers, J., Solomon, M.M., 2005. Column generation, Springer Science & Business Media.

Dohn, A., Kolind, E., Clausen, J., 2009. The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. Computers & Operations Research 36, 1145–1157. https://doi.org/10.1016/j.cor.2007.12.011

Dohn, A., Rasmussen, M.S., Larsen, J., 2011. The vehicle routing problem with time windows and temporal dependencies. Networks 58, 273–289. https://doi.org/10.1002/net.20472

Drexl, M., 2012. Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints. Transportation Science 46, 297–316. https://doi.org/10.1287/trsc.1110.0400

Eveborn, P., Flisberg, P., Rönnqvist, M., 2006. Laps Care—an operational system for staff planning of home care. European Journal of Operational Research 171, 962–976. https://doi.org/10.1016/j.ejor.2005.01.011

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44, 216–229.

Fikar, C., Hirsch, P., 2017. Home health care routing and scheduling: A review. Computers & Operations Research 77, 86–95. https://doi.org/10.1016/j.cor.2016.07.019

Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., Soumis, F., 2019. Column generation for vehicle routing problems with multiple synchronization constraints. European Journal of Operational Research 272, 699–711. https://doi.org/10.1016/j.ejor.2018.06.046

Gendreau, M., Potvin, J.-Y., Bräumlaysy, O., Hasle, G., 2008. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography, in: The Vehicle Routing Problem: Latest Advances and New Challenges, Springer Science & Business Media. Boston, MA, pp. 143–169. https://doi.org/10.1007/978-0-387-77778-8

Gendreau, M., Tarantilis, C.D., 2010. Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Montreal, QC, Canada : Cirrelt.

Goel, A., Meisel, F., 2013. Workforce routing and scheduling for electricity network maintenance with downtime minimization. European Journal of Operational Research 231, 210–228. https://doi.org/10.1016/j.ejor.2013.05.021

Golden, B., Raghavan, S., Wasil, E. (Eds.), 2008. The Vehicle Routing Problem: Latest Advances and New Challenges, Springer Science & Business Media. https://doi.org/10.1007/978-0-387-77778-8

Haddadene, S.R.A., Labadie, N., Prodhon, C., 2016. A GRASP × ILS for the vehicle routing problem with time windows, synchronization and precedence constraints. Expert Systems with Applications 66, 274–294. https://doi.org/10.1016/j.eswa.2016.09.002

Hojabri, H., Gendreau, M., Potvin, J.-Y., Rousseau, L.-M., 2018. Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. Computers & Operations Research 92, 87–97. https://doi.org/10.1016/j.cor.2017.11.011

Hollis, B.L., Forbes, M.A., Douglas, B.E., 2006. Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post. European Journal of Operational Research 173, 133–150. https://doi.org/10.1016/j.ejor.2005.01.005

Karlsson, J., Rönnqvist, M., Bergström, J., 2004. An optimization model for annual harvest planning. Canadian Journal of Forest Research 34, 1747–1754.

Karoui, W., Huguet, M.-J., Lopez, P., Haouari, M., 2010. Climbing discrepancy search for flowshop and jobshop scheduling with time lags. Electronic Notes in Discrete Mathematics 36, 821–828. https://doi.org/10.1016/j.endm.2010.05.104

Labadie, N., Prins, C., Yang, Y., 2014. Iterated Local Search for a Vehicle Routing Problem with Synchronization Constraints, in: Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems. Presented at the INCORES, Angers, France, pp. 257–263. https://doi.org/10.5220/0004837502570263

Lacomme, P., Tchernev, N., Huguet, M.-J., 2012. Job-Shop with Generic Time-Lags: a heuristic based approach. Presented at the 9th International Conference of Modeling, Optimization and Simulation-MOSIM, pp. 06–08.

Lacomme, P., Tchernev, N., Huguet, M.-J., 2011. Dedicated constraint propagation for Job-Shop problem with generic time-lags. Presented at the ETFA2011, IEEE, pp. 1–7.

Laesanklang, W., Landa-Silva, D., Castillo-Salazar, J.A., 2016. An investigation of heuristic decomposition to tackle workforce scheduling and routing with time-dependent activities constraints, in: International Conference on Operations Research and Enterprise Systems. Springer, pp. 239–260.

Laesanklang, W., Pinheiro, R.L., Algethami, H., Landa-Silva, D., 2015. Extended decomposition for mixed integer programming to solve a workforce scheduling and routing problem, in: International Conference on Operations Research and Enterprise Systems. Springer, pp. 191–211.

Liu, R., Tao, Y., Xie, X., 2018. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. Computers & Operations Research. https://doi.org/10.1016/j.cor.2018.08.002

Maenhout, B., Vanhoucke, M., 2009. The impact of incorporating nurse-specific characteristics in a cyclical scheduling approach. Journal of the Operational Research Society 60, 1683–1698. https://doi.org/10.1057/jors.2008.131

Mankowska, D.S., Meisel, F., Bierwirth, C., 2014. The home health care routing and scheduling problem with interdependent services. Health Care Management Science 17, 15–30. https://doi.org/10.1007/s10729-013-9243-1

Masmoudi, M.A., Cheikhrouhou, N., 2018. Heterogeneous Vehicle Routing Problems with Synchronization: Application to Homecare Scheduling Routing Problem, in: Actes GISEH 2018. Geneva, Switzerland.

Mbiadou Saleu, R.G., Deroussi, L., Feillet, D., Grangeon, N., Quilliot, A., 2018. An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. Networks 72, 459–474. https://doi.org/10.1002/net.21846

Mitten, L.G., 1959. Sequencing n jobs on two machines with arbitrary time lags. Management science 5, 293–298.

Nguyen, V.-P., Prins, C., Prodhon, C., 2010. A multi-start evolutionary local search for the two-echelon location routing problem. Presented at the International Workshop on Hybrid Metaheuristics, Springer, Berlin, Heidelberg, pp. 88–102.

Parragh, S.N., Doerner, K.F., 2018. Solving routing problems with pairwise synchronization constraints. Central European Journal of Operations Research 26, 443–464. https://doi.org/10.1007/s10100-018-0520-4

Potvin, J.-Y., Kervahut, T., Garcia, B.-L., Rousseau, J.-M., 1996. The vehicle routing problem with time windows part I: tabu search. INFORMS Journal on Computing 8, 158–164.

Pour, S.M., Drake, J.H., Ejlertsen, L.S., Rasmussen, K.M., Burke, E.K., 2018. A hybrid Constraint Programming/Mixed Integer Programming framework for the preventive signaling maintenance crew scheduling problem. European Journal of Operational Research 269, 341–352. https://doi.org/10.1016/j.ejor.2017.08.033

Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J., 2012. The Home Care Crew Scheduling Problem: Preference-based visit clustering and temporal dependencies. European Journal of Operational Research 219, 598–610. https://doi.org/10.1016/j.ejor.2011.10.048

Rossi, F., van Beek, P., Walsh, T., 2006. Handbook of Constraint Programming, Elsevier.

Rousseau, L.-M., Gendreau, M., Pesant, G., 2013. The Synchronized Dynamic Vehicle Dispatching Problem. INFOR: Information Systems and Operational Research 51, 76–83. https://doi.org/10.3138/infor.51.2.76

Roy, B., Sussmann, B., 1964. Les problèmes d'ordonnancement avec contraintes disjonctives. Note Ds 9.

Simonis, H., Cornelissens, T., Simonis, H., Cornelissens, T., 1995. Modelling producer/consumer constraints, in: Montanari, U., Rossi, F. (Eds.), Principles and Practice of Constraint Programming — CP '95. Springer, Berlin, Heidelberg, pp. 449–462. https://doi.org/10.1007/3-540-60299-2_27

Solomon, M.M., Desrosiers, J., 1988. Survey Paper—Time Window Constrained Routing and Scheduling Problems. Transportation Science 22, 1–13. https://doi.org/10.1287/trsc.22.1.1

Toth, P., Vigo, D. (Eds.), 2014. Vehicle routing: problems, methods, and applications, Society for Industrial and Applied Mathematics.

Toth, P., Vigo, D. (Eds.), 2002. The vehicle routing problem, SIAM monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics, Philadelphia, Pa.

Xiang, Z., Chu, C., Chen, H., 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. European Journal of Operational Research 174, 1117–1139. https://doi.org/10.1016/j.ejor.2004.09.060

Conclusion générale

Ce manuscrit aborde des problèmes de production et de transport intégrés, présents dans les milieux industriels et dans les services à domicile. Ces problèmes consistent à planifier la production de pièces sur des machines ou la réalisation de visites à domicile, en prenant en compte le transport qui doit être réalisé pour déplacer des pièces entre des machines ou pour déplacer des cadres de santé entre les patients. Ces problèmes se modélisent sous la forme de graphes qui permettent une modélisation explicite du transport.

Ces problèmes intégrés sont très différents des problèmes classiques d'ordonnancement ou de transport car, dans la majorité des cas, il ne s'agit pas de minimiser un critère unique comme le makespan ou la somme des longueurs des tournées. Généralement, dans ces problèmes intégrés, un critère est lié à la qualité de la solution et un ou plusieurs critères sont liés à la qualité de service soit des pièces (ou patients) transportées, soit des personnels (chauffeurs) réalisant les opérations de transport.

La plupart du temps, les problèmes « classiques » de production et de transport intégrés sont modélisés sous forme de graphes et des fonctions d'évaluation sont utilisées. Ces fonctions fournissent des solutions semi-actives, où le critère minimisé est uniquement un critère de coût ou de temps. La partie transport est prise en compte uniquement sous la forme de contraintes qui doivent être vérifiées, mais n'est pas explicitement présente dans la fonction objectif. Ces problèmes sont référencés sous la dénomination « d'ordonnancement avec transport » par opposition aux problèmes de type « ordonnancement avec routing » qui, eux, modélisent explicitement le « routing » et le prennent en compte dans la fonction objectif. En général, la prise en considération d'un critère de qualité de service impose que la solution recherchée ne soit pas semi-active, et donc de définir de nouvelles fonctions d'évaluation.

Les travaux de ce manuscrit s'inscrivent dans cette voie de recherche et sont spécifiquement orientés sur trois problèmes d'ordonnancement avec transport, où la fonction objectif inclut un critère de qualité de service dont une partie porte sur le transport. Ces trois problèmes peuvent alors être qualifiés de problèmes d'ordonnancement avec « routing ».

La première partie des travaux concerne un nouveau problème, le Job-shop Scheduling Problem with Routing (JSPR), qui est un Job-shop Scheduling Problem with Transport (JSPT) prenant en compte un critère de qualité de service. Dans le JSPR, le problème de transport appartient à la famille des Dial-A-Ride Problems. La qualité de service inclut le temps passé par les pièces dans les véhicules, mais aussi dans les buffers d'entrée et de sortie des machines, et la durée totale pour fabriquer une pièce (ce dernier critère est un critère souvent désigné par « Mean Flow Time »). Les contributions de ce chapitre sont :

- La proposition d'un critère de qualité de service pour le JSPR.
- L'introduction d'une fonction d'évaluation d'un graphe disjonctif (modélisant le JSPR), nommée Time-Lag Heuristic (TLH) qui minimise le makespan et maximise la qualité de service.
- La définition d'une métaheuristique définissant un cadre complet de résolution.
- La validation de la fonction TLH avec des résolutions optimales obtenue par PLNE.

La définition d'une fonction d'évaluation d'un graphe disjonctif (modélisant le JSPR), nommée Time-Lag Heuristic (TLH) s'inscrit dans la continuité de la contribution de (Cordeau and Laporte, 2003). Ces derniers proposent une fonction d'évaluation qui permet la maximisation de la qualité de service des clients dans un Dial-A-Ride Problem. En se basant sur l'utilisation des time-lags entre opérations, l'évaluation TLH fournit une solution, non semi-active, qui minimise le makespan et maximise la qualité de service. Le JSPR étudié prend en compte une flotte de véhicules de capacité unitaire et une flotte de véhicules de capacité non unitaire. Les études numériques montrent que la fonction d'évaluation TLH est capable de trouver des solutions très proches des solutions optimales obtenues par PLNE. Les études mettent également en avant que l'insertion de la fonction d'évaluation TLH dans une métaheuristique permet d'obtenir des résultats comparables avec la littérature pour le critère du makespan seul. Enfin, les études démontrent qu'il y a un véritable gain concernant la qualité de service lorsque l'évaluation TLH est utilisée, en comparaison avec une évaluation du graphe disjonctif, par un algorithme classique de plus long chemin de type Bellman-Ford (Ford and Lester, 1956; Bellman, 1958; Moore, 1959).

La seconde partie des travaux portent sur le Workforce Scheduling and Routing Problem (WSRP), qui est un problème combinant production, transport et affectation. Il relève typiquement des problèmes de maintenance ou de services à domicile. Toutefois, le WSRP se différencie des problèmes « classiques de tournées de véhicules » par la prise en compte de la qualité de service, qui concerne les employés et les clients (Castillo-Salazar et al., 2016). Elle dépend entre autres des dates de début des visites, et impose que la solution ne soit pas semi-active. Les points clés de cette seconde partie sont :

- Une fonction d'évaluation qui maximise la qualité de service.
- La définition d'un modèle Programmation Par Contraintes (PPC) qui permet de trouver plus rapidement que la PLNE une solution faisable de bonne qualité.
- La conception d'un schéma de résolution de type Génération de Colonnes.
- La définition d'un nouveau de jeu de données plus réaliste que les jeux de données utilisés précédemment dans la littérature.

Une des contributions est la proposition d'une fonction d'évaluation d'une tournée fournissant les dates de début des visites qui maximisent la qualité de service. Cette fonction d'évaluation est une nouvelle extension de l'algorithme à labels qui est défini pour Elementary Shortest Path Problem With Resource Constraints (Feillet et al., 2004). Un modèle PLNE et un modèle de Programmation Par Contraintes (PPC) sont également proposés, de même qu'une Génération de Colonnes où le problème maître est soit un Set Partitioning Problem soit un Set Covering Problem. Le problème esclave est formulé sous forme de PLNE, et une nouvelle procédure de programmation dynamique est présentée pour le résoudre.

La troisième partie des travaux aborde le Generalised Workforce Scheduling and Routing Problem (GWSRP) qui étend le WSRP, en ajoutant des contraintes de coordination entre les visites. La majorité des travaux de la littérature se concentrent sur des contraintes de coordination limitées à des synchronisations des dates de visite ; tandis que le GWSRP est défini avec neuf types de contraintes de coordination, issues des problèmes classiques de tournées de véhicules et des problèmes classiques d'ordonnancement. Ces contraintes de coordination impliquent des dépendances entre les tournées des différents employés, à la différence du WSRP où les tournées sont indépendantes les unes des autres. Les points clés de nos travaux sont :

- La définition d'un nouveau problème qui étend le WSRP.

- La conception d'une approche de résolution itérative qui fait appel à une Génération de Colonnes et un modèle de Programmation Par Contraintes.
- La prise en compte simultanée de neuf types de contraintes de coordination qui aboutissent à des dépendances entre les tournées. Ces contraintes de coordination sont modélisées par des time-lags minimaux, des time-lags maximaux, et en plus des disjonctions qui créent une combinatoire sur l'ordre des visites.

Les contraintes de coordination nécessitent l'ajout de time-lags minimaux et maximaux et/ou des disjonctions rendant le problème très contraint. Notre proposition repose sur la conception d'un schéma de résolution globale – Constraint-Programming based Decomposition Method (CPDM) – où des tournées sont générées sans prendre en compte de contraintes de coordination par une génération de colonnes. Les contraintes de coordination sont ensuite insérées itérativement dans les tournées par un modèle PPC. Un nouveau jeu d'instances est introduit. La CPDM est capable de trouver, en un temps de calcul faible, des solutions très proches de celles optimales obtenues par PLNE.

Les travaux qui ont été présentés dans ce manuscrit offrent plusieurs perspectives.

À court et moyen termes, un modèle PPC pourrait être proposé pour le JSPR, et des approches hybrides pourraient être envisagées. Dans ces dernières, la PPC servirait à évaluer le graphe disjonctif et pourrait définir une recherche locale. Des prémisses à ce type de recherches, sur le Job-shop Scheduling Problem, sont proposées dans (Bourreau et al., 2020) et ont fait l'objet d'une conférence (Bourreau et al., 2019). Il a également démontré que la PPC peut servir de recherche locale dans les problèmes de tournées de véhicules (Rousseau et al., 2013; Hojabri et al., 2018).

Il faut aussi noter que des contraintes supplémentaires pourraient être ajoutées dans le JSPR. Ces contraintes concerneraient la partie transport : par exemple, une distance maximale que les véhicules peuvent parcourir et qui peut se justifier par l'usage de véhicules électriques et donc par le besoin de recharger les batteries du véhicule.

Le JSPR pourrait prendre en compte des contraintes sur les ordres de chargement et de déchargement des pièces dans les véhicules : ainsi, la première pièce à pouvoir être livrée serait celle chargée en dernière dans le véhicule. Cet axe de recherche s'inscrirait dans la suite des travaux publiés par (Erdoğan et al., 2009) pour le pickup and delivery traveling salesman problem with first-in-first-out loading. De manière similaire, des contraintes de type bin-packing pourraient être établies pour le transport, dans le même courant que les travaux de (Iori et al., 2007) et (Malapert et al., 2008) concernant le Vehicle Routing Problem with two-dimensional Loading constraints. Un autre axe de recherche serait de permettre aux véhicules de s'échanger les produits transportés. Dans ce cas, le transport du JSPR se rapprocherait des problèmes de Vehicle Routing Problem with Trailers and Transshipments (Gendron and Semet, 2009; Drexl, 2013) et imposerait des contraintes de coordination entre les tournées.

Concernant le WSRP, des recherches pourraient être orientées vers une procédure de type SPLIT, qui permet de passer d'une représentation indirecte à une solution évaluée. Cette procédure pourrait ensuite être incluse dans une métaheuristique. Enfin, une fonction d'évaluation du GWSRP et des opérateurs de recherche locale basés sur le graphe pourraient être investigués.

Pour le WSRP et le GWSRP, la qualité de service pourrait être redéfinie pour prendre en compte la durée d'attente de l'employé entre les visites, ou bien un ratio entre le temps de travail et le temps de transport. Ceci peut être particulièrement pertinent en milieu urbain où des cadres de santé passent parfois plus de temps dans les transports qu'à l'exercice de leur métier. De même, les délais entre deux visites constituent un facteur d'acceptabilité des solutions pour éviter des emplois du temps socialement peu acceptables pour les salariés. Dans cette optique, il pourrait même être envisagé que les employés puissent retourner chez eux si le délai entre deux visites dépasse une certaine durée. Des notions de pauses et de repas pourraient également être prises en compte (Gérard et al., 2016), ou encore le besoin de mutualiser les véhicules pour transporter simultanément plusieurs employés (Fink et al., 2019).

Sur le long terme des axes de recherche nouveaux sont à définir dans la continuité des travaux actuels, en se basant sur une meilleure résolution intégrée des problèmes. Dans cette optique, deux axes prioritaires de recherches semblent intéressants à développer, puisqu'ils correspondent à des enjeux sociétaux ou scientifiques majeurs liés à la qualité de service au niveau du transport ou à un aspect économique.

De ce point de vue, un enjeu scientifique pourrait être de s'interroger sur la manière d'utiliser au mieux les avancées des différentes communautés qui parfois abordent de manière complémentaire des problématiques identiques. L'exemple typique est le problème de Job-Shop avec Routing traité au chapitre 2 et qui appartient ou étend les problèmes d'ordonnancement avec time-lags minimaux et maximaux. Les méthodes et outils de la communauté « ordonnancement » pourraient être avantageusement revisités pour définir une nouvelle approche au GWSRP. Les tournées des véhicules pourraient être considérées comme l'équivalent des gammes du Job-Shop dans le sens où cela définit des contraintes temporelles entre certains sous-ensembles d'opérations. Le problème reste alors d'adapter les outils de type graphe disjonctif liés à l'ordonnancement à ces problèmes de tournées.

Finalement, un axe de recherche pourrait porter sur des problèmes de type GWSRP multi-périodes en considérant qu'il s'agit de demandes de soins (ou de services) à fournir de manière régulière aux usagers. La notion d'équité dans ce type de résolution apparaît alors naturelle puisque le GWSRP, d'une part, autorise à ne pas traiter tous les clients et, d'autre part, introduit la notion de qualité de service pour les usagers. Ainsi, la version multi-périodes du GWSRP pourrait avoir une fonction objectif (éventuellement multi-objectifs) qui définit une notion d'équité entre clients pour éviter qu'un client ne soit jamais traité sur l'ensemble de l'horizon de planification. Naturellement, il est envisageable que le passage d'un cadre de santé soit obligatoire chez certains clients sous peine de mettre en danger leur vie ; tandis que pour d'autres patients, le service doit avoir lieu « si possible ».

Références :

Bellman, R., 1958. On a routing problem. Quarterly of applied mathematics 16, 87–90.

Bourreau, É., Gondran, M., Lacomme, P., 2019. A local search for the Job-Shop Scheduling Problem with Constraint Programming. Presented at the 30th European Conference on Operational Research (EURO), Dublin, Ireland.

Bourreau, É., Gondran, M., Lacomme, P., Vinot, M., 2020. Programmation Par Contraintes : démarches de modélisation pour des problèmes d'optimisation. Ellipses.

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R., 2016. Workforce scheduling and routing problems: literature survey and computational study. Annals of Operations Research 239, 39–67.

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological 37, 579–594. https://doi.org/10.1016/S0191-2615(02)00045-0

Drexl, M., 2013. Applications of the vehicle routing problem with trailers and transshipments. European Journal of Operational Research 227, 275–283. https://doi.org/10.1016/j.ejor.2012.12.015

Erdoğan, G., Cordeau, J.-F., Laporte, G., 2009. The pickup and delivery traveling salesman problem with first-in-first-out loading. Computers & Operations Research 36, 1800–1808. https://doi.org/10.1016/j.cor.2008.05.005

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44, 216–229.

Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., Soumis, F., 2019. Column generation for vehicle routing problems with multiple synchronization constraints. European Journal of Operational Research 272, 699–711. https://doi.org/10.1016/j.ejor.2018.06.046

Ford, J., Lester, R., 1956. Network flow theory. Rand Corp Santa Monica Ca.

Gendron, B., Semet, F., 2009. Formulations and relaxations for a multi-echelon capacitated location–distribution problem. Computers & Operations Research 36, 1335–1355. https://doi.org/10.1016/j.cor.2008.02.009

Gérard, M., Clautiaux, F., Sadykov, R., 2016. Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce. European Journal of Operational Research 252, 1019–1030. https://doi.org/10.1016/j.ejor.2016.01.036

Hojabri, H., Gendreau, M., Potvin, J.-Y., Rousseau, L.-M., 2018. Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. Computers & Operations Research 92, 87–97. https://doi.org/10.1016/j.cor.2017.11.011

Iori, M., Salazar-González, J.-J., Vigo, D., 2007. An Exact Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints. Transportation Science 41, 253–264. https://doi.org/10.1287/trsc.1060.0165

Malapert, A., Guéret, C., Jussien, N., Rousseau, L.-M., 2008. Two-dimensional pickup and delivery routing problem with loading constraints, in: First CPAIOR Workshop on Bin Packing and Placement Constraints (BPPC'08). p. 184.

Moore, E.F., 1959. The shortest path through a maze, in: Proc. Int. Symp. Switching Theory 1959. pp. 285–292.

Rousseau, L.-M., Gendreau, M., Pesant, G., 2013. The Synchronized Dynamic Vehicle Dispatching Problem. INFOR: Information Systems and Operational Research 51, 76–83. https://doi.org/10.3138/infor.51.2.76

Annexes

1. Différentes idées pour un schéma de résolution itératif

La troisième étape de la CPDM est itérée plusieurs fois afin d'obtenir des solutions qui vérifient toutes les contraintes de coordination ou des solutions de bonne qualité. Il est démontré que ce choix est efficace lors des expériences numériques. Une unique itération de cette troisième étape ne permet pas toujours à la CPDM de fournir une solution qui respecte toutes les contraintes de coordination, ou d'obtenir des solutions de bonne qualité. L'itération dans le schéma global de la CPDM est mise en évidence par la flèche rouge sur la Figure 1.

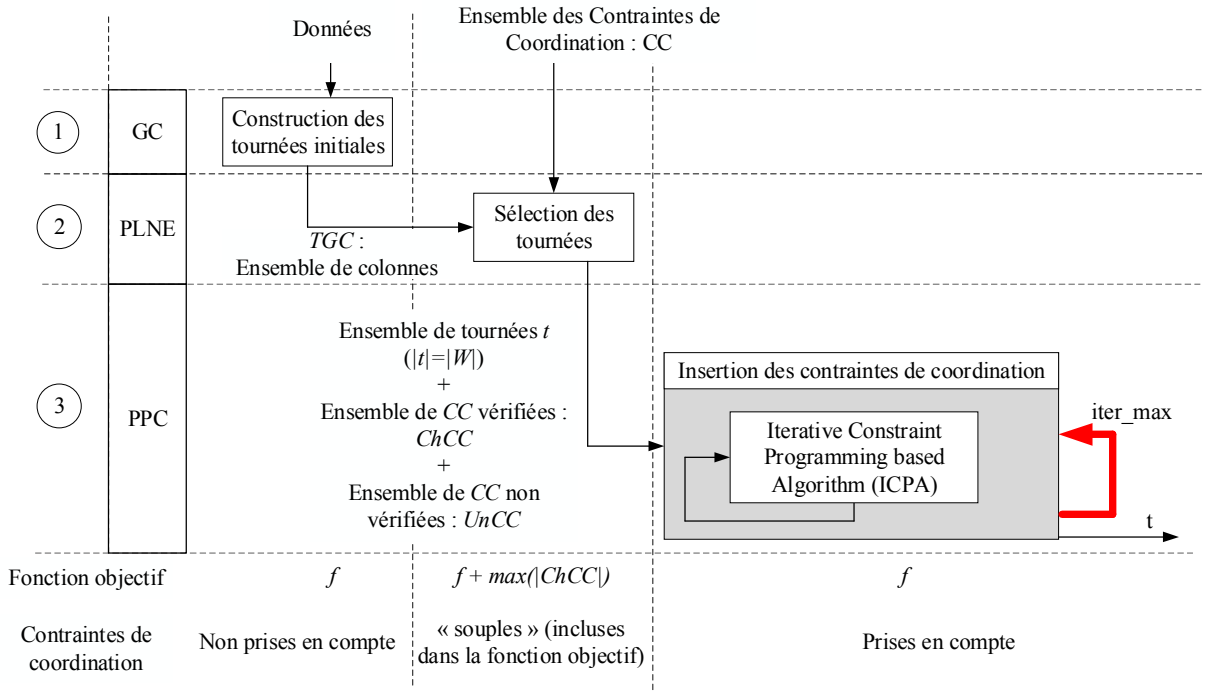


Figure 1 Rappel du schéma global de la CPDM

Plusieurs autres schémas itératifs ont été envisagés pour obtenir des solutions vérifiant toutes les contraintes de coordination ou de meilleure qualité, sans aboutir à de résultats significatifs. Des tests ont été effectués afin d'itérer soit sur l'étape 2 de la CPDM pour sélectionner un nouvel ensemble de tournées comme point de départ de l'étape 3 (section 1.1), soit sur l'étape 1 pour générer de nouvelles tournées (section 1.2).

1.1. Itération sur la solution en nombres entiers

La première idée consiste à itérer sur l'étape 2 de la CPDM (Figure 2). Lors de cette étape, un ensemble TGC issu de la génération de colonnes, est transmis en entrée. Cette phase retourne un ensemble t de tournées ($|t| = |W|$: une tournée par employé), un ensemble $ChCC$ de contraintes de coordination respectées, et un sous-ensemble $UnCC$ de contraintes de

coordination qui n'ont pas été prises en compte. L'objectif de cette phase, limitée en durée d'exécution, est de fournir une solution maximisant le nombre de contraintes de coordination respectées, et minimisant la valeur de la fonction objectif du GWSRP.

Un autre choix de sélection des tournées t peut amener à des ensembles t, $ChCC$ et $UnCC$ différents, et entraîner lors de troisième étape de la CPDM une solution différente et meilleure. L'idée est de tirer parti de la solution et des informations obtenues lors des itérations précédentes pour guider la seconde étape de la CPDM.

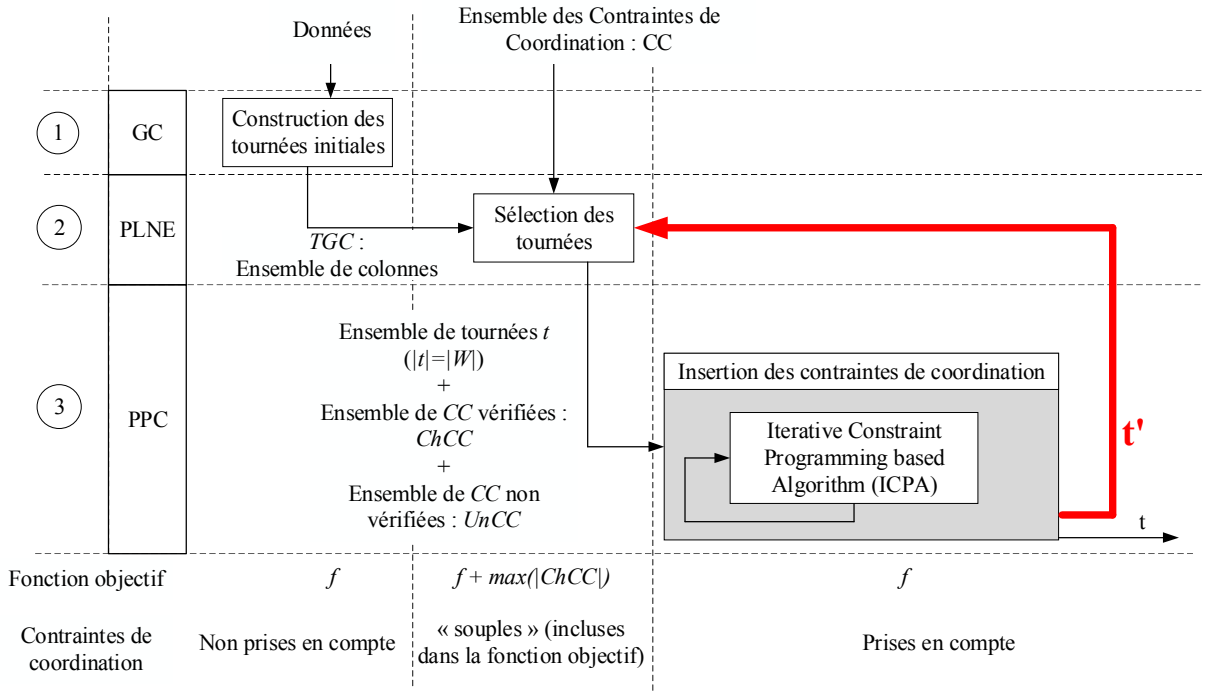


Figure 2 Itération avec la sélection des tournées

La question se pose de savoir est comment fournir, à l'itération i, un nouvel ensemble t^i qui soit différent de l'ensemble t^{i-1} de l'itération précédente. Des tests ont été réalisés en insérant des contraintes qui interdisent deux colonnes y_{k_1} et y_{k_2}, présentes dans la solution t^{i-1}, d'être sélectionnées ensemble ($y_{k_1} + y_{k_2} \leq 1$) dans la solution t^i. Ces contraintes ont également été généralisées à un ensemble de colonnes (et non plus spécifiquement à deux colonnes). Généralement, soit le solveur n'arrive plus du tout à trouver de solutions, soit il sélectionne des colonnes quasiment identiques qui sont réalisées par un autre employé. Par exemple, la colonne y_1 est réalisée par w_1 et y_2 par w_2. Alors les nouvelles tournées t^i vont inclure la colonne y_3 dont la séquence des visites est la même que pour la colonne y_1, mais est réalisée par w_2. Ajouter des contraintes dans le Set Partitionning Problem n'aide pas le solveur dans le temps imparti et ne permet pas d'assurer l'obtention d'une nouvelle solution différente.

Des tests avec des « warm starts » ont été effectués pour repartir d'une solution donnée, mais les résultats ne sont pas concluants. Le warm start est l'équivalent en PLNE, pour CPLEX, de démarrer d'une solution en PPC. Une première solution est donnée à CPLEX, qui commence sa recherche à partir de cette solution.

Une autre idée est d'interdire un « motif » (ou pattern) dans les tournées. Par exemple, dans la solution t^i, il est interdit que la visite v_1 soit directement suivie de la visite v_2, ou alors que les visites v_3 et v_4 soient dans la même tournée. Mais là encore, aucun résultat satisfaisant

n'a été obtenu. La question fondamentale est donc la suivante : comment choisir le schéma à interdire et sur quel critère ?

Afin de ne pas dégrader la solution, les nouvelles tournées t'^i trouvées à la fin de l'étape 3 de la CPDM sont insérées dans l'ensemble des tournées TGC issues de l'étape 1 de la CPDM. Mais lors de l'étape 2, la limite de temps autorisée pour la résolution du Set Partitioning Problem empêche au solveur de trouver une solution à l'itération $i + 1$ qui soit meilleure qu'à l'itération i, malgré la présence de ces nouvelles tournées t'^i. Il est raisonnable de penser qu'une solution à l'itération $i + 1$, meilleure que celle de l'itération i, inclue des tournées issues de l'ensemble t'^i. Il est même fréquent que la meilleure solution (c'est-à-dire les tournées t') ne soit pas trouvée dans la limite de temps de deux minutes.

Enfin, le dernier frein à une itération remontant à l'étape 2 réside dans le fait que cette étape de la CPDM ne permet pas la création de nouvelles tournées, mais correspond plutôt à un filtre entre la génération de colonnes et la phase de réparation. Si les tournées en entrée de cette étape ne sont pas intéressantes, alors il est impossible d'obtenir une solution plus intéressante ensuite.

En conclusion, aucune proposition n'a permis d'aboutir à la création d'une procédure qui itère sur la seconde étape de la CPDM.

1.2. Itération sur la construction des tournées initiales

Puisqu'un schéma itératif bouclant sur la seconde étape de la CPDM semble compliqué à mettre en place, des pistes de réflexion sont proposées pour un schéma itérant sur la première étape de la CPDM, c'est-à-dire pour la construction des tournées initiales (Figure 3).

L'objectif de la première étape de la CPDM est de fournir un ensemble TGC de tournées initiales. Ces tournées sont créées pas une génération de colonnes qui ne prend pas en compte les contraintes de coordination.

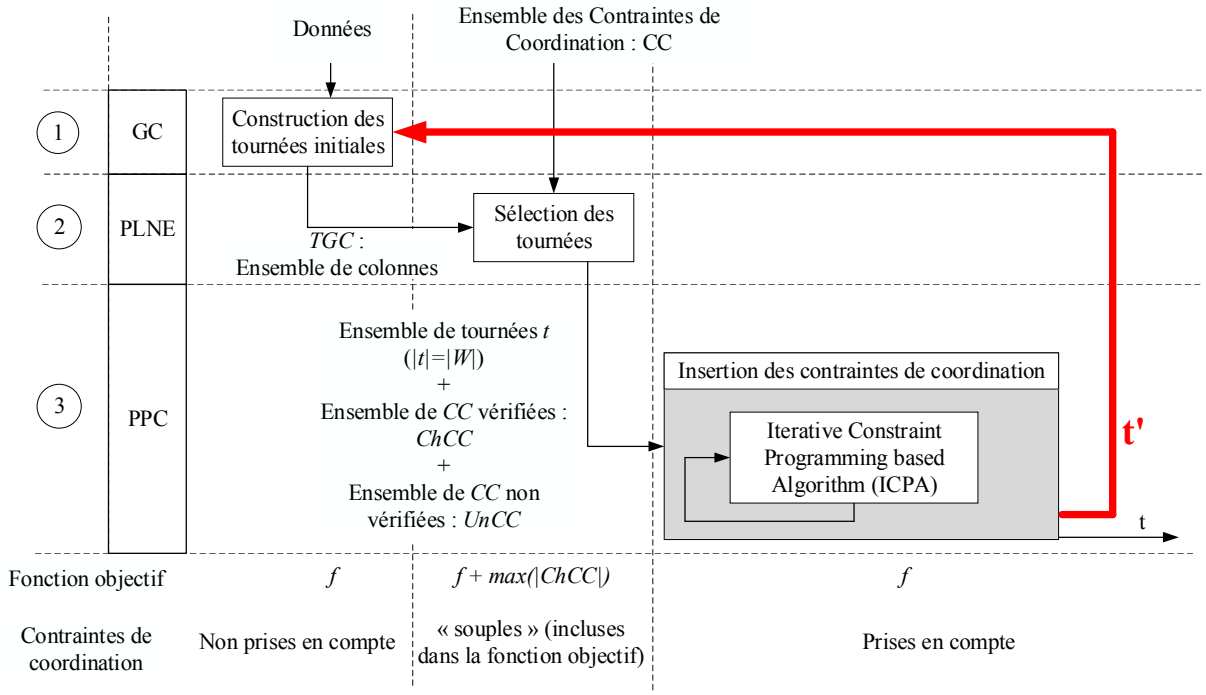


Figure 3 Itération sur la construction des tournées initiales

Ce nouveau schéma itératif se heurte à plusieurs difficultés dont la première est de savoir comment « forcer » la génération de colonnes à générer de nouvelles colonnes. Puisque la génération de colonnes est déterministe, les mêmes colonnes sont obtenues à chaque fois qu'elle s'exécute.

La première idée pour obtenir un schéma itératif est d'insérer les colonnes t', qui définissent une solution à la fin de l'étape 3, dans l'ensemble Ω des colonnes de la génération de colonnes, puis de relancer celle-ci (avec une base de colonnes donc enrichies). Le problème est qu'il est très peu probable que les colonnes t' soient prises en compte dans la solution fractionnaire du problème maître, car celles-ci n'ont pas été trouvées lors de la résolution précédente. Par ailleurs, elles permettent de prendre en compte les contraintes de coordination, donc elles ont probablement un coût plus élevé que les colonnes déjà présentes dans la solution sans contrainte de coordination. Cette solution sans contrainte de coordination est une borne inférieure au problème. Si aucune colonne issue de l'ensemble t' n'est prise en compte dans la solution fractionnaire de la GC, alors les variables duales sont identiques à celles de la dernière résolution de la génération de colonnes durant laquelle la résolution des problèmes esclaves n'a pas pu fournir de nouvelle colonne.

Une seconde idée pour la création d'un schéma itératif est d'insérer des contraintes de coordination (toutes ou un sous-ensemble) dans le problème maître. Cette idée a déjà été discutée dans le chapitre 4. Prendre en considération les contraintes de coordination durant la génération de colonnes est délicat, car dans ce cas, les colonnes ne sont plus indépendantes les unes des autres. De plus, l'insertion des contraintes de coordination impose une prise en compte explicite des dates de début des visites par le problème maître (ce qui n'est pas nécessaire sans ces contraintes). En ajoutant les dates de début des visites, le problème maître devient plus difficile à résoudre, et surtout la valeur de la variable duale associée à chaque visite est dépendante du temps, la résolution du problème esclave requiert donc de considérer le coût de la tournée en fonction de la date de début de la visite.

Une troisième idée pour le schéma itératif est d'interdire ou de rendre obligatoires certains arcs dans le problème esclave, à la manière d'un Branch-and-Price, mais la question de savoir quels arcs autoriser ou interdire est une question difficile. De plus cette insertion ne garantit pas de trouver de nouvelles colonnes.

Une autre proposition est de « bruiter » les variables des duales en les modifiant légèrement, avec une part d'aléatoire, lorsque la génération de colonnes est relancée. L'objectif est de permettre au problème esclave de trouver de nouvelles colonnes avec un coût réduit négatif afin de les ajouter au problème maître. Le grand risque de cette stratégie est de ne plus pouvoir obtenir une génération de colonnes qui converge. La résolution du problème esclave avec des variables duales bruitées peut permettre de trouver de nouvelles colonnes de coût réduit négatif, mais qui ne seront pas prises en compte par le problème maître. En effet, les mêmes colonnes avec les valeurs initiales des duales n'auraient pas un coût réduit négatif. Si les variables duales ne sont pas bruitées aléatoirement, une colonne remontée par le problème esclave est forcément utilisée par le problème maître à l'itération suivante.

En créant un schéma itératif qui boucle sur la première étape, la dernière difficulté réside dans la seconde étape de la CPDM : celle-ci nécessite un temps de calcul relativement long. Cette seconde étape résout le Set Partitioning Problem sur l'ensemble des tournées TGC obtenues par la GC. La seconde étape de la CPDM est limitée dans le temps, et l'ajout de colonnes dans l'ensemble TGC ne permet pas forcément d'améliorer la qualité lors de cette seconde étape. Cette seconde étape agit comme un « entonnoir » entre l'ensemble des colonnes TGC issu de la GC et qui comprend une multitude de tournées, et l'ensemble t (retourné par l'étape 2) qui contient une unique tournée pour chaque employé. Des essais ont été menés en supprimant une partie des colonnes de l'ensemble TGC pour aider la résolution de la seconde étape, mais le fait de trop réduire cet ensemble risque d'empêcher de trouver une solution faisable, et il est difficile de prédire à priori quelles colonnes il vaut mieux conserver.