Résolution conjointe de problèmes d'ordonnancement et de routage

Marina Vinot

N° d'ordre EDSPIC : 815

Université Clermont Auvergne

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

Thèse

présentée par

Marina VINOT

pour obtenir le grade de

Docteur d'Université

Spécialité : Informatique

Résolution conjointe de problèmes d'ordonnancement et de routage

Soutenue publiquement le 26 octobre 2017 devant le jury composé de :

Président :

Jean-Charles BILLAUT	Professeur des Universités, Université François-Rabelais de Tours

Rapporteurs :

Christian ARTIGUES	Directeur de Recherche, CNRS-LAAS
Caroline PRODHON	Maître de Conférences, HDR, Université de Technologie de Troyes

Examinateur :

Jean-Charles BILLAUT	Professeur des Universités, Université François-Rabelais de Tours

Directeurs de thèse :

Philippe LACOMME	Maître de Conférences, HDR, Université Clermont Auvergne
Aziz MOUKRIM	Professeur des Universités, Université de Technologie de Compiègne
Alain QUILLIOT	Professeur des Universités, Université Clermont Auvergne

Ce travail a bénéficié d'un financement géré par l'Agence Nationale de la Recherche (Ref. : ANR-11-IDEX-0004-02) au titre du programme « Investissements d'avenir » dans le cadre du projet ATHENA (Ref. : ANR-13-BS02-0006).

REMERCIEMENTS

J’aimerais tout d’abord remercier mes directeurs de thèse, Philippe Lacomme, Aziz Moukrim et Alain Quilliot, pour m'avoir encouragée dans cette voie. Ils ont été disponibles tout au long de ce travail de recherche et ont été des encadrants exemplaires par leurs compétences, leur partage et leur gentillesse.

Je remercie Christian Artigues, Jean-Charles Billaut et Caroline Prodhon pour avoir accepté de faire partie de mon jury de thèse. Leur lecture attentive et leurs remarques pertinentes ont permis d'améliorer la qualité de ce manuscrit.

Je remercie mes collègues et amis (en particulier Sahar, Alexis, Raksmey, Maxime, Matthieu, Damien, Benajim, Arnaud, Benjamin, David … pour ne citer qu'eux) pour leurs conseils et pour les bons moments que nous avons partagés. Je n’oublie pas la team du PTC que je remercie pour leur bonne humeur et leur folie.

Je souhaite remercier aussi toutes les personnes avec lesquelles j’ai discuté et qui ont pu m’aider à avancer au cours de cette thèse (Christophe Duhamel, Nikolay Tchernev, Hélène Toussaint, Gérard Fleury et beaucoup d’autres) ainsi que tout le personnel du LIMOS pour leur professionnalisme et leur réactivité.

Je tiens également à dire un grand merci à mes parents et à ma petite sœur pour leur soutien inconditionnel et la joie qu’ils m’apportent.

Enfin, je tiens à remercier mon conjoint pour m'avoir toujours encouragée et pour la patience dont il a fait preuve pour relire ce manuscrit.

RESUME

Cette thèse porte sur la modélisation et la résolution de différents problèmes intégrés d'ordonnancement et de transport. Ces problèmes demandent, entre autre, une coordination entre des activités/opérations de production, qui se définissent par une date de début et une durée, et des opérations de transport, qui se définissent par une date de début, une date de fin et une quantité transportée.

Pour résoudre ces problèmes, plusieurs méthodes d'optimisation de type métaheuristique sont proposées, afin d'obtenir des solutions de bonne qualité dans des temps raisonnables. Trois problèmes intégrés sont traités successivement : 1) un problème d'ordonnancement à une machine avec un problème de transport limité à un seul véhicule ; 2) un problème d'ordonnancement à une machine avec un problème de transport à plusieurs véhicules ; 3) un problème d'ordonnancement de type RCPSP avec une flotte hétérogène de véhicules, permettant le transport des ressources entre les activités.

Le premier problème est un problème d'ordonnancement/transport de type PTSP (Production and Transportation Scheduling Problem - PTSP), limité à un seul véhicule, présenté en 2008 par Geismar *et al.*. Une méthode de résolution de type GRASP×ELS est proposée dans le chapitre 2, les résultats obtenus avec cette méthode sont comparés aux meilleurs résultats de la littérature. Cette méthode est étendue dans le chapitre 3, afin de traiter du problème de PTPSP, avec une flotte homogène de véhicules. La méthode proposée possède un champ d'application plus large que la méthode de Geimar *et al.*, dédiée au PTSP avec un véhicule, mais permet de résoudre efficacement le cas à un véhicule.

Le dernier problème traité concerne la résolution d'un RCPSP, dans lequel une flotte de véhicules assure le transport d'une ressource d'une activité à l'autre. L'objectif est d'offrir une approche tirant profit de décisions stratégiques (organiser des échanges – flot – entre des sites), pour déterminer un plan de transport. La difficulté principale consiste à utiliser le flot, pour déterminer les opérations de transport (création de lots), afin de résoudre le problème d'affectation des véhicules, pour finalement ordonnancer les opérations de transport. Sur ce problème, une méthode heuristique de transformation est présentée dans le chapitre 4, ainsi qu'une méthode exacte (basée sur un algorithme de plus court chemin à contraintes de ressources) dans le chapitre 5.

Mots-clés : Recherche opérationnelle, ordonnancement, transport, heuristiques, métaheuristiques

ABSTRACT

This dissertation focuses on modelling and resolution of integrated scheduling and routing problems. Efficient resolutions of these problems require a proper coordination of activities/production operations, defined by starting and finishing times, and of transportation operations, fully defined by starting times, finishing times and quantities of resources transferred.

The resolution of this problem is based on several metaheuristics, with the aim to obtain high quality solutions in acceptable computational time. Three problems are iteratively studied considering: 1) a single machine scheduling problem and a transportation problem with a single vehicle; 2) a single machine scheduling problem with a homogeneous fleet of vehicles for the transport; 3) a RCPSP where the flow transferred between activities is transported by a heterogeneous fleet of vehicles.

The first problem addressed is the PTSP (Production and Transportation Scheduling Problem - PTSP) where the routing part is devoted to a single vehicle (Geismar *et al.*, 2008). The chapter 2 focuses on a GRASP×ELS method benchmarked with the best published methods. This method is extended to the PTSP with multiple vehicles in the chapter 3, and the method shows its capacity to address a wide range of problem, since the PTSP with a single vehicle is a special case.

The second problem deals with the RCPSP, where a heterogeneous fleet of vehicles is devoted to the transportation of resources between activities. The objective consists in considering a flow (activity exchanges solved at a strategic level), to compute a transportation plan. The main difficulties consists in using the flow to compute transportation batches. A heuristic-based approach is introduced in the chapter 4 and an exact method is provided in the chapter 5.

Key-words: Operational research, scheduling, routing, heuristics, metaheuristics

TABLE DES MATIERES

INTRODUCTION GENERALE

Les problèmes d'ordonnancement et de transport jouent un rôle très important dans la gestion de la chaîne logistique. Pour optimiser globalement les processus de décision dans la chaîne logistique, une résolution intégrée de ces deux problèmes doit être effectuée. On peut considérer que, résoudre efficacement ce type de problème, consiste à planifier dans le temps et dans l'espace, des opérations sur des ressources qui sont le plus souvent des opérations de production, et des opérations de transport. Ce type de problèmes se rencontre dans le cadre de la production de marchandises qui doivent être acheminées depuis un ou plusieurs lieux de production vers des centres d'approvisionnement (on parle de « hubs ») ou de clients finaux.

Ces problèmes sont des problèmes d'optimisation combinatoire, c'est-à-dire des problèmes pour lesquels, il existe un grand nombre de solutions, et dont l'énumération est généralement impossible, sauf pour de petites instances. Dans cette thèse, des combinaisons différentes de problèmes d'ordonnancement et de transport sont étudiées, chacune présentant des contraintes spécifiques à prendre en compte.

Cette thèse se compose de 5 chapitres.

Le *premier chapitre* présente le contexte général de cette thèse et donne des définitions sur les problèmes d'ordonnancement et de transport, avec une présentation des modèles théoriques associés. Des notions de bases en complexité et sur les différentes méthodes d'optimisation sont présentées. Il introduit également les principes de base des métaheuristiques, et situe dans un contexte historique, différentes approches possibles en termes de représentation des solutions. Il se base en particulier sur les travaux de (Cheng *et al.*, 1996) pour rappeler les notions de fonction de codage/décodage et la notion d'espace de recherche.

Le *second chapitre* s'intéresse à un problème de type PTSP (Production and Transportation Scheduling Problem) défini initialement par (Geismar *et al.*, 2008). Dans ce problème, la production se modélise sous la forme d'un problème à une machine, intégrant un problème de transport avec un seul véhicule. La résolution conjointe de ce problème est motivée par le fait que les produits fabriqués et transportés sont périssables. Cette contrainte se traduit par une durée maximale à respecter, entre la fin de la production et la livraison des clients : elle se modélise sous la forme d'un time-lags maximal entre l'opération qui modélise la production du produit, et l'opération qui modélise sa livraison à un client. Dans ce chapitre sont présentés les différents points-clés d'une méthode de résolution de type GRASP×ELS, qui se base sur une approche par représentation indirecte des solutions avec des tours géants. Parmi les procédures nécessaires à la définition de cette méthode, une attention particulière est portée à la fonction d'évaluation, qui utilise une méthode SPLIT pour créer des tournées. À partir des informations données

par la méthode SPLIT, le problème du PTSP peut être modélisé comme un flow-shop à deux machines avec des time-lags maximaux. Deux approches de résolution permettant la résolution de ce problème d'ordonnancement sont alors étudiées. Les méthodes de résolution proposées sont comparées à deux autres méthodes de la littérature, dont la méthode de (Geismar *et al.*, 2008) et celle de (Karaoğlan et Kensen, 2017), sur les instances proposées par (Geismar *et al.*, 2008). Les résultats prouvent que les méthodes proposées sont performantes par rapport aux méthodes déjà publiées.

Le *troisième chapitre* s'intéresse au problème du PTSP (Production and Transportation Scheduling Problem) étendu à une flotte homogène de véhicules. Cette extension du PTSP requiert la définition d'un nouvel algorithme SPLIT. À partir des informations données par la méthode SPLIT, le problème du PTSP peut être modélisé comme un flow shop hybride à deux étages avec des time-lags maximaux. Une nouvelle méthode permettant la résolution de ce problème est proposée ainsi qu'une recherche locale. Ces deux éléments reposent sur la définition d'un graphe disjonctif. Ce problème étant une extension du PTSP, introduit par (Geismar *et al.*, 2008), de nouveaux jeux de données sont créés, et couvrent maintenant une large gamme de configurations (plusieurs zones de clients, des positions du dépôt très variées et non nécessairement au « centre » de la zone client, etc.). La méthode de résolution proposée est aussi testée, sur le cas particulier à un véhicule, sur les instances de (Geismar *et al.*, 2008). Les résultats prouvent que la méthode proposée est plus performante que les méthodes dédiées au cas à un véhicule.

Le *quatrième chapitre* aborde la résolution d'un problème, noté RCPSPR pour RCPSP (Resource-Constrained Scheduling Problem) avec routage (with Routing), qui est défini comme un RCPSP mono-ressource, dans lequel le déplacement des ressources entre les activités doit être effectué par une flotte hétérogène de véhicules. Ce problème se différencie des problèmes d'ordonnancement et de transport « classiques » (Job-shop avec Transport, etc.), par le fait que les opérations de transport ne sont pas définies par les données du problème d'ordonnancement, mais doivent être définies par la méthode de résolution. Une contribution de ce chapitre repose sur la définition d'une méthode d'évaluation, permettant d'obtenir à partir d'un vecteur d'activités, à la fois, une solution du RCPSP, mais aussi une solution à un problème de type VRPPD (Vehicle Routing Problem with Pickup and Delivery). Pour cela, la méthode associe à chaque vecteur d'activités, un flot qui permet de définir des lots d'opérations de transport qui sont utilisés dans une approche de type SPLIT, pour ensuite définir les opérations de transport affectées aux véhicules. L'algorithme SPLIT proposé, rentre dans la catégorie des algorithmes de plus court chemin à contraintes de ressources, et gère des labels modélisant à la fois les activités et les véhicules. De nouvelles instances avec un nombre important de véhicules sont proposées et servent à valider l'approche.

Le *cinquième chapitre* présente une méthode exacte de programmation dynamique, permettant la résolution du RCPSPR à partir d'un flot. Le principe de la méthode est basé sur le calcul d'un plus court chemin à contraintes de ressources, dans un graphe spécifique, avec l'utilisation de labels. Les labels sont gérés grâce à une règle de propagation, permettant de prendre en compte simultanément, les contraintes du problème d'ordonnancement et celles du problème de routage. Des règles de dominance limitent le nombre de labels sur les nœuds du graphe. L'approche s'inscrit dans la lignée des approches de (Eiselt *et al.* 1995) et plus récemment de celle introduite par (Corberan

et al., 2017). Cette méthode s'inscrit naturellement dans l'idée de développer un module de post-optimisation pour la méthode proposée dans le chapitre 4.

La *conclusion* reprend les apports de la thèse et dégage les perspectives de recherche.

Dans cette thèse, les problèmes sont identifiés par leur nom anglais, et lorsqu'il existe un équivalent français, celui-ci est rappelé. Cependant, les abréviations correspondent aux termes anglais, car il s'agit des acronymes les plus utilisés et les plus connus. Comme l'usage en recherche est de privilégier les expressions anglo-saxonnes, le parti pris a été de les conserver et on parle ainsi de makespan, ou de time-lags par exemple. Notons de plus, que le terme « borne inférieure » dans cette thèse est la traduction littérale de « lower bound » et est utilisée dans le sens anglo-saxon.

Liste des publications effectuées pendant la thèse

Revues publiées

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. A New Shortest Path Algorithm to Solve the Resource-Constrained Project Scheduling Problem with Routing from a Flow Solution. Engineering Applications of Artificial Intelligence. Volume 66, Issue C, Pages 75-86, **2017**.

Revues soumises

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. Integration of Routing into Resource Constrained Project Scheduling Problem. EURO Journal on Computational Optimization.

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. The Integrated Production and Transportation Scheduling Problem with Multiple Vehicles for a Single Product. International Journal of Production Research.

« Keynote presentation »

Philippe Lacomme, Matthieu Gondran et Marina Vinot. At the crossroads of scheduling problems and routing problems. 18th Free workshop on metaheuristics for a better world (EU/ME), Rome, Italy, April 3-4, **2017**.

Conférences Internationales

Philippe Lacomme, Aziz Moukrim, Alain Quilliot, Nikolay Tchernev et Marina Vinot. A Linear Program for the Resource-Constrained Project Scheduling Problem with Routing. International Conference on Industrial Engineering and Systems Management (IESM), Saarbrücken, Germany, October 11-13, **2017**.

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. Optimal resolution of the transport problem from a flow into a RCPSP with routing. The sixth meeting of the EURO Working Group on Vehicle Routing and Logistics optimization (VeRoLog), Amsterdam, Netherlands, July 10-12, **2017**.

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. The Integrated Production and Transportation Scheduling Problem based on a GRASPxELS resolution scheme. 8th IFAC Conference on Manufacturing Modelling, Management and Control (MIM), Troyes, France, June 28-30 2016, IFAC-PapersOnLine, Volume 49, Issue 12, Pages 1466-1471, **2016**.

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. Scheduling resource-constrained projects with transportation constraints. The fifth meeting of the EURO Working Group on Vehicle Routing and Logistics optimization (VeRoLog), Nantes, France, June 6-8, **2016**.

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. A GRASPxELS Approach for the Resolution of the Integrated Production and Transportation scheduling Problem, 45th International Conference on Computers & Industrial Engineering (CIE45), Metz, France, October 28-30, **2015**. Best paper third place award.

Philippe Lacomme, Aziz Moukrim, Alain Quilliot, Daniele Vigo et Marina Vinot. Simultaneously Handling Routing and Scheduling Through a GRASPxELS Algorithm, 27th European Conference on Operational Research (EURO), Glasgow, United Kingdom, July 12-15, **2015**.

Conférences Françaises

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. Formalisation linéaire d'un problème de RCPSP avec transport de ressources, 18ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Metz, France, Février 22-24, **2017**.

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. Résolution conjointe du problème d'ordonnancement et de transport des ressources dans un RCPSP avec une flotte hétérogène de véhicule, 18ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Metz, France, Février 22-24, **2017**.

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. Résolution conjointe des problèmes de production et de transport avec plusieurs véhicules, 17ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Compiègne, France, Février 10-12, **2016**. Finaliste pour le « Prix Jeune Chercheur »

Autres Présentations

Philippe Lacomme, Aziz Moukrim, Alain Quilliot et Marina Vinot. Résolution exacte du problème d'ordonnancement de projets sous contraintes de ressources avec transport à partir d'un flot de ressources. Exposé thématique réalisé à l'Ecole d'Eté du GT2L « les nouveaux enjeux de la logistique et du transport ». Gardanne, France, Mai 23-24, **2017**.

CHAPITRE 1

Contexte de l'étude

Ce chapitre introduit la problématique de la thèse, avec la présentation des problèmes intégrés d'ordonnancement et de transport. L'optimisation de la production et du transport est un enjeu majeur dans la gestion de la chaîne logistique, qui a pour but d'optimiser les échanges, ou flux, que l'entreprise entretient avec ses fournisseurs et ses clients. Cependant, malgré l'interdépendance de ces deux étapes, celles-ci sont souvent traitées séquentiellement.

Ce chapitre a pour but de rappeler les définitions principales et le vocabulaire utilisé, dans le domaine de l'ordonnancement et du transport et non de donner une liste exhaustive des schémas algorithmiques existants ou des travaux précédents. À la fois pour les problèmes d'ordonnancement et de transport, les notions d'espace de codage et de fonction d'évaluation sont situées dans le cadre général de l'optimisation.

Deux exemples de codage sont donnés, le premier pour un problème d'ordonnancement de type Job Shop et le deuxième pour un problème de type VRP. Les notions générales sur la théorie de la complexité ainsi que différentes méthodes de résolution sont présentées d'un point de vue formel.

1.1 Introduction

Ce chapitre est consacré à la présentation de la problématique : l'optimisation conjointe des problèmes d'ordonnancement et de transport. La coordination de ces deux problèmes constitue un défi majeur au sein des chaînes logistiques. Dans la société actuelle, la croissance des activités de production, de distribution et de collecte de produits a créé une pression sur de nombreuses entreprises et collectivités. Ces dernières doivent résoudre des problèmes d'optimisation de grande taille liés aux secteurs de la production et du transport.

Ces problèmes ont fait apparaître de nouveaux besoins, soit dans l'étude de ces problèmes pris séparément, soit dans la nécessité de traiter ces problèmes de manière intégrée. En termes de production, les problèmes à résoudre relèvent de manière générale des problématiques de type « ordonnancement » et pour la distribution des problématiques de type « transport » pour reprendre les termes scientifiques usuels.

On distingue plusieurs familles de problèmes d'ordonnancement selon, par exemple, le type d'opérations à ordonnancer, l'utilisation de machines ou l'existence de contraintes sur les ressources. De la même façon, pour les problèmes liés au transport et aux tournées de véhicules plusieurs catégories de problèmes peuvent être rencontrées, suivant l'emplacement des tâches à effectuer. Habituellement, on distingue les cas où les demandes sont situées le long des arcs (on parle alors de problèmes de type « arc routing ») et les cas où les demandes sont localisées sur des nœuds du réseau (on parle

alors de problèmes de type « node routing »). Un grand nombre de problèmes de transport sont difficiles à résoudre, et il n'existe pas de méthode mathématique efficace permettant de le faire dans des temps de calcul acceptables.

L'étude des problèmes intégrés de production et de transport est relativement récente et suscite de plus en plus d'intérêt. Cependant la résolution efficace de ce type de problèmes reste difficile puisque combinant deux problèmes déjà difficiles à résoudre indépendamment. Il existe néanmoins des méthodes pour résoudre ces problèmes de manière exacte ou approchée.

1.2 La chaîne logistique ou supply chain

Traditionnellement, dans une entreprise, la planification, la production, les achats et la distribution opèrent de manière indépendante. Dans une telle organisation chaque service a ses propres objectifs, et de plus, les objectifs des différents services ne sont pas nécessairement en adéquation. Souvent, l'activité de production est organisée pour maximiser la productivité, tout en minimisant le coût de revient, sans se soucier de l'impact sur le volume de stock et la capacité de la distribution. La conséquence de ces observations est qu'il n'y a pas un plan unique et intégré pour l'ensemble de l'organisation. Il pourrait cependant être bénéfique de disposer d'un mécanisme qui permettrait à ces fonctions de se coordonner, et de travailler ensemble, pour optimiser le processus dans sa globalité.

1.2.1 Définition

Une chaîne logistique peut être définie par un processus intégré au sein duquel différentes entités (fournisseurs, fabricants, distributeurs et détaillants) travaillent ensemble afin d'acquérir, de transformer et de livrer des produits aux détaillants et aux clients finaux.

Afin de mieux comprendre ce concept relativement récent de Chaîne Logistique CL (Supply Chain SC), plusieurs définitions de ce terme ont été données et sont utilisées dans la littérature. (Christopher, 1992) définit la chaîne logistique comme étant «le réseau d'entreprises qui participent, en amont et en aval, aux différents processus et activités qui créent de la valeur sous forme de produits et de services apportés au consommateur final. En d'autres termes, une chaîne logistique est composée de plusieurs entreprises, en amont (fourniture de matières et composants) et en aval (distribution), et du client final ». (Lummus *et al.*, 1998) définissent la chaîne logistique comme étant « le réseau d'entités par lequel le flux matériel passe. Ces entités incluent fournisseurs, transporteurs, sites d'assemblages, centres de distribution, détaillants et clients ». Une définition plus générale est celle proposée par (Poirier et Reiter, 2001) : « Une chaîne logistique est le système grâce auquel les entreprises amènent leurs produits et leurs services jusqu'à leurs clients ».

1.2.2 La prise de décision en logistique : un processus hiérarchisé

Trois niveaux de la prise de décision dans une chaîne logistique ont été identifiés selon l'horizon de temps impliqué (Ballou, 1992) : le niveau stratégique, le niveau tactique et le niveau opérationnel. Le niveau stratégique prend en compte un horizon de temps à long

terme. À l'opposé, les décisions opérationnelles concernent des décisions à court terme. Le niveau tactique se situe entre les deux, on parle de moyen terme (Figure 1-1).

Niveau stratégique :

Les décisions stratégiques concernent typiquement l'ouverture, la fermeture ou la localisation de sites de production et d'entrepôts. On souhaite par exemple déterminer l'emplacement des nouveaux entrepôts ou leurs capacités. Un autre exemple de décision stratégique concerne le système de distribution. La compagnie doit-elle posséder ses propres camions ou plutôt sous-traiter l'activité de transport ? Certaines décisions stratégiques peuvent également être affectées au niveau tactique suivant la flexibilité de la production ou de la distribution. Par exemple, lorsque les activités de transport et d'entreposage sont sous-traitées, la flexibilité de la distribution augmente et il est possible d'augmenter rapidement l'espace utilisé pour l'entreposage ou le nombre de camions utilisés.

Niveau tactique :
Les décisions au niveau tactique concernent principalement le contrôle des processus opérationnels : le stockage, la fabrication et le transport. D'un côté, le rôle de la gestion du stock est de répondre aux questions suivantes : quoi, quand et combien faut-il approvisionner. D'un autre côté, le contrôle de la production s'intéresse surtout à l'ordonnancement des opérations dans un atelier, à la définition des priorités et au suivi du programme de fabrication. Les décisions tactiques portent sur les problèmes liés à la gestion des ressources de l'entreprise, en particulier la planification des activités sur ces ressources.

Niveau opérationnel :

Les décisions opérationnelles sont liées elles aussi aux activités suivantes : stockage, production et transport. Un exemple de décision opérationnelle sur la gestion d'un stock est de commander une quantité supplémentaire pour faire face à une augmentation de la demande dans un futur très proche. Une décision opérationnelle dans un système de transport est le choix de la route à suivre, ou de l'ordre de passage chez les différents clients.

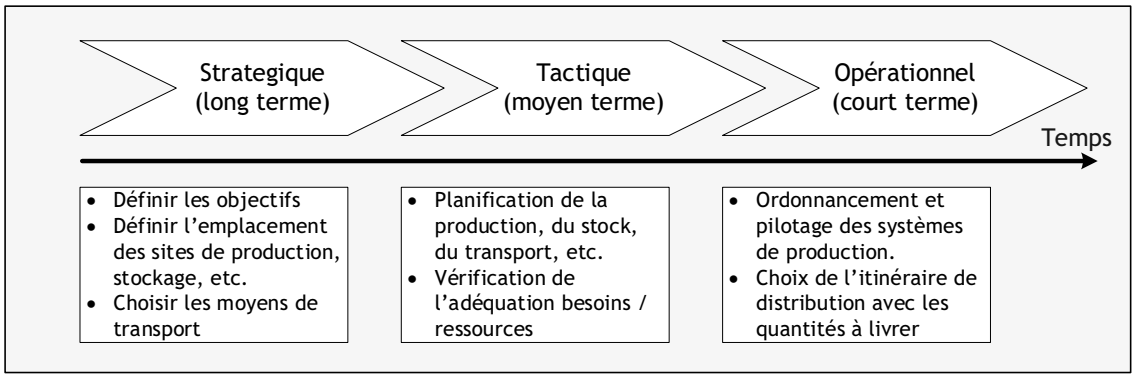


Figure 1-1. Les différents niveaux de décision dans la chaîne logistique

À cause de la complexité des problèmes au sein de la chaîne logistique, rares sont les méthodes qui traitent en même temps des décisions stratégiques, tactiques et opérationnelles ainsi que des risques liés à la dynamique de la chaîne logistique

(fluctuations dans les commandes clients, ainsi que dans les délais de fabrication et de transport).

1.2.3 Les problématiques d'ordonnancement de la production et du transport

Récemment, un intérêt accru a émergé pour la conception, l'évaluation des performances et l'optimisation d'une chaîne logistique dans sa globalité. Des travaux apparaissent sur l'interaction de plusieurs catégories de décisions rencontrées dans la configuration d'une chaîne logistique. On remarque notamment que les problèmes de production et de transport interviennent à la fois au niveau tactique et au niveau opérationnel de la chaîne logistique. Un état de l'art sur ces problèmes est donné par (Moons *et al*., 2017). Ils proposent un schéma général permettant de classer les problèmes intégrés déjà étudiés, en fonction du type de problèmes d'ordonnancement étudié sur lequel est intégré un problème de tournées de véhicules.

Les différents problèmes d'ordonnancement et de tournées de véhicules (que l'on retrouve dans les problèmes intégrés) peuvent être étudiés indépendamment à l'aide de modèles théoriques sur lesquels des contraintes spécifiques peuvent être ajoutées.

1.3 Problèmes théoriques

1.3.1 Introduction

Les systèmes de production et de distribution peuvent être très variés. Pour étudier ces systèmes, il est nécessaire de pouvoir les modéliser. Des modèles très généraux ont été proposés dans le but de s'adapter à la grande variété des problèmes d'ordonnancement et de tournées de véhicules.

Les principaux modèles utilisés pour modéliser les problèmes d'ordonnancement sont détaillés dans la partie 1.3.2, et dans la partie 1.3.3 pour les problèmes de tournées de véhicules.

1.3.2 Problèmes classiques d'ordonnancement

D'après la définition générale des problèmes d'ordonnancement proposée par (Roy, 1970), on parle de problème d'ordonnancement, lorsqu'un ensemble de travaux est à réaliser, que cette réalisation est décomposable en tâches, et que le problème consiste à définir la localisation temporelle des tâches et/ou la manière de leur affecter les moyens nécessaires.

Résoudre un problème d'ordonnancement revient donc à répondre aux questions « quand » et « avec quels moyens » exécuter chacune des tâches. De manière synthétique, on peut dire que l'on a un problème d'ordonnancement dès que l'on doit planifier un ensemble de tâches. On se rend compte que les domaines d'application de l'ordonnancement sont immenses. Globalement, on distingue deux grands groupes de problèmes d'ordonnancement en fonction du mode d'utilisation des ressources mises en jeu dans la réalisation du projet :

- les problèmes d'ordonnancement cumulatif ;
- les problèmes d'ordonnancement disjonctif.

Dans les problèmes d'ordonnancement cumulatif (CuSP), les ressources peuvent exécuter plusieurs activités simultanément. Ce faisant, dans le cadre de ressources disponibles en quantité limitée, celles-ci sont utilisées dans la limite de la quantité disponible. Dans les problèmes d'ordonnancement disjonctif, une ressource exécute exclusivement une et une seule activité à la fois.

On peut distinguer alors deux grandes familles de problèmes d'ordonnancement :
- les problèmes d'ordonnancement de projet (cumulatif),
- les problèmes d'ordonnancement en atelier (disjonctif).

La terminologie utilisée dans ces problèmes se rapproche des termes employés dans les systèmes industriels de production (gamme, machine, job, opération, activité, ressource, *etc*.).

1.3.2.1 Les problèmes d'ordonnancement de projet

Les problèmes d'ordonnancement de projet sont constitués d'un ensemble d'activités/jobs à ordonnancer et d'un ensemble de ressources. Il peut y avoir différents types de ressources, chaque type de ressources étant disponible en quantité limitée. Une activité consomme une certaine quantité de ressources appartenant à un ou plusieurs types.

Le problème d'ordonnancement de projet classique est le RCPSP (Resource-Constrained Project Scheduling Problem) dans lequel toutes les ressources sont renouvelables et disponibles en quantité fixée. Le RCPSP est un problème d'ordonnancement cumulatif qui a été très étudié, avec de nombreuses applications industrielles qui lui sont liées. Les premières publications qui introduisent explicitement la terminologie RCPSP remontent à la fin des années 1960 avec (Prisker *et al.*, 1969). Depuis, plusieurs méthodes exactes (Demeulemeester et Herroelen, 1992), (Mingozzi *et al.*, 1998), (Sprecher, 2000) et de nombreuses méthodes heuristiques (Kolisch, 1996), (Kolisch et Hartmann, 2006) ont été développées pour le RCPSP.

Il existe de nombreuses variantes et extensions de ce problème et (Hartmann et Briskorn, 2010) dressent une classification des variantes en cinq classes, portant sur :
- les contraintes liées aux activités (préemptivité, consommation de ressources variable en fonction du temps, multi-modes, *etc*.) ;
- les contraintes temporelles (time lags, contraintes sur les dates de début et fin des activités, *etc*.) ;
- les contraintes de ressources (disponibilité des ressources variable en fonction du temps, ressources non renouvelables, .*etc*.) ;
- la fonction objectif (optimiser la robustesse du planning, minimiser les coûts dans le cas de ressources payantes, *etc*.) ;
- la planification de plusieurs projets en même temps.

Les variantes les plus étudiées concernant les ressources sont :
- le RCPSP avec ressources non renouvelables : il peut exister des ressources non renouvelables (ressources qui ne peuvent être utilisées qu'une seule fois) ;
- le Multi-mode RCPSP (MRCPSP) : une activité peut être traitée de différentes manières appelées « modes ». La consommation de ressources et la durée de l'activité

sont différentes dans chaque mode. Le traitement dans certains modes peut nécessiter la consommation de ressources non renouvelables. La préemption est interdite et une fois le traitement commencé dans un mode, il n'est plus possible de changer de mode ;
- le RCPSP avec ressources à disponibilité variable : la disponibilité des ressources varie en fonction du temps.

1.3.2.2 Les problèmes d'ordonnancement en atelier

Les problèmes d'ordonnancement d'atelier sont constitués d'un ensemble d'opérations à ordonnancer et d'un ensemble de machines/ressources qui sont disjonctives et renouvelables. Les opérations doivent être effectuées sur une ou plusieurs machines pour effectuer la fabrication/transformation d'une ou plusieurs pièces nommées généralement « job ». Dans ces problèmes, les durées des opérations sont en général connues, elles font partie des données des problèmes. Un job se caractérise par un ensemble d'opérations qui doivent être effectuées sur une machine. Dans l'immense majorité des cas, le but est de trouver un ordonnancement des opérations qui minimise le makespan, *i.e.* le temps écoulé entre le début de la première opération et la fin de la dernière opération sur la dernière machine. Dans les problèmes d'ordonnancement en atelier, une machine ne peut traiter qu'une opération à la fois.

Les problèmes d'ordonnancement d'atelier se distinguent essentiellement par leur gamme opératoire qui définit l'ordre des opérations à réaliser pour chaque job. Trois grands problèmes d'ordonnancement d'atelier peuvent donc être définis suivant les caractéristiques de la gamme opératoire :
- le problème du flow shop (Flow Shop Scheduling Problem - FSSP) : tous les jobs ont la même gamme opératoire. Chaque pièce passe par toutes les machines dans l'ordre défini par la gamme opératoire ;
- le problème du job shop (Job Shop Scheduling Problem - JSSP) : la gamme opératoire varie d'un job à l'autre, l'ordre des opérations pour chaque job est donné. Chaque pièce passe par toutes les machines dans l'ordre défini par sa gamme opératoire ;
- le problème de l'open shop (Open Shop Scheduling Problem - OSSP) : la gamme opératoire varie d'un job à l'autre et l'ordre des opérations pour un job n'est pas connu. Dans ce problème, on doit en plus, déterminer l'ordre des opérations de chaque job.

Il existe également deux problèmes, plus spécifiques :
- le problème à une machine : la gamme opératoire contient une unique opération à réaliser sur une unique machine. Ce problème peut contenir des contraintes additionnelles (typiquement des délais qui dépendent de la séquence) qui le rendent non trivial :
- le problème à machines parallèles : la gamme opératoire contient une unique opération à réaliser sur une machine parmi un ensemble de machines identiques dites parallèles.

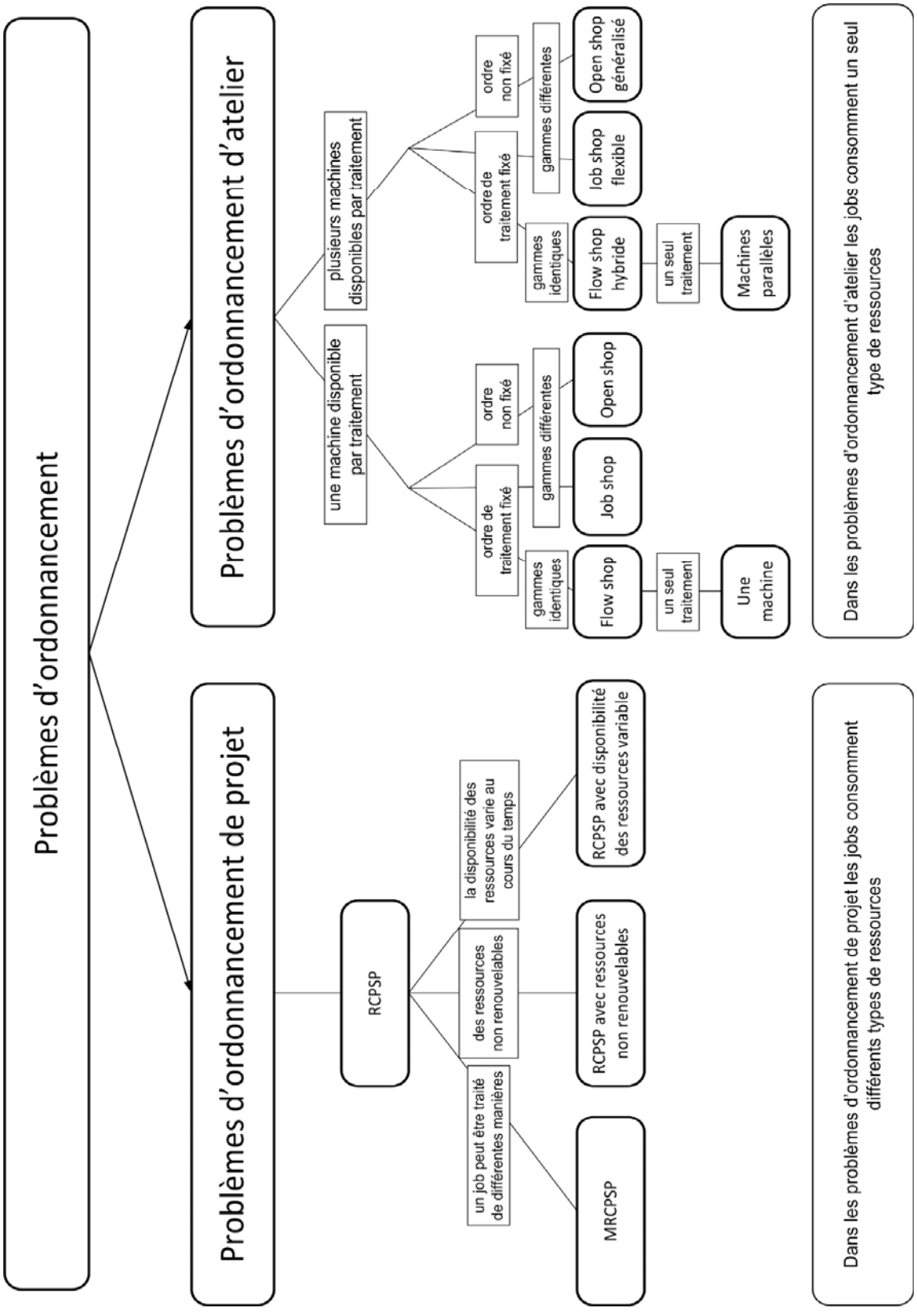


Figure 1-2. Les problèmes d'ordonnancement classiques selon (Toussaint 2010).

Dans les problèmes de flow shop, de job shop et d'open shop, une opération donnée ne peut être réalisée que par une machine. Il existe des formes étendues de ces problèmes dans lesquelles plusieurs machines peuvent réaliser la même opération donnant alors lieu à un problème d'affectation en plus du problème d'ordonnancement. Ces problèmes se nomment flow shop hybride, job shop flexible et open shop généralisé.

La Figure 1-2, extraite de la thèse d'Hélène Toussaint (Toussaint, 2010), présente une synthèse des différents problèmes classiques d'ordonnancement en faisant apparaître les deux classes de problèmes précédemment citées.

1.3.3 Problèmes classiques de tournées de véhicules

La collecte ou la distribution (de marchandises ou de personnes) constitue en particulier l'un des maillons de la chaîne logistique les plus étudiés vu les enjeux environnementaux et économiques associés. Ces aspects sont devenus encore plus cruciaux aujourd'hui, par la nécessité du développement durable admise par le plus grand nombre. Cela s'est traduit par une forte activité de recherche dans le domaine du transport ces dernières années. Les chercheurs s'emploient à concevoir des modèles et approches de résolution de mieux en mieux adaptés aux problèmes réels. L'ensemble des problèmes d'optimisation correspondants est regroupé dans la littérature sous l'expression générique de problèmes de tournées (« routing problems »). Cette appellation unique recouvre cependant des problèmes qui peuvent présenter des caractéristiques très différentes.

Les problèmes de tournées de véhicules consistent à déterminer, pour un ensemble donné de clients, par quels véhicules de la flotte ils sont visités et à quel moment dans la tournée. Deux grandes classes émergent en fonction de la position des clients. S'ils se trouvent sur des points précis d'un réseau, le problème étudié est un problème de tournées sur nœuds (partie 1.3.3.1). En revanche, si les clients sont localisés sur des liens du réseau, il s'agit d'un problème de tournées sur arcs. La deuxième classe apparaît quand les clients se trouvent distribués le long des rues ou lorsque la rue elle-même nécessite un service (partie 1.3.3.2). La Figure 1-3 donne une représentation synthétique des problèmes présentés, elle est située à la fin de cette partie.

1.3.3.1 Les problèmes de tournées sur nœuds

Le problème du voyageur de commerce (Traveling Salesman Problem - TSP) (Dantzig *et al.*, 1954) est sans doute le plus simple, le plus connu et le plus étudié des problèmes de tournées sur nœuds (Node Routing Problem – NRP). Il consiste à définir la tournée d'un voyageur de commerce visitant un ensemble de villes et retournant à la ville de départ (généralement nommée dépôt). Une solution définit l'ordre dans lequel les villes seront visitées. Le problème est généralement modélisé sous la forme d'un graphe dans lequel les nœuds représentent les villes à visiter et les arcs représentent les routes liant les villes deux à deux. On travaille de plus, généralement, avec un graphe complet dans lequel les plus courts chemins ont été précalculés. Ainsi, le voyageur de commerce doit visiter une et une seule fois chaque nœud (la tournée correspond à un cycle hamiltonien) dans un graphe non orienté complet et pondéré. L'objectif du TSP est de minimiser la distance totale parcourue.

De nombreux autres problèmes de tournées de véhicules ont été définis par extension de ce problème de base, en rajoutant différentes, contraintes et/ou objectifs. Dans le cadre de cette thèse, seules sont présentées quelques extensions qui concernent, notamment, la présence de nœuds nécessitant une collecte et/ou une livraison.

Un de ces problèmes est le problème de tournées de véhicules (Vehicle Routing Problem - VRP) introduit par (Dantzig et Ramser, 1959). C'est une généralisation du TSP qui peut être décrite comme un problème de conception d'itinéraires (routes) pour plusieurs véhicules, en minimisant le coût, *i.e.* la distance totale parcourue. Chaque route débute au niveau du dépôt puis effectue des livraisons (ou collectes) sur un ensemble de points géographiquement dispersés (villes, magasins, entrepôts, écoles, clients, machines de fabrication dans un atelier, *etc.*), pour finalement retourner au dépôt. Les contraintes qu'une solution doit respecter sont les suivantes :

- un client ne peut être servi que par un et un seul véhicule ;
- chaque véhicule effectue au plus une tournée ;
- tous les clients doivent être traités.

Le VRP est souvent défini avec des contraintes de capacité pour les véhicules ainsi que des contraintes sur la longueur maximale des tournées (autonomie). Lorsque seules les contraintes de capacité sont présentes, le problème est nommé CVRP (Capacitated Vehicle Routing Problem). Dans les problèmes de tournées de véhicules, il est fréquent de supposer que la vitesse des véhicules est unitaire, de sorte que les distances, les temps de déplacement et les coûts de déplacement sont considérés comme équivalents.

Contrairement au VRP classique dans lequel tous les clients ont soit des demandes de collecte, soit des demandes de livraison de produits, dans le problème du VRPB (Vehicle Routing Problem with Backhauls), les deux types de demandes coexistent (Parragh *et al.*, 2008a). Dans cette variante du VRP, pour chaque tournée, les livraisons doivent être effectuées avant les collectes afin d'éviter les manipulations dans les véhicules. Tous les produits à livrer sont chargés au dépôt et tous les produits collectés sont déchargés au dépôt. L'objectif reste toujours la minimisation de la distance totale parcourue.

Le problème de tournées de véhicules avec collecte et livraison (Vehicle Routing Problem with Pickup and Delivery - VRPPD) est presque identique au VRPB à l'exception du fait que les produits livrés peuvent être chargés, soit au dépôt soit chez les clients avec une requête de collecte. Dans le cas où les demandes des clients entre les collectes et les livraisons sont liées, c'est-à-dire qu'il existe un couple origine destination, deux autres problèmes sont définis : le GPDP et le DARP, (Parragh *et al.*, 2008b). Le GPDP (General Pickup and Delivery Problem) porte sur le transport de marchandise alors que le DARP (Dial-A-Ride Problem) porte sur le transport de personnes avec une notion de qualité de service. Il faut noter qu'il existe de très nombreuses variantes aux problèmes de tournées de véhicules et qu'une classification a été proposée dans la thèse de (Ren, 2012).

1.3.3.2 Les problèmes de tournées sur arcs

Les problèmes de tournées sur arcs ont été moins étudiées, comparativement aux problèmes de tournées sur nœuds, bien qu'ils permettent de modéliser un grand nombre d'applications réelles comme la collecte des déchets ménagers, le déneigement des routes, le relevé de compteurs d'électricité ou encore la distribution de courrier.

L'étude des problèmes de tournées sur arcs a commencé en 1735 lorsque Leonhard Euler a présenté sa solution au problème du pont Königsberg (Sachs *et al.*, 1988). Théoriquement, ce problème est maintenant connu sous le nom d'Euler Tour Problem. Étant donné un graphe connecté $G = (N, E)$, avec N les nœuds du graphe et E les arrêtes du graphe, trouver un parcours Eulérien revient à trouver un cycle qui visite chaque arrête dans E exactement une fois. Euler a prouvé qu'un parcours Eulérien existe si et seulement si chaque nœud dans G a un degré pair.

Suite à cela, le problème du postier chinois (Chinese Postman Problem - CPP) a été suggéré d'abord par (Mei-Ko, 1962). Ce problème est formellement décrit comme suit : étant donné un graphe connecté $G = (N, E, C)$, où C est la matrice des distances, trouver un tour qui passe par toutes les arêtes au moins une fois avec un coût total minimal. Lorsque G est complètement orienté ou complètement non orienté, le CPP peut être résolu en temps polynomial, (Edmonds et Johnson, 1973), mais lorsque G est un graphe mixte, le problème devient NP-difficile, (Papadimitriou, 1976).

Introduit par (Orloff, 1974) le problème du postier rural (Rural Postman Problem - RPP), est défini comme suit : étant donné un graphe non orienté $G = (N, E, C)$, où C est la matrice des coûts pour les arêtes, trouver un tour de coût minimum, qui passe chaque arête dans un sous-ensemble $R \subseteq E$ au moins une fois. Le RPP est NP-difficile (Lenstra et Kan, 1976).

Il aura fallu quelques années pour voir apparaître un modèle plus général nommée CARP (Capacitated Arc Routing Problem). Le CARP, d'abord suggéré par (Golden et Wong, 1981), est formellement décrit comme suit : étant donné un graphe non orienté pondéré $G = (N, E, C, Q)$, où C est une matrice de coûts et Q est une matrice de demande et compte tenu d'une flotte de véhicules identiques de capacité W, le CARP consiste à déterminer un ensemble de tournées telles que : 1) chaque arc avec une demande positive est desservie par exactement un véhicule, 2) la somme de la demande de ces arcs desservis par chaque véhicule ne dépasse pas W, et 3) le coût total des visites est minimisé. Le CARP est NP-difficile comme le rappelle (Golden et Wong, 1981).

Le SDCARP (Split Delivery Capacitated Arc Routing Problem) remet en cause une contrainte classique des problèmes de tournée sur arcs. La demande associée à un arc peut être supérieure à la capacité des véhicules et peut donc être répartie sur plusieurs tournées.

Le VRP peut être transformé en CARP (Golden et Wong, 1981), et le CARP peut se transformé en VRP (Pearn *et al.*, 1987), ce qui rend les deux classes de problèmes équivalentes. De manière générale, un NRP peut être transformé en ARP, et un ARP peut être transformé en NRP, au prix toutefois de transformations coûteuses aboutissant à des problèmes de plus grandes tailles.

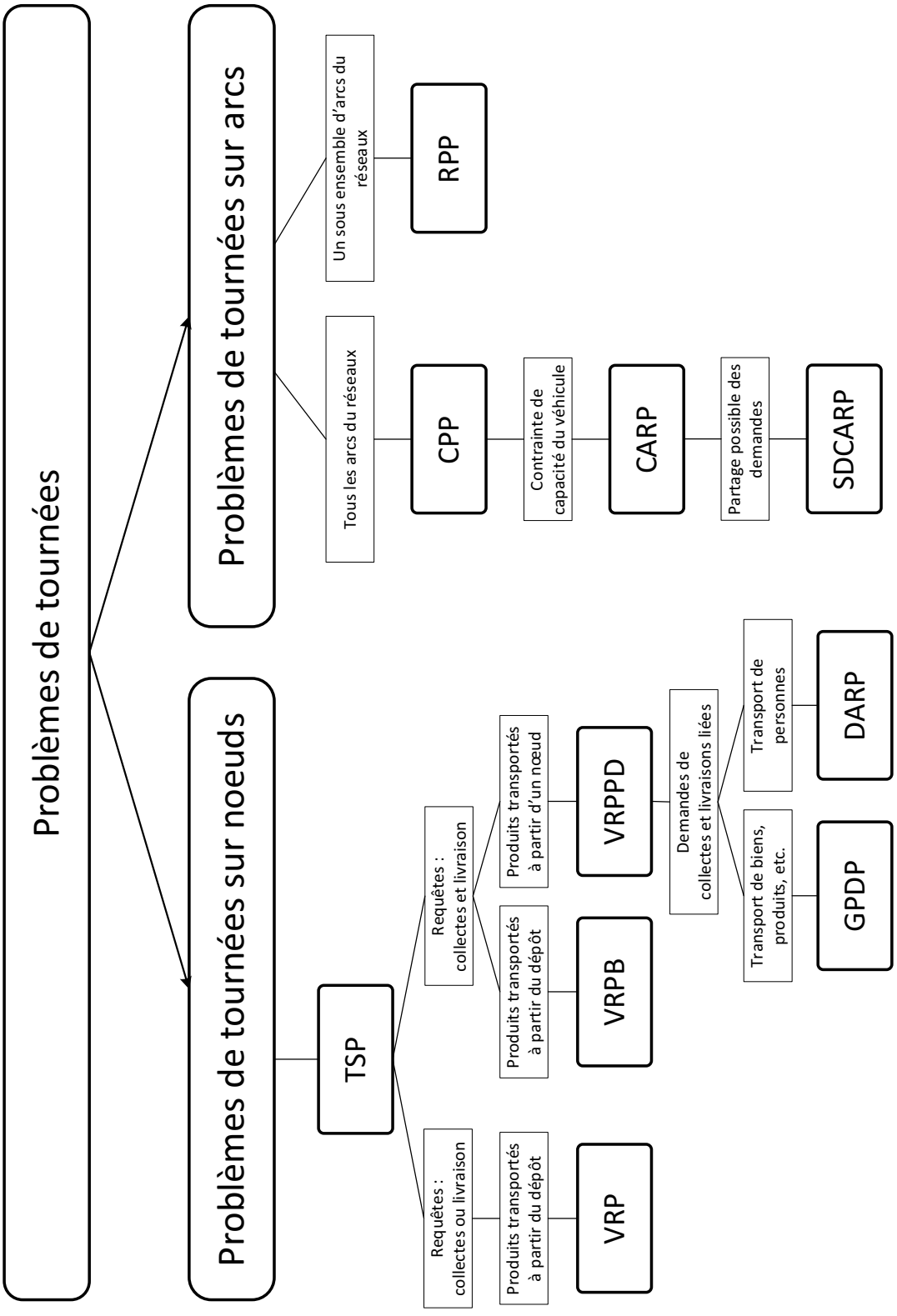


Figure 1-3. Les problèmes de tournées classiques.

1.3.4 Conclusion

Parmi les problèmes d'ordonnancement présentés, deux problèmes font l'objet d'une étude particulière durant cette thèse, le FSSP et le RCPSP. De plus, parmi les problèmes de tournées de véhicules présentés, deux problèmes sont étudiés en détail, le VRP et le VRPPD. Dans la suite, sont présentées les démarches de modélisation permettant de construire un algorithme d'optimisation, soit pour les problèmes d'ordonnancement, soit pour les problèmes de tournées.

1.4 Méthodes de résolution

Les problèmes d'optimisation et les méthodes permettant de les résoudre sont nombreux. Dans cette partie, une synthèse sur les principaux problèmes d'optimisation et les principales méthodes est présentée.

1.4.1 Introduction

Un problème d'optimisation, noté $P(X, f)$, est caractérisé par un ensemble réalisable ou admissible X non-vide et une fonction objectif f qui associe un scalaire dans $\mathbb{R}$ à chaque élément x de l'ensemble X. Les éléments de X sont dits solutions réalisables. Résoudre le problème $P(X, f)$ revient à trouver, parmi les solutions réalisables, une solution qui minimise ou maximise f , c'est-à-dire dans le cas d'un problème de minimisation, trouver une solution $x^* \in X$ telle que $f(x) \geq f(x^*)$ pour tout élément x dans X . Une telle solution est dite optimale et est notée $x(X, f)$.

Les problèmes d'optimisation combinatoire sont des problèmes d'optimisation dont les ensembles réalisables sont finis mais combinatoires. Aussi, le nombre de solutions réalisables des problèmes combinatoires augmente exponentiellement en fonction de la taille du problème, et c'est ce qui exclut des méthodes de résolution basées sur l'énumération de toutes les solutions réalisables.

Un problème d'optimisation combinatoire peut être un problème de minimisation ou un problème de maximisation. Dans la suite, seuls les problèmes de minimisation sont considérés car on peut toujours transformer un problème de maximisation en un problème de minimisation.

Les problèmes d'optimisation peuvent être regroupés en différentes classes selon leur complexité. La complexité d'un problème est définie par rapport au nombre maximal d'instructions élémentaires effectuées durant la résolution de n'importe quelle instance, de taille fixée. La complexité d'un algorithme est exprimée en fonction de n, la taille d'une instance du problème à résoudre. La complexité d'un problème est la complexité du meilleur algorithme qui permet de le résoudre.

La complexité d'un algorithme s'exprime souvent par un ordre de grandeur comme le rappelle (Toussaint, 2010) :

$O(1)$: complexité constante (indépendante de la taille de la donnée)
$O(log(n))$: complexité logarithmique
$O(n)$: complexité linéaire
$O(n^p)$: complexité polynomiale ($p \in \mathbb{N}, p \geq 2$)
$O(a^n)$: complexité exponentielle ($a \in \mathbb{R}, a > 1$)

Un algorithme est dit de complexité $O(p(n))$ s'il se termine après un nombre d'opérations élémentaires ne dépassant pas, au pire cas $\times\, p(n)$, avec c constant et n la taille d'une instance. On dit d'un algorithme de complexité $O(p(n))$ qu'il est efficace (ou polynomial) s'il existe un polynôme en n majorant $p(n)$ (*i.e.* s'il existe deux constantes $c, k \geq 0$ et $n_0 \in N$, telles que $\forall\, n \geq n_0$, $p(n) \leq c \times n^k$).

Pour classifier les algorithmes, il est nécessaire de considérer que l'exécution d'un algorithme se fait sur une machine de Turing, et on distingue deux machines de Turing différentes : la machine de Turing déterministe et celle non déterministe. La machine de Turing déterministe est celle qui modélise les ordinateurs actuels. À chaque étape, pour une machine de Turing déterministe, le choix de la prochaine étape à réaliser est unique. Par opposition, la machine de Turing non déterministe associe à chaque étape, un certain nombre d'étapes suivantes possibles. La machine de Turing non déterministe ne permet pas de choisir quelle est la prochaine étape.

Un problème est dit polynomial (ou polynomial-temps), s'il peut être résolu en temps polynomial par une machine de Turing déterministe. L'ensemble des problèmes polynomiaux forme la classe P.

Un problème appartient à la classe NP, s'il peut être résolu en temps polynomial par une machine de Turing non déterministe.

Les problèmes dans P sont aussi dans NP, cependant, les problèmes dans NP ne sont pas forcément dans P. Aujourd'hui, il est communément admis que $P \neq NP$, mais cette inégalité n'est pour l'instant qu'une conjecture.

La classe NP contient la plupart des problèmes d'optimisation combinatoire, dont les problèmes NP-complets. La sous-classe NP-complet a été définie par (Cook, 1971). Un problème est dit NP-complet s'il est dans NP et si tout problème de NP peut-être polynomialement réduit en ce problème. Les problèmes NP-complets sont donc liés par une relation d'équivalence dans le sens où s'il existe un algorithme polynomial pour un des problèmes de cette sous-classe, alors tous les problèmes NP-complets peuvent être résolus en un temps polynomial et, par conséquent, tout problème de NP peut être résolu en temps polynomial, ce qui signifie donc, dans ce cas $P = NP$.

Actuellement, on ne dispose d'aucun algorithme polynomial permettant de résoudre un problème NP-complet et on ne sait ni s'il en existe un, ni s'il n'en existe pas.

Cette classe englobe les problèmes les plus étudiés de l'optimisation combinatoire, parmi lesquels les problèmes abordés dans cette thèse. Dans la suite de ce chapitre, plusieurs méthodes d'optimisation sont présentées, dont les méthodes exactes et les heuristiques.

1.4.2 Familles de méthodes d'optimisation

Pour résoudre les problèmes d'optimisation, différents types d'approches existent. Plusieurs méthodes d'optimisation sont présentées dans cette partie :

- programmation linéaire ;
- programmation dynamique ;
- algorithmes constructifs ;
- méthodes de recherche locale.

Programmation linéaire

On peut distinguer trois types de programmes linéaires : les programmes linéaires simples, les programmes linéaires en nombres entiers, et les programmes linéaires mixtes. Tous ces programmes sont dits linéaires, ils sont donc composés de contraintes sous forme d'équations ou d'inéquations faisant intervenir les variables du problème.

Les programmes linéaires simples ont des variables dites continues et ces programmes linéaires peuvent être efficacement résolus par la méthode du simplexe ou à l'aide d'une méthode de points intérieurs. Bien que la méthode du simplexe soit exponentielle dans le pire des cas, et que la méthode des points intérieurs soit polynomiale, en pratique, la méthode du simplexe est plus performante et c'est la méthode la plus répandue.

Les programmes linéaires en nombres entiers font intervenir des variables entières. Ces programmes peuvent sembler plus simples que les précédents dans la mesure où le nombre de solutions possibles est fini. Néanmoins, ce type de problème est plus difficile à résoudre.

Les programmes linéaires mixtes contiennent à la fois des variables continues et des variables en nombres entiers. D'un point de vue résolution, ces programmes linéaires ressemblent fortement aux programmes linéaires en nombre entiers.

De nombreux solveurs sont disponibles dans le commerce, et la résolution d'un programme linéaire peut être faite de manière relativement efficace.

Programmation dynamique

Ces méthodes tirent parti des propriétés de décomposition du problème. Ce type de méthodes peut être utilisé sur un problème P de taille n lorsque la solution de ce problème peut être construite à partir de la solution d'un sous-problème de taille inférieure. Une des méthodes de programmation dynamique les plus connues est l'algorithme de Bellman pour le calcul des chemins de longueurs optimales dans un graphe.

Algorithmes constructifs

Les algorithmes constructifs procèdent par construction progressive de la solution. À chaque étape de l'algorithme une nouvelle partie de la solution est construite jusqu'à l'obtention de la solution finale. Ces algorithmes sont à la base de nombreuses heuristiques. Parmi les algorithmes constructifs les plus connues, on peut citer la famille des algorithmes gloutons. Cependant, certains algorithmes constructifs sont optimaux.

Méthodes de recherche locale

Ces méthodes consistent à définir, autour de toute solution, un voisinage contenant des solutions que l'on peut considérer proches. Ce type de méthodes débute par une solution initiale et de proche en proche, d'une solution voisine à une autre, tend à obtenir des solutions de meilleure qualité. Parmi les méthodes de recherche locale, on distingue deux grandes catégories : les méthodes à individu, qui considèrent une unique solution et la font évoluer, et les méthodes à population qui utilisent un ensemble de solutions.

Dans la partie suivante, quelques métaheuristiques sont présentées. Ces méthodes d'optimisation tirent parti des méthodes de recherche locale.

1.4.3 Les métaheuristiques

Le but des métaheuristiques est similaire à celui des heuristiques : obtenir des solutions de bonne qualité en temps raisonnable. Cependant, contrairement à une heuristique, le schéma général des métaheuristiques est totalement indépendant du problème à traiter. De plus, les métaheuristiques peuvent contenir des mécanismes qui permettent que la méthode ne reste bloquée dans un minimum local. Elles permettent ainsi d'explorer l'espace des recherches efficacement, afin de donner de très bonnes solutions mais sans garantie d'optimalité.

Le terme métaheuristique est introduit par (Glover, 1986) et avant que ce terme soit largement adopté, les métaheuristiques étaient souvent appelées heuristiques modernes (Reeves, 1993). Dans la littérature plusieurs définitions ont été proposées pour le terme métaheuristique dans les années 90, on peut citer notamment (Osman et Laporte, 1996) et (Stützle, 1998).

On peut citer une définition plus récente et très connue, mentionnée dans (Blum et Roli, 2003), « une métaheuristique est un ensemble de concepts pouvant être utilisés pour définir des méthodes heuristiques applicables à une large classe de problèmes différents. En d'autres mots, une métaheuristiques peut être vue comme un cadre algorithmique général pouvant s'appliquer à plusieurs problèmes d'optimisation avec assez peu de modifications afin de les faire s'adapter à un problème ».

Certaines métaheuristiques utilisent les concepts additionnels de diversification et d'intensification. Par diversification, on entend généralement une exploration assez large de l'espace de recherche, alors que le terme intensification vient plutôt mettre l'accent sur l'exploitation de l'information accumulée pendant la recherche. Ces notions d'intensification et de diversification sont prépondérantes dans la conception des métaheuristiques, qui doivent atteindre un équilibre entre ces deux dynamiques de recherche. Les deux notions ne sont pas contradictoires, mais complémentaires, et il existe de nombreuses stratégies mêlant à la fois l'un et l'autre des aspects. Bien qu'il existe plusieurs façons de classer et de décrire les métaheuristiques, (Blum et Roli, 2003) proposent de considérer une classification des métaheuristiques avec trois classes : les métaheuristiques basées sur la recherche locale, les métaheuristiques basées sur la population et les métaheuristiques hybrides. Ce choix est motivé par le fait que ce classement permet une description plus claire des algorithmes.

Métaheuristiques basées sur la recherche locale

La méthode GRASP :

La méthode GRASP (Greedy Randomized Adaptive Search Procedure) est une procédure itérative introduite par (Feo et Resende, 1989). C'est une métaheuristique qui cherche à combiner les avantages des heuristiques constructives et des méthodes de voisinage. Elle est constituée de deux étapes :

- La première étape dite constructive, permet de générer une solution initiale.
- La deuxième étape permet l'amélioration de la solution grâce à un processus de recherche locale.

Ces deux étapes sont répétées jusqu'à un critère d'arrêt (par exemple un nombre d'itérations) et la meilleure solution est retournée à la fin. Le GRASP est une méthode très utilisée comme le fait remarquer (Feo et Resende, 1995). Son succès est due en grande partie au fait qu'elle a permis, ces dernières années, d'améliorer beaucoup de bonnes solutions sur des problèmes très différents. Cette méthode a été utilisée sur différents problèmes d'optimisation combinatoires : on peut citer, par exemple, les travaux de (Binato *et al.*, 2002) pour le JSSP ou ceux de (Reghioui *et al.*, 2007) pour le CARP.

La méthode ILS :

La méthode ILS (Iterated Local Search) consiste à effectuer un mouvement aléatoire (mutation) sur une solution qu'on considère comme un minium local, afin « de sortir » de celui-ci puis d'effectuer une recherche locale sur la nouvelle solution. Cette nouvelle solution remplace la solution initiale en cas d'amélioration. Le rôle de l'opérateur de mutation est de créer des perturbations dans les solutions, perturbation qui ne soient ni trop petites (sous peine de rester bloquer dans le même optimum local), ni trop grandes (sous peine d'obtenir des solutions totalement différentes très éloignées de la solution courante). Plusieurs méthodes se sont inspirées de ce principe initialement introduit par (Baum, 1986) pour le TSP, on peut citer (Lourenço *et al.*, 2003) et (Chen *et al.*, 2010).

Métaheuristiques basées sur la population

La méthode ELS :

La méthode ELS (Evolutionary Local Search) est introduite par (Wolf et Merz, 2007). Cette méthode débute avec une population d'un seul individu. En effectuant des mutations et des recherche locales, un ensemble de solutions appelé généralement voisinage peut être généré. La méthode ELS est une extension de la méthode ILS. Cependant, contrairement à la méthode ILS, la méthode ELS met à jour la meilleure solution trouvée lorsque celle-ci est dominée par la meilleure solution générée dans le voisinage.

Métaheuristiques hybrides

Il existe plusieurs possibilités d'hybridation, cependant l'hybridation la plus répandue repose sur la combinaison de métaheuristiques basée sur la recherche locale et de métaheuristiques basées sur une population. L'idée de cette hybridation consiste à exploiter les points forts des deux méthodes, ce qui augmente la puissance des méthodes.

Le GRASP×ELS :

Le GRASP×ELS est une métaheuristique hybridée introduite par (Prins, 2009) pour un problème de VRP. Cette méthode repose sur les deux concepts fondamentaux cités précédemment, à savoir la diversification des solutions grâce au GRASP et l'intensification apportée par la recherche locale de type ELS.

Les schémas algorithmiques des métaheuristiques agissent directement sur les solutions ou parfois sur des objets intermédiaires. Les métaheuristiques sont définies pour parcourir l'ensemble des solutions qui doivent faire l'objet d'un codage le plus performant possibles, puisque chaque itération de la métaheuristique nécessite l'évaluation d'une ou plusieurs solutions ou objets.

1.5 Représentations des solutions

Les premiers éléments de réflexions ayant donné lieu à une classification sur les types de représentations de solutions ont été proposés par (Cheng *et al.*, 1996), dans un article portant sur les problèmes d'ordonnancement et plus particulièrement le problème du job shop. Il faut remarquer que cet article, datant de 1996, a été publié peu de temps après la publication de Bierwirth en 1995 également sur le JSSP, mais 10 ans après les deux articles fondateurs de (Beasley, 1983) et de (Ulusoy, 1985) sur les approches de type SPLIT pour les problèmes de tournées de véhicules.

(Cheng *et al.*, 1996) donne une vision unifiée, en définissant une représentation comme étant à la fois un objet (les auteurs parlent de chromosomes) utilisé comme codage de la solution, appartenant à un espace de codage, et la fonction de codage. Une représentation est donc dépendante d'un problème, et plusieurs représentations sont possibles pour un problème donné. Par exemple, on peut citer neuf représentations pour le problème du JSSP, représentations qui sont détaillées dans l'article de (Cheng *et al.*, 1996).

Les représentations de solutions peuvent être classées suivant un certain nombre de critères, dont :

- la complexité de la fonction de codage ;
- les propriétés de l'espace de codage.

La complexité de la fonction de codage est caractérisée par quatre niveaux d'après (Cheng *et al.*, 1996). Pour le niveau 0, la solution est entièrement caractérisée par l'objet et la fonction de codage n'est alors par utile. Pour le niveau 1, une fonction de codage simple permet de trouver la solution associée à l'objet. Pour le niveau 2, une fonction de codage simple mais heuristique permet de trouver la solution associée à l'objet. Enfin pour le niveau 3, une fonction de codage heuristique et plus complexe permet de trouver la solution associée à l'objet. La complexité des fonctions de codages est directement liée à la quantité d'information transmise par l'objet de codage par rapport à la quantité d'information nécessaire à l'obtention d'une solution du problème.

De plus, en s'intéressant à l'espace de codage, (Cheng *et al.*, 1996) mettent en avant différents types d'espaces de codage. Ils distinguent les espaces de codage ne contenant que des objets permettant d'atteindre l'espace des solutions, et les espaces de codage contenant aussi des objets ne permettant pas d'atteindre l'espace des solutions (violation de certaines contraintes). De plus, certains espaces de codage permettent l'obtention de

solution dans un sous-espace de l'espace des solutions, et certaines solutions ne sont alors pas atteignables.

1.5.1 Introduction

Afin de représenter une solution d'un problème d'ordonnancement ou de tournées de véhicules deux approches peuvent être employées (Cheng *et al.*, 1996) :

- une approche par représentation directe ;
- une approche par représentation indirecte ;

En utilisant une approche par représentation directe de la solution, les méthodes de résolution travaillent directement sur l'espace des solutions. Cette approche a l'avantage de tirer profit des caractéristiques et des propriétés du problème et des solutions. L'objet appartenant à l'espace de codage contient donc directement un ordonnancement pour le problème.

En utilisant une représentation indirecte de la solution, les méthodes de résolution ne travaillent plus directement sur l'espace des solutions mais sur l'espace de codage. Cette approche repose donc sur la définition d'une fonction de codage plus complexe, permettant de définir une solution du problème à partir de sa représentation indirecte. La résolution du problème repose alors sur l'alternance entre les deux espaces, et il s'agit le plus souvent de tirer profit de ces deux espaces pour obtenir un schéma algorithmique performant.

De manière générale, les fonctions de codage peuvent appartenir à plusieurs classes (Figure 1-4) comme le souligne (Cheng *et al.*, 1996) :

- le codage 1-to-n, où un élément de l'espace de codage est associé à plusieurs solutions ;
- le codage n-to-1, où plusieurs éléments de l'espace de codage sont associés à une solution ;
- le codage 1-to-1, où chaque élément de l'espace de codage est associé à une unique solution, ce type de codage est le plus recherché.

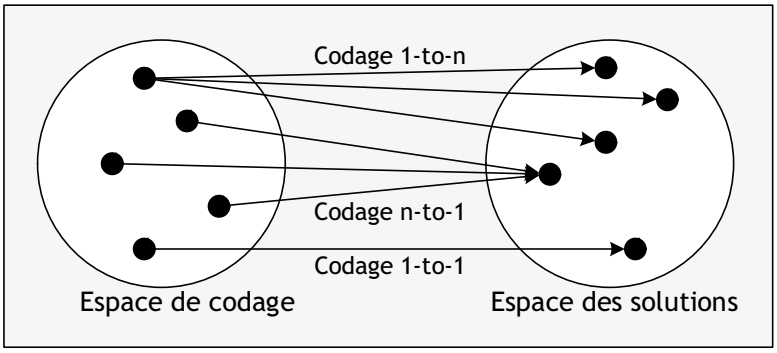


Figure 1-4. Espaces de codages selon (Cheng et al., 1996)

Deux exemples de représentation indirecte de solution sont présentés dans cette partie, avec un exemple dans le domaine des problèmes d'ordonnancement et un exemple dans le domaine des problèmes de tournées de véhicules.

Les fonctions de codage proposées dans la suite de ce chapitre appartiennent aux fonctions de type n-to-1. Tous les objets de l'espace de codage présentés dans la suite permettent d'atteindre l'espace des solutions et chaque solution peut être codée dans l'espace des solutions. Les termes de fonction de codage et fonction d'évaluation (souvent notée f) sont utilisés indifféremment par la suite.

1.5.2 Une représentation par répétition pour les problèmes d'ordonnancement

Dans le cadre du JSSP (Job Shop Scheduling Problem), il est courant d'utiliser comme objet de codage des vecteurs par répétition (Bierwirth, 1995) et de définir une fonction d'évaluation permettant d'associer à un vecteur une solution du problème.

Le vecteur de Bierwirth est une suite ordonnée de numéros de jobs et la $j^{\text{ème}}$ occurrence du numéro de job i dans le vecteur par répétition désigne la $j^{\text{ème}}$ opération du job i. L'avantage de cette représentation est de ne représenter que des ordres valides d'opérations (*i.e.* des ordres compatibles avec les contraintes de précédence). De fait, un vecteur de Bierwirth définit un ordre topologique.

Afin de définir une solution du JSSP à partir d'un vecteur de Bierwirth, une fonction d'évaluation doit être définie. Celle-ci repose sur l'utilisation d'un graphe disjonctif-conjonctif et sur le calcul d'un plus long chemin dans celui-ci. On obtient alors une solution du JSSP avec les dates de début au plus tôt de toutes les opérations. On peut noter que pour le JJSP les vecteurs par répétition permettent d'obtenir uniquement des graphes acycliques.

Le graphe disjonctif-conjonctif est introduit pour la première fois par (Roy et Sussmann, 1964). La particularité de ce graphe est qu'il utilise des arêtes non orientées qui peuvent être orientées dans les deux sens, l'évaluation du graphe n'est possible qu'une fois toutes les arêtes orientées : on parle alors d'un graphe conjonctif (ou d'un graphe disjonctif orienté).

De manière formelle, le graphe disjonctif $G = (V, A, E)$ est un graphe dont les nœuds V représentent des opérations, les arcs A et les arêtes E des contraintes temporelles. Pour modéliser une contrainte de précédence entre une opération i et j on introduit une arête de i vers j d'un coût égal à la durée de l'opération i. L'opération j ne peut alors débuter que lorsque l'opération i est achevée.

Pour orienter un graphe disjonctif, il suffit donc, de fixer l'ordre des opérations puis d'évaluer le graphe avec un algorithme de plus long chemin de type Bellman pour obtenir une solution. On obtient alors les dates de début au plus tôt de toutes les opérations.

Exemple sur une instance du JSSP

Un exemple pour le problème du JSSP est donné ci-dessous, afin d'illustrer cette approche de résolution basée sur la définition du vecteur de Bierwirth et sur le graphe disjonctif.

Une instance d'un problème de JSSP composé de trois jobs et de trois machines est donnée dans le Tableau 1-1. Chaque job possède une gamme composée de trois opérations, et chaque opération est caractérisée par une machine et une durée.

Tableau 1-1
Détail de l'instance pour le JSSP

Job	Opération 1	Opération 2	Opération 3
Job 1	(M1, 2)	(M2, 1)	(M3, 3)
Job 2	(M2, 3)	(M1, 4)	(M3, 1)
Job 3	(M3, 1)	(M2, 2)	(M1, 2)

Á partir des informations contenues dans le Tableau 1-1, le graphe disjonctif correspondant au problème peut être construit (Figure 1-5). Sur ce graphe, toutes les disjonctions sont représentées et les durées des opérations sont reportées sur les arcs modélisant les contraintes de précédence entre des opérations consécutives d'un job. La convention utilisée en ordonnancement pour dessiner le graphe repose sur le fait que les nœuds modélisent le début des opérations, par conséquent la durée indiquée sur un arc correspond à la durée de l'opération à l'origine de l'arc. L'usage en vigueur consiste à représenter toutes les opérations d'un même job sur une ligne. Ainsi, sur la Figure 1-5, la ligne numéro 1 représente les trois opérations du job 1 avec les machines $M1$, $M2$ et $M3$.

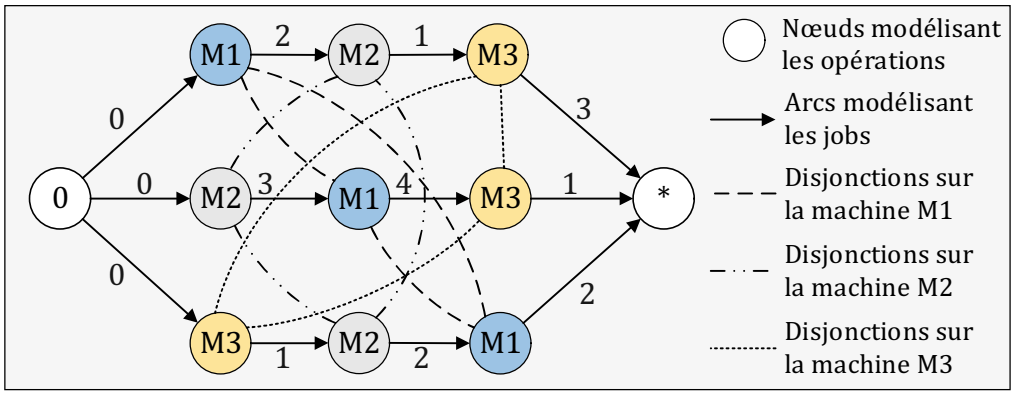


Figure 1-5. Graphe disjonctif du problème

Pour obtenir une solution à ce problème, un vecteur de Bierwirth VB peut être généré aléatoirement, ou être obtenu par une heuristique, ici on considère le vecteur $VB = [2,3,1,2,1,3,2,3,1]$. À partir de ce vecteur et des données du problème, un ordre d'exécution sur chaque machine peut être déduit permettant ainsi d'arbitrer les disjonctions (d'orienter les arêtes) entre deux mêmes machines dans le graphe disjonctif. Comme illustré sur la Figure 1-6, à chaque numéro de job dans le vecteur VB peut être associé la machine permettant de réaliser l'opération du job (ceci dépend de sa position relative par rapport aux autres opérations du même job).

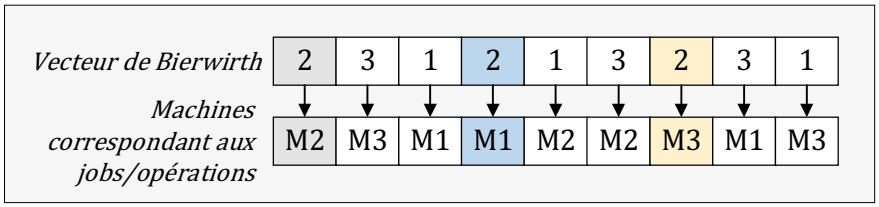


Figure 1-6. Détail du vecteur de Bierwirth

En ne s'intéressant qu'au job 2, la première occurrence correspond à la machine $M2$, la deuxième à la machine $M1$ et la troisième à la machine $M3$. D'autre part, en ne s'intéressant qu'à la machine $M2$, on remarque que celle-ci traite le job 2 puis le job 1 et

enfin le job 3. Avec le vecteur de Bierwirth, le graphe disjonctif peut être orienté. La Figure 1-7 donne la représentation du graphe dans lequel certains arcs non nécessaires à la définition des disjonctions ont été supprimés par transitivité (par exemple l'arc entre le job 1 et le joib 3 pour la machine $M3$ ne figure pas).

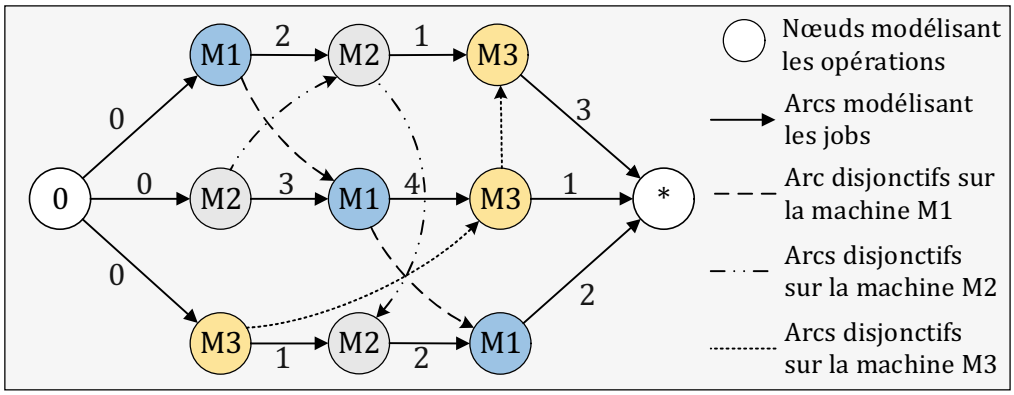


Figure 1-7. Graphe disjonctif orienté lié au vecteur de Bierwirth

Le graphe disjonctif peut alors être évalué afin de calculer un ordonnancement semi-actif pour obtenir une solution du JSSP. Le makespan associé à la solution est donné par la date de début de l'opération finale, modélisée par le nœud $*$. Sur la Figure 1-8, toutes les opérations sont ordonnancées avec des dates de début au plus tôt fournies après l'évaluation du graphe. La solution représentée a un makespan égal à 11.

Le chemin représenté en gras sur la Figure 1-8 constitue le chemin critique et la succession de deux opérations (entre les jobs 1 et 2) sur la machine $M3$ est un bloc critique. Ce chemin est caractérisé par le fait que tout retard d'une opération le long de ce chemin retarde la date de fin de l'ordonnancement. Le chemin critique est le chemin le plus long du graphe conjonctif, et c'est lui qui définit le makespan de la solution. Les blocs sont définis par rapport au chemin critique, et un bloc est une suite de nœuds faisant référence à la même machine. La notion de bloc est proposée par (Grabowski *et al.*, 1996), et elle permet de définir des systèmes de voisinage très efficaces pour le JSSP.

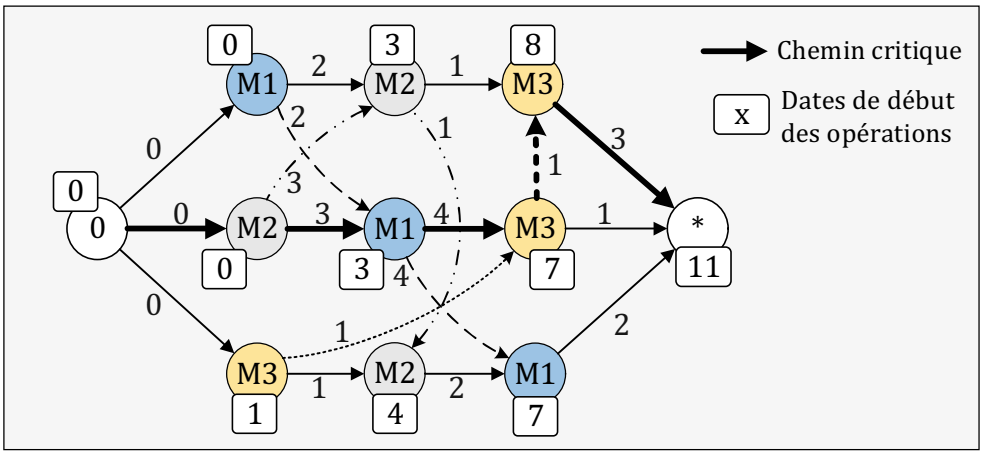


Figure 1-8. Graphe disjonctif orienté et évalué

Une solution du JSSP peut également être représentée avec un diagramme de Gantt. Ce type de diagramme a été imaginé par Henry Laurence Gantt en 1917. Ce diagramme

est très utilisé en gestion de projet, car il donne une représentation de l'état d'avancement des différentes activités (opérations) qui constituent un projet. Sur l'axe des ordonnées du diagramme toutes les tâches à effectuer sont énumérées, tandis que l'axe des abscisses représente les unités de temps (jours, semaines, mois, *etc.*). Chaque tâche est modélisée par un rectangle, dont la position et la longueur indiquent la date de début, la durée et la date de fin. Ce diagramme permet donc de visualiser rapidement :

- les différentes tâches à ordonnancer ;
- la date de début et la date de fin de chaque tâche ;
- la durée de chaque tâche ;
- le chevauchement éventuel des tâches, et la durée de ce chevauchement ;
- la date de début et la date de fin du projet dans son ensemble.

Pour le problème du JSSP, le diagramme de Gantt prend en ordonnées les différentes machines du problème. La même solution que celle représentée sur la Figure 1-8 est représentée sur la Figure 1-9 par un diagramme de Gantt.

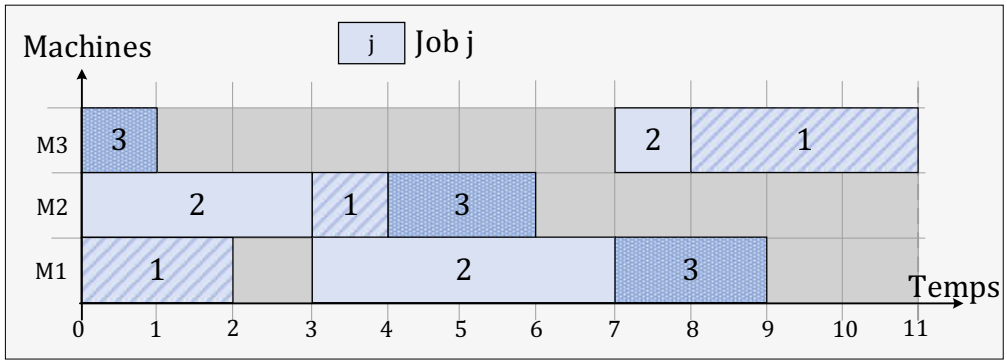


Figure 1-9. Représentation de la solution du JSSP avec un diagramme de Gantt

1.5.3 Une représentation par ordre total pour les problèmes de tournées de véhicules

Sur les tournées de véhicules, beaucoup de méthodes travaillent sur un vecteur de permutations sur l'ensemble des clients, nommé tour géant (solution du TSP) qui est ensuite découpé pour obtenir une solution du problème de VRP par exemple.

Ce vecteur de permutations, avec la fonction de codage associée, repose sur une approche de résolution par représentation indirecte de la solution. Cette idée a été proposée de manière concomitante par Beasley en 1983 et par Ulusoy en 1985 avec des approches nommées « route-first cluster-second ». Notons que ces deux articles, à la base de très nombreuses méthodes utilisées sur les problèmes de tournées de véhicules, sont restés peu cités jusqu'en 2001, date à laquelle ce schéma a été inclus dans un cadre plus général par Lacomme, Prins et Ramdane Chérif (Lacomme *et al.*, 2001).

La fonction de codage proposée porte le nom de SPLIT (Beasley, 1983), et elle repose sur la définition d'un graphe auxiliaire avec un algorithme de plus court chemin. Étant donné un tour géant λ, le graphe auxiliaire G est constitué de $n+1$ nœuds numérotés de 0 à n. Le nœud i correspond au client $\lambda(i)$ à la position i dans le tour géant, et un arc (i,j) dans le graphe auxiliaire représente une tournée réalisable, *i.e.* respectant toutes les

contraintes, partant du dépôt et visitant tous les clients de la sous-séquence $[\lambda(i+1), \dots, \lambda(j)]$ puis retournant au dépôt. Le coût associé à un arc est égal au coût de la tournée qu'il modélise. Le découpage optimal de λ en sous-séquences (tournées) correspond au chemin de coût minimal du nœud 0 au nœud n dans G.

Afin de trouver le plus court chemin dans ce graphe, une marque (label) est propagée de nœud en nœud dans le graphe à l'aide des arcs. Un label sur un nœud j représente une évaluation d'une solution partielle composée des clients entre $\lambda(0)$ et $\lambda(j)$. Pour le problème du VRP, le label est composé d'un coût (date à laquelle le véhicule retourne au dépôt) et d'une information définissant le nœud père. Seul le meilleur label sur chaque nœud est conservé, *i.e.* le label minimisant le coût de la solution partielle associée. Dans l'algorithme SPLIT classique, l'algorithme parcourt G en largeur d'abord. L'algorithme s'arrête donc lorsque le nœud final est atteint, le label alors sauvegardé sur ce nœud correspond à la meilleure solution. Pour retrouver les tournées associées à cette solution, il peut être nécessaire de parcourir de graphe du début à la fin en utilisant les informations sur le nœud père.

La complexité de cet algorithme est en $O(nB)$, avec n le nombre de clients et B le nombre maximal de clients par tournée, en accord la contrainte de capacité. En 2015, (Vidal, 2015) a proposé une autre version du SPLIT en $O(n)$ qui exploite les propriétés de Monge détaillées dans (Aggarwal *et al.*, 1994). Un graphe orienté et pondéré composé de $n+1$ noeuds qui attribue le coût $c(i,j)$ à l'arc (i,j) respecte les propriétés de Monge, si la matrice des coûts associés aux arcs, forme une matrice de Monge. Une matrice $C = \{c(i,j)\}$ de dimension $(n+1) \times (n+1)$ est dite de Monge, si $\forall (i_0, i_1, j_0, j_1) \in [\![0,n]\!]^4, 0 \leq i_0 < i_1 < j_0 < j_1 \leq n$, on a :

$$c(i_0, j_0) + c(i_1, j_1) \leq c(i_0, j_1) + c(i_1, j_0)$$

L'une des raisons pour lesquelles les graphes de Monge sont intéressants vient du fait que les plus courts chemins dans ce graphe peuvent être calculés assez rapidement. En particulier, (Wilber, 1988) a montré que le plus court chemin du nœud 1 au nœud n dans un graphe de Monge peut être calculé en $O(n)$.

Il peut être noté que le graphe auxiliaire pour le SPLIT est un graphe de Monge (Figure 1-10). À partir d'une séquence de client $[i,j,k,l]$ appartenant au tour géant λ et permettant de modéliser les nœuds du graphe auxiliaire, on a $0 \leq i < j < k < l \leq n$. Par analogie, le coût des arcs étant égal au coût de la tournée modélisant l'arc (avec $d_{i,j}$ le coût de transport entre le client i et le client j), on a :

$$\left.\begin{matrix} c(i,k) = d_{0,j} + d_{j,k} + d_{k,0} \\ c(j,l) = d_{0,k} + d_{k,l} + d_{l,0} \end{matrix}\right\} c(i,k) + c(j,l) = d_{0,j} + d_{j,k} + d_{k,0} + d_{0,k} + d_{k,l} + d_{l,0}$$

$$\left.\begin{matrix} c(i,l) = d_{0,j} + d_{j,k} + d_{k,l} + d_{l,0} \\ c(j,k) = d_{0,k} + d_{k,0} \end{matrix}\right\} c(i,l) + c(j,k) = d_{0,j} + d_{j,k} + d_{k,l} + d_{l,0} + d_{0,k} + d_{k,0}$$

On remarque alors que $c(i,k) + c(j,l) = c(i,l) + c(j,k)$, la propriété de Monge est respectée.

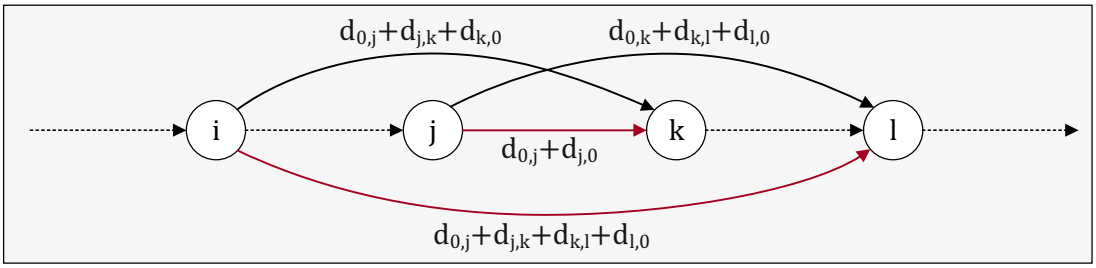


Figure 1-10. Propriété de Monge dans le graphe auxiliaire pour le SPLIT

Par conséquent, l'algorithme SPLIT peut être adapté afin d'obtenir une complexité en $O(n)$. Pour ce faire, (Vidal, 2015) introduit plusieurs notations :

- $p[i]$ le coût du plus court chemin pour visiter les clients 0 à i ;
- $D[i] = \sum_{k=1}^{i-1} d_{k,k+1}$ la distance cumulée entre le client 0 et i, ce vecteur est précalculé ;
- $Q[i] = \sum_{k=1}^{i} q_k$ la quantité de demande cumulée entre le client 0 et i, ce vecteur est précalculé ;

Grâce à ces notations et avec la capacité Q du véhicule, le coût d'une tournée peut s'écrire :

$$c(i,j) = d_{0,i+1} + D[j] - D[i+1] + d_{j,0}, \text{ ssi } Q[j] - Q[i] \leq Q$$

Le but de l'algorithme est de parcourir l'ensemble des nœuds dans l'ordre donné par le tour géant, et pour chaque nœud *i*, calculer le coût le plus petit pour y accéder (grâce aux coûts associés aux nœuds précédents et à l'application d'une règle de dominance). Cet algorithme repose également sur une structure de données de type « double-ended queue » contenant l'ensemble des nœuds par lesquels une tournée peut s'achever.

L'algorithme consiste à déterminer, lors de chaque itération, le plus court chemin jusqu'au nœud courant. Pour chaque itération, deux configurations peuvent être envisagées. Prenons l'exemple de l'itération $i+1$, dans laquelle le plus court chemin de 0 à $i+1$ dans le graphe auxiliaire doit être calculé (l'itération précédente a permis de calculer le plus court chemin de 0 à i, le coût $p[i]$ est alors connu).

Deux configurations sont alors possibles (si la demande totale n'excède pas la capacité du véhicule) :

- le nœud $i+1$ est le premier nœud d'une nouvelle tournée (Figure 1-11, cas 1) ;
- le nœud $i+1$ est ajouté à la tournée précédente (Figure 1-11, cas 2).

Pour connaître laquelle de ces deux configurations est la meilleure, les solutions prenant en compte le nœud suivant $(n+2)$ sont comparées (Figure 1-11, cas 1a, 1b, 2a et 2b).

Pour comparer les deux configurations (1 et 2), il suffit de comparer la meilleure solution (1b) obtenue à partir de la configuration (1) avec la pire solution (2a) obtenue à partir de la configuration (2).

Il s'agit donc de comparer

$$c_{1b} = p[i] + d_{0,i+1} + D[i+2] - D[i+1]$$

et

$$c_{2a} = p[i] - d_{i,0} + D[i+1] - D[i] + d_{i+1,0} + d_{0,i+2}$$

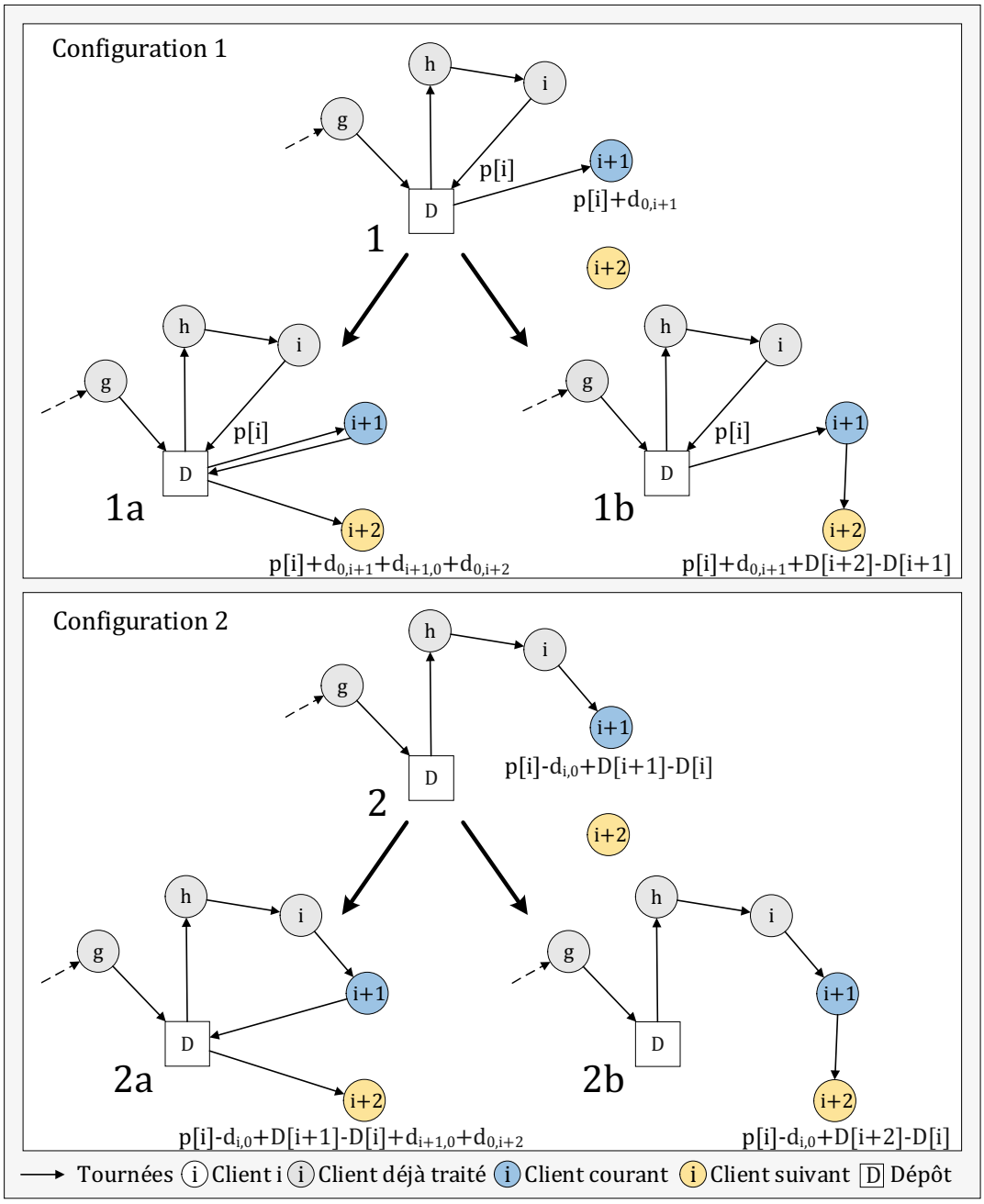


Figure 1-11. Solutions possibles à l'itération $i+1$

Si $c_{1b} > c_{2a}$, alors la pire solution obtenue avec la configuration 2 est meilleure que la meilleure solution obtenue avec la configuration 1, et par conséquent, la configuration 2 domine la configuration 1.

Si $c_{2a} < c_{1b}$, cette inégalité peut se réécrire $p[i+1] + d_{0,i+2} - D[i+2] < p[i] + d_{0,i+1} - D[i+1]$. Dans ce cas, on dit que le nœud $i+1$ domine i puisqu'il est plus intéressant de terminer la tournée courante par le nœud $i+1$ que par le nœud i. La liste des nœuds non dominés est stockée dans la structure de données de type « double-ended queue ». Lors de l'itération i, pour chaque nœud j stocké dans la « double-ended queue », la tournée entre j et i doit respecter la capacité du véhicule, sinon le nœud j est supprimé

de la structure de données. L'algorithme se termine lorsque tous les nœuds ont été parcourus.

Exemple de découpage sur une instance du VRP

Un exemple pour le problème du VRP est donné ci-dessous, afin d'illustrer cette approche de résolution basée sur la définition d'un tour géant et sur l'exécution du SPLIT classique sur un graphe auxiliaire.

L'instance du problème de VRP traitée est composée de cinq clients, et d'un véhicule de capacité 10. Chaque client est caractérisé par une demande donnée dans le Tableau 1-2.

Tableau 1-2
Demandes des clients pour l'instance du VRP

Client	1	2	3	4	5
Demande	5	3	4	2	6

On considère ici, le tour géant reprenant l'ordre lexicographique des clients $\lambda = [1,2,3,4,5]$. Ce tour géant peut être représenté sur un graphe représentant les clients avec les distances séparant deux clients consécutifs dans le tour géant, ainsi que les distances par rapport au dépôt noté D (Figure 1-12).

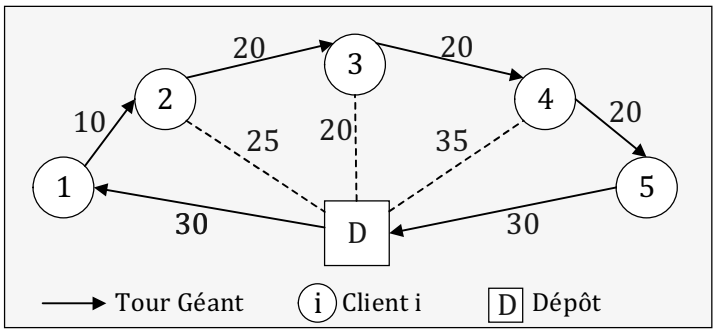


Figure 1-12. Représentation du tour géant avec les distances

Á partir des informations contenues dans le Tableau 1-2 et la Figure 1-12, le graphe auxiliaire correspondant au problème et au tour géant peut être construit (Figure 1-13). Sur ce graphe, tous les arcs modélisant les tournées réalisables sont représentés et les durées des tournées sont reportées sur les arcs.

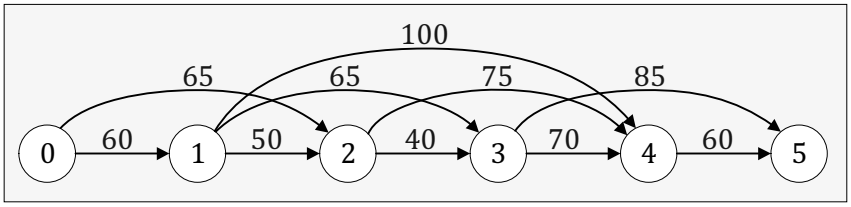


Figure 1-13. Graphe auxiliaire

Le graphe auxiliaire est initialisé avec le label 0 (label initial) sur le nœud 0. Toutes les itérations, ainsi que le plus court chemin sont explicités sur la Figure 1-14. La première itération consiste à propager le label initial, sur tous les arcs d'origine 0 du graphe auxiliaire. Cette première étape génère deux labels : un label sur le nœud 1 de coût 60

traduisant la tournée $(0,1,0)$ et un label sur le nœud 2 de coût 65 traduisant la tournée $(0,1,2,0)$. Lors de la deuxième itération, le label sur le nœud 1 est propagé en utilisant les trois arcs sortant du nœud i, et trois labels sont alors générés. Le label généré pour le nœud 2 modélise la solution partielle composée des tournées $(0,1,0)(0,2,0)$ de coût 110. Ce label est dominé par le label déjà présent sur le nœud 2 (de coût 65) et celui-ci n'est donc pas inséré. Les deux autres labels (de coûts 125 et 160) sont insérés respectivement sur les nœuds 3 et 4.

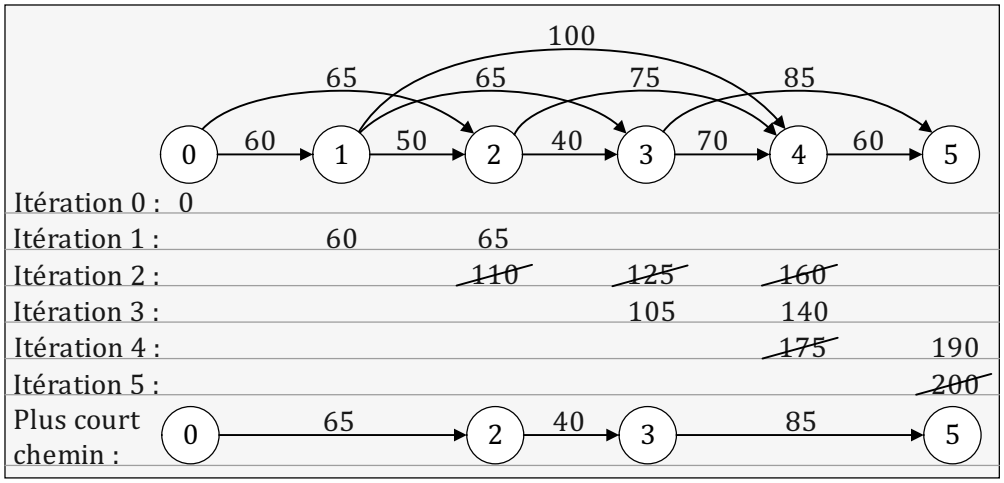


Figure 1-14. Détail de l'algorithme SPLIT avec le détail des labels

À la fin de l'algorithme, le label sur le nœud final modélise la meilleure solution qui est reconstruite à partir des labels ayant permis de générer le meilleur label. La meilleure solution possède donc un makespan égal à 190 et est constituée de trois tournées, $(0,1,2,0)$, $(0,3,0)$ et $(0,4,5,0)$. Cette solution est représentée sur la Figure 1-15.

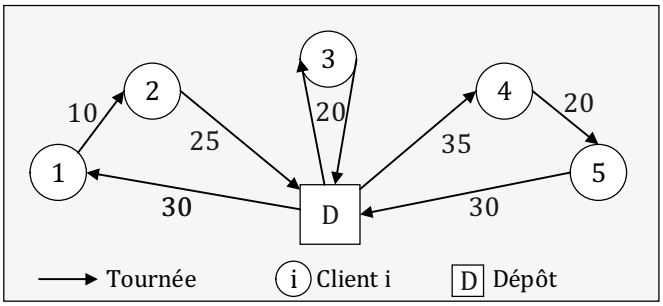


Figure 1-15. Représentation de la meilleure solution pour le tour géant $[1,2,3,4,5]$

1.6 Conclusion du chapitre

Ce chapitre présente les grandes familles de problème d'ordonnancement et de tournées de véhicules avec des exemples théoriques. Un cadre global est présenté, permettant une mise en perspective des problématiques soulevées par les problèmes intégrés, notamment au sein de la chaîne logistique. Plusieurs méthodes de résolution sont détaillées, aussi bien pour les problèmes d'ordonnancement que pour les problèmes de transport. Un intérêt particulier est accordé aux métaheuristiques, qui constituent un schéma de résolution très utilisé pour traiter des problèmes de « grande taille ». Les liens entre ces métaheuristiques et les notions d'espace de codage sont rappelés.

La résolution des problèmes intégrés repose sur l'utilisation d'approches venant du domaine de l'ordonnancement et d'approches venant du transport. L'enjeu de la thèse est de développer des méthodes de résolution et des outils de modélisation génériques, en adaptant des outils spécifiques aux deux familles de problèmes.

Deux problèmes différents sont abordés par la suite, et ces problèmes se distinguent à la fois par le type de problèmes d'ordonnancement et par le type de problèmes de transport (on passe d'un problème à une machine à un problème de type RCPSP et d'un problème à un véhicule à plusieurs véhicules), mais aussi par le fait que le premier problème étudié (le PTSP) a déjà fait l'objet de plusieurs publications (récentes) alors que le deuxième est un nouveau problème.

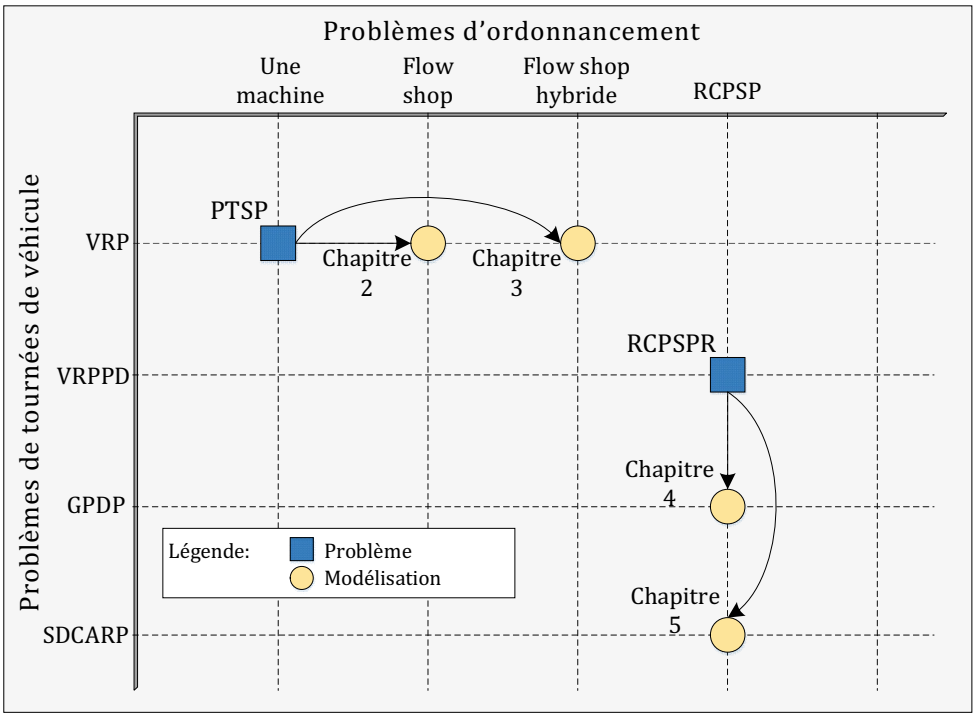


Figure 1-16. Schéma synthétique des problèmes intégrés étudiés

Les problèmes de type PTSP (Production and Transportation Scheduling Problem) intègrent, à un problème d'ordonnancement à une machine un problème de transport de type VRP, et les problèmes de type RCPSPR (Resource-Constrained Project Scheduling

Problem with Routing) intègrent à un problème d'ordonnancement de type RCPSP un problème de transport de type VRPPD (Figure 1-16).

Ces problèmes sont étudiés dans les différents chapitres de cette thèse, chaque chapitre proposant différentes modélisations des sous-problèmes au cours des étapes de résolution. Par exemple, le problème du PTSP étudié dans les chapitres 2 et 3 peut se modéliser soit comme un flow shop, soit comme un flow shop hybride en fixant certaines variables du problème. De même, le problème du RCPSPR étudié dans les chapitres 4 et 5 peut se modéliser soit comme un problème proche du GPDP, soit comme un problème proche du SDCARP en fixant certaines variables du problème.

CHAPITRE 2

Le problème d'ordonnancement de la production et du transport avec un seul véhicule

Ce chapitre traite du problème d'ordonnancement de la production et du transport (Production and Transportation Scheduling Problem - PTSP) avec un seul véhicule. Ce problème intégré modélise un problème de production et de livraison de produits périssables. La contrainte liée à la périssabilité des produits impose une durée maximale entre la fin de production du produit et sa livraison : cette durée correspond à la durée de vie du produit. L'originalité du problème repose sur cette contrainte de durée de vie des produits qui se modélise sous la forme de time-lags maximaux.

Nous proposons dans ce chapitre une nouvelle méthode de résolution du PTSP à l'aide d'une métaheuristique de type GRASP×ELS permettant la résolution conjointe de l'ordonnancement et du routage. La métaheuristique s'appuie sur un codage indirect des solutions sous forme de tours géants. Un tour géant est alors optimalement découpé en lots par une méthode SPLIT minimisant le temps de transport. Les lots ainsi constitués par la méthode SPLIT, permettent une modélisation du problème sous la forme d'un flow shop à deux machines avec time-lags maximaux. La résolution de ce sous-problème est réalisée par deux approches différentes. La première approche, de type « relaxation » consiste, à relaxer la contrainte sur la durée de vie du produit (time-lags maximaux) et le problème se résout optimalement avec l'algorithme de Johnson. La deuxième approche consiste cette fois à « sur-contraindre » le problème pour le transforme en flow shop à deux machines avec no-wait qui peut être résolu par l'algorithme de Gilmore-Gomory.

Les résultats numériques présentés montrent que l'approche basée sur une modélisation de type flow shop à deux machines avec no-wait est meilleure que les méthodes publiées précédemment et notamment meilleure que celle proposée par Geismar *et al.* en 2008.

2.1 Introduction et état de l'art du problème

L'étude des problèmes intégrés de production et de transport est récente et suscite de plus en plus d'intérêt. Sur ces problèmes plusieurs états de l'art ont été publiés, dont ceux de (Saramiento *et al.*, 1999), (Chen, 2010) et (Moons, 2017). Ces problèmes peuvent porter différentes dénominations, dont par exemple Integrated Production and Outbound Distribution Scheduling (IPODS), Integrated Production and Distribution Problem (IPDP) ou encore Production and Transportation Scheduling Problem (PTSP). Dans ces problèmes, une solution est caractérisée par un ordonnancement de la production et des livraisons, c'est-à-dire par deux sous-problèmes qu'il est difficile de résoudre itérativement pour obtenir de bonnes solutions. L'ordonnancement de la production consiste à déterminer, pour chaque demande traitée, les dates de réalisations ainsi que

l'affectation de celles-ci à un lieu de production. L'ordonnancement des livraisons consiste à déterminer le nombre de véhicules nécessaire ainsi que les dates de livraison des clients. Les problèmes se distinguent les uns des autres en fonction de critères variés dont par exemple :

- le nombre de lieux de production avec différentes configurations machines (une unique machine, des machines en parallèle) ;
- les types d'ateliers particuliers comme des configurations de type flow shop, job-shop, *etc*. ;
- le nombre de clients, chaque client pouvant formuler des contraintes spécifiques (précédences, fenêtres de temps pour la livraison, *etc*.) ;
- le nombre de véhicules disponibles (limité ou non) ;
- les capacités des véhicules (limitées ou non).

(Chen, 2010) classifie les problèmes IPODS existants en cinq classes :

- les problèmes de livraison individuelle et immédiate ;
- les problèmes de livraison par lots à un seul client par livraison direct ;
- les problèmes de livraison par lots à plusieurs clients par livraison direct ;
- les problèmes de livraison par lots à plusieurs clients avec des tournées de livraison ;
- les problèmes avec des dates de départ fixe pour la livraison.

Dans le cadre de ce chapitre, notre intérêt s'est porté sur les problèmes intégrés de production et de transport, dans le cas de produits périssables. Dans ce cas particulier, la production et le transport sont intimement liés puisque ce genre de produit ne peut pas être stocké pendant de longues périodes. Une résolution conjointe est donc nécessaire pour satisfaire les demandes et les livraisons en minimisant les coûts. En plus de cette contrainte, ce chapitre se focalise sur les problèmes avec un seul véhicule, un seul lieu de production et un seul type de produit.

(Chen, 2010) mentionne deux articles traitant de ce type de problèmes avec des produits périssables : (Armstrong *et al.*, 2008) et (Geismar *et al.,* 2008). Ils font tous les deux partie de la même classe de problèmes : livraison par lots à plusieurs clients avec des tournées de livraison. Les contraintes de périssabilité dans ces problèmes sont dues à des composés chimiques industriels. La livraison des produits doit être réalisée en temps limité une fois leurs productions terminées. Dans ces deux articles la durée de vie d'un produit commence dès que la production du lot contenant le produit est achevée. Un lot est composé des demandes d'un ou de plusieurs clients.

(Armstrong *et al.*, 2008) ont étudié le problème intégré de production et de transport avec un seul véhicule et un unique lieu de production. L'ordre de livraison des clients est fixé et chaque demande client est caractérisée par la quantité de produit qui doit lui être livrée avec une fenêtre de livraison associée. Le problème consiste alors à définir le transport d'un sous-ensemble de clients minimisant le nombre de clients non traités.

(Geismar *et al.,* 2008) ont traité le problème du PTSP composé d'un ensemble de n clients, d'un unique lieu de production 0 dont le taux de production $r > 0$ peut varier et d'un véhicule de capacité limité Q. À chaque client est associé une demande, c'est-à-dire, une quantité de produits qui doit lui être livrée en respectant la durée de vie du produit B. L'objectif est de minimiser le temps nécessaire pour satisfaire l'ensemble des demandes clients, *i.e.* le makespan C_{max}. Le problème peut être assimilé à un problème

de flow shop à deux machines avec time-lags maximaux. Pour résoudre ce problème, les auteurs ont implémenté un algorithme génétique et un algorithme mémétique avec la définition de trois bornes inférieures. Ils ont également prouvé que ce problème est NP-complet au sens fort.

Un autre article a été publié récemment sur le problème introduit par (Geismar *et al.,* 2008). (Karaoğlan et Kesen, 2017) ont proposé une autre méthode de résolution, avec un algorithme de type branch-and-cut, sur le même problème que (Geismar *et al.,* 2008). Leur algorithme a été testé sur les mêmes instances que (Geismar *et al.,* 2008), les résultats numériques prouvent que leur algorithme fournit de meilleures solutions.

De nombreux autres articles ont été publiés sur les problèmes intégrés de production et de transport ces dernières années. Le Tableau 2-1 regroupe l'ensemble des articles traitant des problèmes de production avec un problème de tournées de véhicules intégré, dans le cas où la production est réalisée par une seule machine et le transport par un seul véhicule (Moons, 2017).

Tableau 2-1
Caractéristiques des problèmes de production et de transport intégré avec une machine de production et un véhicule.

	Production				Distribution							Objectif		
	Un seul lieu de production	Production en lots	Un seul produit	Temps de production	Un seul véhicule	Temps de transport	Plusieurs tournées	Ordre des clients fixé	Fenêtres de temps	Dates limites de livraison	Produits périssables	Makespan	Coût	Demandes satisfaites
Chang et Lee (2004)	●	●	●	●	●	●	●					●		
Li *et al.* (2005)	●	●	●	●	●	●	●						●	
Geismar *et al.* (2008)	●	●	●	●	●	●	●				●	●		
Armstrong *et al.* (2008)	●	●	●	●	●	●	●	●	●		●			●
Cheref *et al.* (2016)	●	●	●	●	●	●	●			●			●	
Karaoğlan et Kesen (2017)	●	●	●	●	●	●	●				●	●		
Notre proposition	●	●	●	●	●	●	●				●	●		

Les trois articles cités dans le Tableau 2-1 de (Chang et Lee, 2004), (Li *et al.*, 2005) et (Cheref *et al.,* 20016), ne traitent pas les cas des produits périssables, mais les problèmes étudiés dans ces articles sont très proches. (Cheref *et al.,* 20016) prennent en compte une contrainte supplémentaire avec des dates limites de livraison. Ces dates limites de livraison sont exprimées par rapport au début de l'horizon de temps du problème, et non pas par rapport à la fin de la production du produit à livrer (cas des produits périssables). Ils sont les premiers à étudier un problème intégré dans un environnement incertain. Leur objectif est de minimiser un critère de robustesse lié au retard maximal de livraison. Dans le problème étudié par (Chang et Lee, 2004), les produits doivent être livrés dans une ou deux zones, chaque client d'une même zone est donc associé à une seule localisation. (Li *et al.*, 2005) ont généralisé le problème de (Chang et Lee, 2004) en associant à chaque

client une localisation spécifique. L'objectif des problèmes étudiés par (Chang et Lee, 2004) et (Li *et al.*, 2005) est de minimiser respectivement, le makespan, ou, la somme des dates d'arrivée (de livraison) sur l'ensemble des clients.

La suite de ce chapitre permet de détailler le problème du PTSP, initialement introduit par (Geismar *et al.,* 2008). Nous proposons dans ce chapitre de nouvelles méthodes de résolution pour le PTSP à un véhicule, intégrées au sein d'une métaheuristique de type GRASP×ELS.

2.2 Définition du problème

Le problème étudié dans ce chapitre a été introduit par (Geismar *et al.,* 2008). Cette section a pour but de donner, en plus de la formalisation du problème, une présentation algorithmique du problème dans un cadre global mettant en évidence les différentes étapes de résolution et les liens entre ces étapes.

2.2.1 Définition générale

Le PTSP avec un seul véhicule peut être défini grâce aux notations suivantes :
0 : le lieu de production situé en $(x, y) = (0, 0)$
r : le taux de production, $r \in \{1, 2, 3\}$
B : la durée de vie du produit
Q : la capacité du véhicule
n : le nombre de clients
E : l'ensemble des clients, $i \in \{1, \dots, n\}$
q_i : la quantité demandée par le client i, $i \in \{1, \dots, n\}$
(x_i, y_i) : les coordonnées géographiques du client i, $i \in \{1, \dots, n\}$
$\tau_{i,j}$: le coût du transport associé au trajet entre le client i et le client j (cette matrice respecte l'inégalité triangulaire)

Le coût du transport associé à la durée du trajet entre le client i et le client j, noté $\tau_{i,j}$, correspond au plus court chemin entre les deux clients, ce coût peut donc être pré-calculé. Pour des raisons de lisibilité, le trajet correspondant à $\tau_{i,j}$ est modélisé dans le graphe par une ligne droite.

Une solution est composée d'un ensemble de $n_j \leq n$ jobs $J_1, J_2, \dots, J_{nj}$. Chaque job $j \in [\![1, n_j]\!]$ regroupe un ensemble de client à servir, et est composé de deux opérations O_{j1} et O_{j2}. Les opérations O_{j1} correspondent aux opérations de production, elles sont toutes affectées à l'unique lieu de production O que l'on peut également nommer dépôt. Les opérations O_{j2} correspondent aux opérations de transport, elles sont toutes affectées au seul véhicule disponible. Chaque opération de production correspond à la production d'un lot produit répondant à un ensemble de demande client. Chaque opération de transport permet de décrire une sous-tournée, l'ensemble des sous-tournées permet de définir la tournée du véhicule. Une tournée est caractérisée par une séquence d'opérations de collecte et de livraison (en Anglais, on parle de pickup et delivery). Une sous-tournée commence par une opération de collecte suivie d'une ou plusieurs opérations de livraison. Les opérations de collecte sont effectuées au niveau du lieu de production et les opérations de livraison sont associées aux clients en fonction des quantités de produit demandées.

Pour des raisons pratiques on peut considérer que chaque client est situé au niveau d'un nœud dans un graphe modélisant un réseau routier.

Pour caractériser la solution les notations suivantes sont introduites :
nj : le nombre de jobs
C_j : l'ensemble des clients regroupé dans le job numéro j
O_{j1} : opération de production associée au job numéro j
O_{j2} : opération de transport associée au job numéro j
p_j : la durée de l'opération de production O_{j1}
p'_j : la quantité produite pendant l'opération de production O_{j1}
t_j : la durée de l'opération de transport O_{j2}^k
d_{j1} : date de début de l'opération de production associée au job numéro j
d_{j2} : date de début de l'opération de transport associée au job numéro j
f_{j1} : date de fin de l'opération de production associée au job numéro j
f_{j2} : date de fin de l'opération de transport associée au job numéro j

Des contraintes supplémentaires doivent être prises en compte pour définir une solution :
- chaque tournée et sous-tournée doit commencer et terminer au dépôt ;
- l'ordre des opérations de production et de transport est identique ;
- tous les clients d'une sous-tournée doivent être livrés dans les B unités de temps suivant la fin de la production de leurs demandes ;
- le total des demandes client d'une sous-tournée T ne doit pas excéder la capacité du véhicule $\sum_{i \in T} q_i \leq Q$;
- une seule livraison par client (chaque client doit être visité une et une seule fois).

2.2.2 Illustration des problématiques autour du PTSP avec un exemple

Dans ce paragraphe, le problème du PTSP avec un véhicule étudié par (Geismar *et al.,* 2008) est présenté en détail avec un exemple et une solution pour illustrer les points-clé de la résolution intégrée. Pour résoudre ce problème, deux sous-problèmes doivent être résolus conjointement : le transport et l'ordonnancement.

La Figure 2-1 présente un exemple de solution d'un problème du PTSP avec cinq clients, un lieu de production avec un taux de production r égal à 1, un véhicule de capacité Q égale à 10 produits et un produit dont la durée de vie est égale à 7. Rappelons qu'un taux de production r égal à 1 correspond à des situations dans lesquelles la durée de production et la quantité à produire sont liées par un coefficient de proportionnalité égal à 1. La fabrication d'un produit nécessite une unité de temps, toujours dans le cas $r = 1$. Dans l'exemple utilisé pour construire la solution de la Figure 2-1, les temps de production sont calculés grâce au ratio $r = 1$ et aux quantités demandées par chacun des clients. Dans cet exemple, les quantités demandées par les clients sont les suivantes : $q_1 = 1$, $q_2 = 1$, $q_3 = 6$, $q_4 = 1$ et $q_5 = 3$.

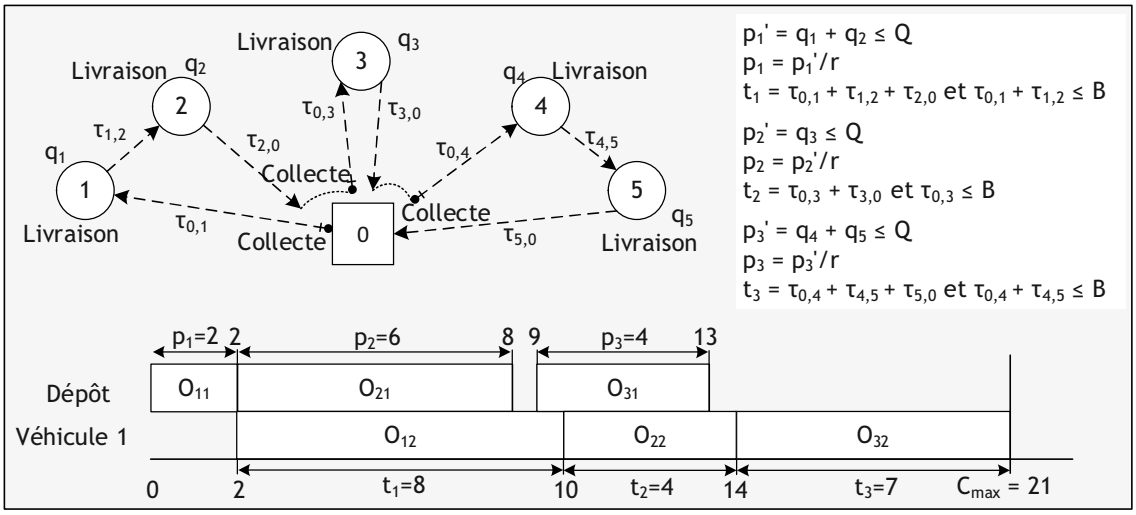


Figure 2-1. Exemple de solution du PTSP avec un véhicule

Dans la solution présentée dans la Figure 2-1, les clients sont divisés en trois groupes qui se composent de 6 opérations, 3 opérations de production et 3 opérations transport. Ces opérations sont notées O_{ij}, et elles permettent de définir 3 jobs :

- la suite ordonnée des opérations O_{11} et O_{12} définit le job 1.
  O_{11} : opération de production correspondant à la fabrication des produits des clients $\{1,2\}$
  O_{12} : opération de transport correspondant à la livraison des produits aux clients $\{1,2\}$
- la suite ordonnée des opérations O_{21} et O_{22} définit le job 2.
  O_{21} : opération de production correspondant à la fabrication des produits du client $\{3\}$
  O_{22} : opération de transport correspondant à la livraison des produits du client $\{3\}$
- la suite ordonnée des opérations O_{31} et O_{32} définit le job 3.
  O_{31} : opération de production correspondant à la fabrication des produits des clients $\{4,5\}$
  O_{32} : opération de transport correspondant à la livraison des produits aux clients $\{4,5\}$

Il s'agit donc d'ordonnancer trois opérations machine *(O_{11}, O_{21}, O_{31})* et trois opérations de transport *(O_{12}, O_{22}, O_{32})* et de calculer leur dates de début au plus tôt. Les trois opérations de transport correspondent à une sous-tournée. La tournée du véhicule est donc composée de trois sous-tournées réalisées consécutivement dans le temps. La première sous-tournée effectue la livraison des clients 1 et 2, modélisant une sous-tournée de la forme $(0,1,2,0)$, la deuxième effectue la livraison du client 3 et modélise une sous-tournée de la forme $(0,3,0)$ et la troisième effectue la livraison des clients 4 et 5 c'est-à-dire une sous-tournée modélisée par $(0,4,5,0)$. La répartition des clients au sein de ces 3 jobs étant connue, les durées de toutes les opérations peuvent être calculées.

Dans l'exemple de la Figure 2-1, le taux de production $\mathrm{r} = 1$ signifie que la durée de chaque opération de production $p_j, j \in \{1,2,3\}$ est égale à la quantité produite pendant chaque opération de production $p'_j, j \in \{1,2,3\}$.

La première opération de production O_{11} consiste en la production de $p'_1 = 2$ produits pendant une durée $p_1 = 2$ unités de temps. Les produits doivent ensuite être livrés aux deux clients durant l'opération de transport O_{12}, effectuée par le véhicule. De la même

façon, la durée de l'opération de production O_{21} est de $p_2 = 6$ unités de temps, et la durée de l'opération de production O_{31} est de $p_3 = 4$ unités de temps. Pour ce qui est des durées des opérations de transport, elles peuvent également toutes être calculées à partir des données suivantes : $\tau_{0,1} = \tau_{2,0} = \tau_{0,4} = \tau_{4,5} = 3$, $\tau_{1,2} = \tau_{0,3} = 2$ et $\tau_{5,0} = 1$. Par conséquent, la première opération de transport O_{12} consiste à livrer les produits demandés par les clients 1 et 2, en effectuant une sous-tournée de durée $t_1 = 8$. De la même façon, la durée de l'opération de transport O_{22} est de $t_2 = 4$ unités de temps, et la durée de l'opération de transport O_{32} est de $p_3 = 7$ unités de temps.

Sur le diagramme de Gantt présent dans la Figure 2-1 permettant de représenter la solution, il peut être noté que l'ensemble des opérations de production et de transport sont ordonnancées au plus tôt. Cependant, un temps d'attente au niveau de la production entre les opérations O_{21} et O_{31} d'une unité de temps demeure. En effet, l'opération de production O_{31} ne peut démarrer à la date 8 si l'opération de transport débute à la date 14 en raison de la contrainte de périssabilité du produit. En effet, l'opération de transport O_{32} est chargée d'effectuer la sous-tournée $(0,4,5,0)$. Dans la solution, le véhicule quitte le dépôt à la date 14, le client 4 est livré à la date 17, le client 5 est livré à la date 20 et le véhicule est de retour au dépôt à la date 21. Pour respecter la contrainte sur la durée de vie du produit, il faut qu'une fois une opération de production achevée, tous les clients de l'opération de transport associée aient été livrés dans les B unités de temps. Cela revient à dire, qu'une fois une opération de production achevée, le dernier client de l'opération de transport associée doit être livré dans les B unités de temps (ce point est détaillé dans la suite). Dans l'exemple de la Figure 2-1, l'opération de production O_{31}doit donc s'achever au plus tôt à la date $20 - 7 = 13$ et donc débuter au plus tôt à la date 9. Le même raisonnement peut être effectué sur les opérations des deux autres jobs.

Sur la Figure 2-1, lorsque l'ordre des clients est défini, la tournée peut être entièrement caractérisée par :

- une séquence d'opérations de collecte C et de livraison L avec des retours au dépôt 0 à la fin de chaque sous tournée : $C, L, L, 0, C, L, 0, C, L, L, 0$ (voir Figure 2-2).
- des dates de départ et d'arrivée du véhicule au niveau du dépôt.

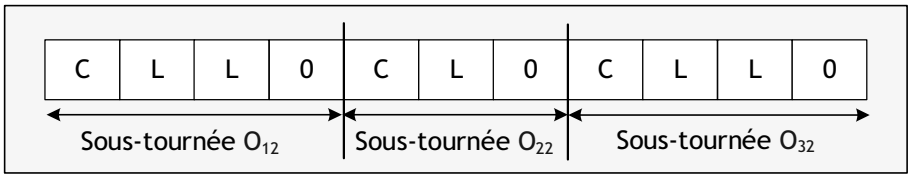


Figure 2-2. Détail des sous-tournées de la solution du PTSP avec un véhicule

Dans la suite de ce chapitre, par abus de langage, on utilise le terme de sous-tournée O_{ij} pour désigner la sous-tournée associée à l'opération de transport O_{ij}.

Le problème de transport associé au PTSP peut être associé à un problème de tournées de véhicules avec livraisons (VRPPD – Vehicle Routing Problem with Pickup and Delivery) comme le rappelle (Chassaing, 2015). Le chargement du véhicule est donc une fonction du temps décroissante sur les intervalles représentant les sous-tournées (voir Figure 2-3).

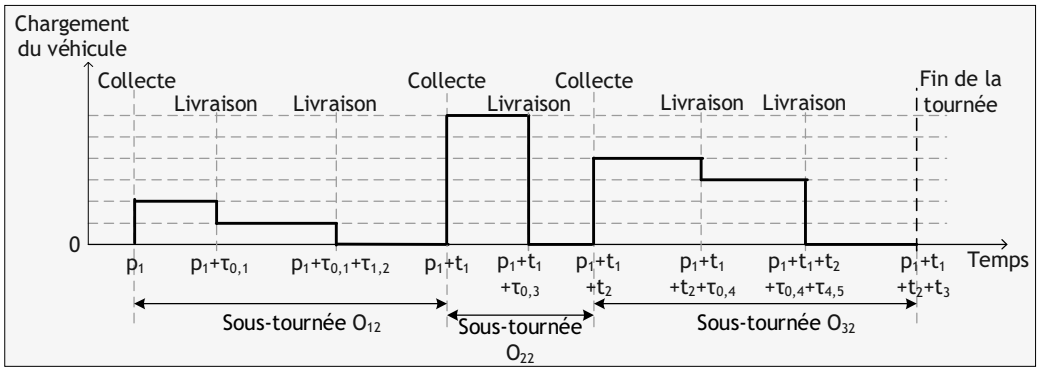


Figure 2-3. Suivi du chargement du véhicule au cours du temps

Chaque nœud client (qu'on peut aussi nommer nœud de livraison) possède une fenêtre de temps permettant de définir la date maximale d'arrivée du véhicule sur le nœud, pour respecter la durée de vie du produit. Cette fenêtre de temps dépend pour un client, de la date de fin de l'activité de production associée et de la disponibilité du véhicule affecté au transport. Ces contraintes mettent en avant l'enjeu d'une résolution intégrée pour ce problème puisque la production et le transport sont directement liés.

La Figure 2-4 illustre la solution détaillée de l'exemple en Figure 2-1 avec, sur la partie haute de la figure, une représentation du chargement du véhicule au cours de la tournée et des trois sous-tournées, et sur la partie basse, un graphe illustrant la coordination entre la production et le transport au cours du temps.

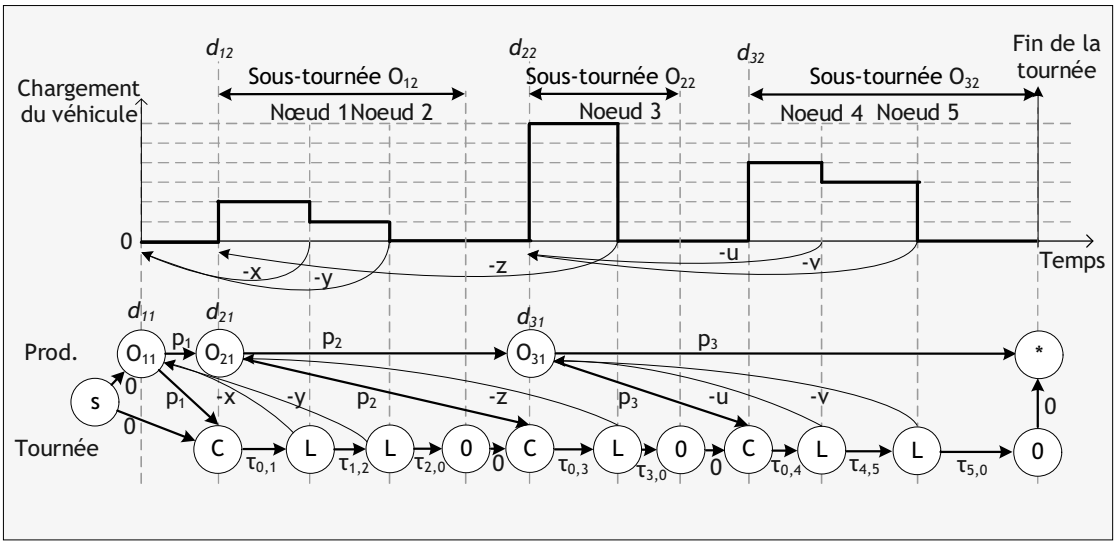


Figure 2-4. Représentation détaillée de la solution au cours de temps

La contrainte de périssabilité impose par définition une durée maximale entre la fin de la production d'un produit et de sa livraison. Pour modéliser cette contrainte, l'utilisation de time-lags maximaux est usuelle. Les time-lags maximaux sont représentés par des arcs de coût négatif entre le début de l'opération de production et l'opération de livraison. Dans le cas général, ou A et B sont deux opérations ($d_A$et d_B respectivement la date de début de l'opération A et B) avec $d_A \leq d_B$. L'expression d'une contrainte liée à un time-lags maximal de valeur tl_{max} peut été traduit par l'équation : $d_B \leq d_A + tl_{max}$. Dans ce cas, tl_{max} est la durée maximale entre le début de l'opération A et le début de

l'opération B. Les time-lags maximaux peuvent aussi être traduits sous forme de fenêtres de temps sur les nœuds d'un graphe correspondants à des opérations. En reprenant l'exemple précédent, la fenêtre de temps associée à l'opération B est de la forme]-∞ ; $d_A + tl_{max}$].

Sur la Figure 2-4, une fenêtre de temps de la forme]-∞ ; d_{11}+ y] est définie pour le nœud client numéro 2, sur lequel on effectue la deuxième livraison de la tournée, avec un arc de coût – y. Ce time-lag maximal impose une durée maximale à ne pas dépasser entre la date de début de l'opération de production O_{11}et la date d'arrivée du véhicule sur le nœud 2 pour la livraison de la demande.

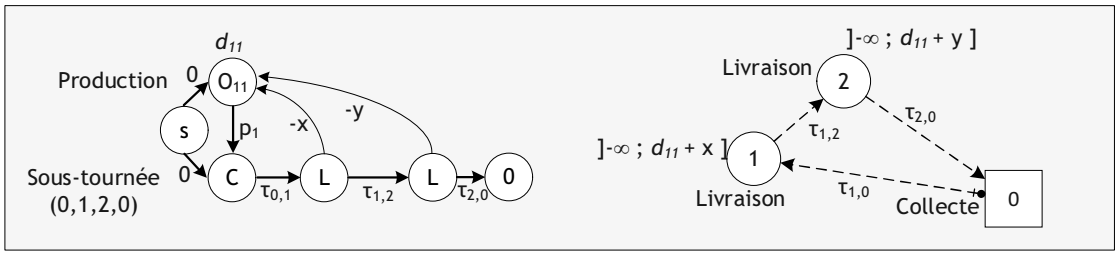


Figure 2-5. Lien entre contrainte de périssabilité et fenêtre de temps, exemple sur la tournée (0,1,2,0)

Ces time-lags maximaux modélisent des fenêtres de temps sur les nœuds modélisant les livraisons associées aux clients comme le montre la Figure 2-5 avec la première sous-tournée. La date de livraison maximale (sur un client donné) dépend à la fois de la date de début (et donc de la date de fin) de l'opération de production dans laquelle les produits (du client concerné) ont été fabriqués, de la durée de l'opération de production, et de la durée de vie des produits en question.

De la même façon des time-lags maximaux sont présents sur les autres nœuds avec les valeurs, – x, – z, – u et – v pour les nœuds de livraison associés aux clients 1, 3, 4 et 5.

Deux autres types d'arcs sont présents dans le graphe de la Figure 2-4, les arcs conjonctifs entre les opérations d'un même job, avec un coût $\tau_{i,j}$ permettant de définir une durée minimale entre deux nœuds. Cette durée correspond au plus court chemin (en temps) entre les deux nœuds. Les ars disjonctifs situés entre les jobs (entre les opérations de production et entre les opérations de transport) permettent de définir l'ordre des jobs.

Les time-lags maximaux peuvent être explicités, pour chacun des nœuds clients. La contrainte de durée de vie du produit impose une durée maximale $B = 7$ entre la fin de production d'un produit et sa livraison. Cette valeur est constante puisque le problème ne transport ne concerne qu'un seul type de produits. Cependant, l'expression d'une time-lag maximal s'effectue entre le début de deux opérations. Par conséquent, pour obtenir les time-lags maximaux, il faut prendre en compte en plus de la durée de vie du produit, la durée de l'opération de production associée.

Les valeurs des time-lags maximaux sur la Figure 2-4 peuvent donc être explicitées, et sont données par les expressions suivantes : $x = y = p_1 + B$, $z = p_2 + B$ et $u = v = p_3 + B$. Pour obtenir les fenêtres de temps, il faut ajouter à cette valeur la date de début de l'opération de production associée.

Le Tableau 2-2 illustre la solution associée à l'exemple de la Figure 2-1 avec les valeurs numériques de l'énoncé. L'ensemble de ces informations permet de calculer toutes les dates et les fenêtres de temps

Tableau 2-2
Exemple de solution du PTSP avec un véhicule

	Sous-tournée O_{12}^1					Sous-tournée O_{22}^1				Sous-tournée O_{32}^1				
	O_{11}	0	1	2	0	O_{21}	0	3	0	O_{31}	0	4	5	0
Date de début	0					2				9				
Date de fin	2					8				13				
Fenêtre de temps		]−∞;9]		]−∞;9]			]−∞;15]				]−∞;20]		]−∞;20]	
Date de départ		2	5	7			10	12			14	17	20	
Date d'arrivée			5	7	10			12	14			17	20	21

Les time-lags maximaux énoncés dans le Tableau 2-2 peuvent être simplifiés (voir Figure 2-6). En effet, au sein d'une sous-tournée, les dates d'arrivée du véhicule sur les nœuds de livraison augmentent avec la position au sein de la sous-tournée. Par conséquent, si la date de livraison d'un client de la sous-tournée respecte la fenêtre de livraison, toutes les livraisons effectuées en amont dans la sous-tournée respectent la fenêtre de livraison. Dans la Figure 2-6, la sous-tournée G avec trois nœuds de livraison comprend trois time-lags maximaux pour modéliser la durée de vie du produit. Ces trois time-lags maximaux concernent les fenêtres de temps sur le premier, le deuxième et le dernier point de livraison.

Dans le cas du PTSP à un produit, la prise en compte d'un unique time-lags maximal sur le dernier nœud de livraison est suffisante pour assurer le respect de la durée de vie du produit. Comme vu précédemment, ce time-lags maximal dépendra donc de la durée de vie du produit, B et de la durée de l'opération de production du produit en question (p_1 dans la Figure 2-6).

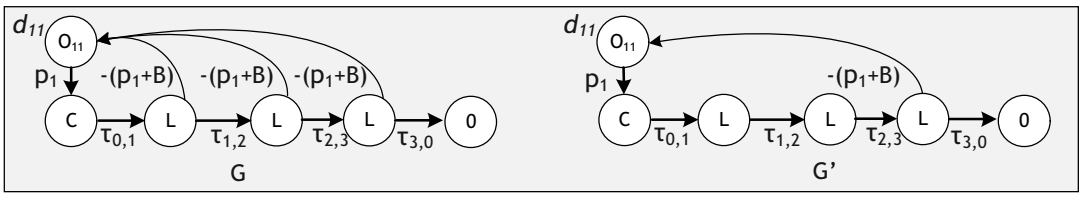


Figure 2-6. Simplification de la modélisation des time-lags maximaux

Le problème du PTSP à un véhicule, avec la contrainte de durée de vie sur le produit peut se modéliser comme un problème de flow shop à deux machines avec des time-lags maximaux. Ce problème est NP-difficile au sens fort (Fondrevelle *et al.*, 2004). Cependant, dans le cas où la contrainte sur les times-lags maximaux n'est pas prise en compte, le problème se ramène à un problème de flow shop classique, et dans le cas où

la contrainte sur les times-lags maximaux traduit une contrainte de no-wait entre les deux machines, le problème se ramène à un problème de flow shop avec no-wait. Le problème de flow shop classique et le problème de flow shop avec no-wait peuvent être résolus de façon optimale en temps polynomial avec respectivement l'algorithme de (Johnson, 1954) ou (Gilmore et Gomory, 1964).

La Figure 2-7 illustre les trois types de contrainte (pour les flow shop à deux machines) qui peuvent être utilisées pour la résolution du PTSP. Dans le premier cas, aucune contrainte additionnelle n'est rattachée au problème (pas de time-lags). Le début des opérations de transport doit être planifié après la fin des opérations de production : $d_{12} \geq d_{11} + p_1 = f_{11}$.

Cette contrainte reste vraie dans les deux autres cas (no-wait et time-lags maximaux). Dans les cas deux et trois, des time-lags maximaux sont exprimés, imposant une durée maximale tl_{max} entre le début des opérations de production et le début des opérations de transport : $d_{12} \leq d_{11} + tl_{max}$. Dans le deuxième cas (no-wait), cette durée maximale est égale à $tl_{max} = p_1$ et donc $d_{12} = d_{11} + p_1 = f_{11}$. Dans le troisième cas, correspondant au PTSP avec un produit périssable de durée de vie B, on doit livrer le dernier client ($\tau_{.,0}$ durée du transport entre le dernier client et le lieu de production) au maximum B unité de temps après la fin de l'opération de production : $d_{12} + t_1 - \tau_{.,0} \leq d_{11} + p_1 + B = f_{11} + B$. Par conséquent, $tl_{max} = p_1 + B - t_1 + \tau_{.,0}$.

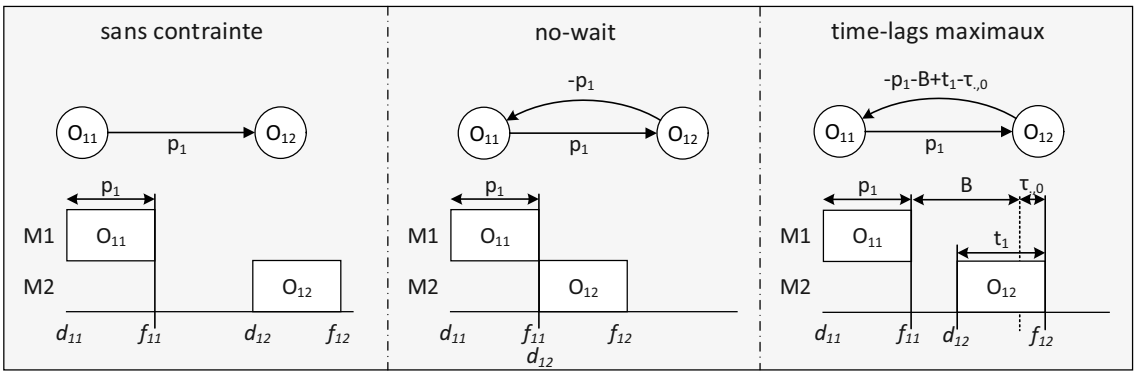


Figure 2-7. Illustration des 3 types de contraintes pour les flow shop à deux machines

2.2.3 Approche de résolution proposée par (Geismar et al., 2008) : description et analyse du contexte

L'approche de résolution du PTSP proposée par (Geismar *et al.,* 2008) est constituée de trois étapes (Figure 2-8). Leur première étape ne porte que sur le transport dans le PTSP et permet la création de la tournée du véhicule avec les sous-tournées qui la composent. Suite à cela, la deuxième étape permet de modéliser le problème du PTSP en problème de flow shop à deux machines avec des time-lags maximaux. Cette modélisation favorise l'utilisation d'algorithmes et d'outils tels que le graphe disjonctif au sein de la troisième étape pour résoudre le problème du PTSP.

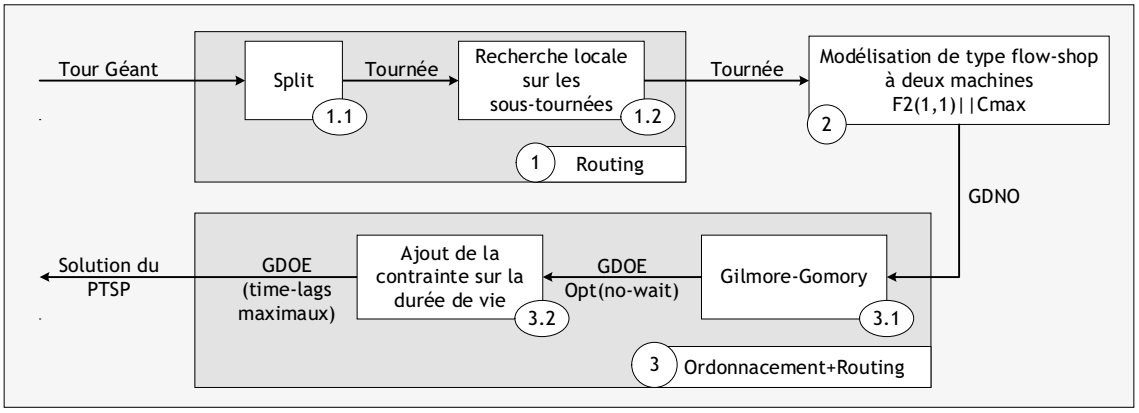


Figure 2-8. Schéma de la résolution du PTSP proposée par (Geismar et al., 2008).

Pour résoudre le problème du PTSP, (Geismar *et al.,* 2008) proposent une résolution basée sur une représentation indirecte de la solution avec un tour géant. Un tour géant est une permutation de l'ensemble des clients à livrer dans le problème du PTSP. Ce tour géant permet la création d'un ensemble de tournées (Beasley, 1983) grâce à la méthode « route-first and cluster-second ». Ce principe a été repris pour le PTSP par (Geismar *et al.,* 2008) avec l'utilisation d'un algorithme de type SPLIT directement adapté. Pour le PTSP, le SPLIT permet à partir d'un tour géant de définir les opérations de transport qui vont minimiser la durée totale du transport en respectant la capacité du véhicule et la durée de vie du produit. En modifiant l'ordre relatif des clients au sein des sous-tournées grâce à des mouvements 2-opt, la durée totale du transport peut être réduite. Une fois ces deux sous-étapes réalisées (étape 1 de la Figure 2-8) et grâce à la définition du problème, les opérations de transport ainsi définie permettent de définir les opérations de production.

En connaissant, les opérations de transport et de production (composition et durée), le problème du PTSP peut se ramener à un problème de flow shop à deux machines avec time-lags maximaux (étape 2 de la Figure 2-8). Ce problème peut être modélisé à l'aide d'un graphe disjonctif non orienté (GDNO) (Figure 2-9).

La résolution du problème de flow shop à deux machines avec time-lags maximaux est effectuée en 3 sous-étapes par (Geismar *et al.,* 2008) (étape 3 de la Figure 2-8). (Geismar *et al.,* 2008) utilisent l'algorithme de (Gilmore et Gomory, 1964) pour résoudre de façon optimale le problème de flow shop en no-wait (étape 3.1 de la Figure 2-8). Ensuite, en relâchant cette contrainte (étape 3.2 de la Figure 2-8), puis en ajoutant la contrainte sur la durée de vie du produit (étape 3.3 de la Figure 2-8), ils obtiennent une solution du PTSP.

La Figure 2-9 illustre les étapes 2 à 3.2 de la Figure 2-8 avec les différents graphes disjonctifs. L'étape 2 de modélisation permet de définir un graphe disjonctif non orienté mais pondéré (GDNO). Suite à l'exécution de Gilmore-Gomory sur le GDNO avec des time-lags pour la contrainte de no-wait, les disjonctions sont arbitrées, le GDO en no-wait est dès lors connu. Enfin, l'ajout de la contrainte sur la durée de vie du produit permet la modification des time-lags sur le graphe disjonctif pour obtenir le dernier graphe à évaluer, le GDO avec des time-lags maximaux. Pour prendre en compte la durée de vie du produit, les time-lags maximaux sont exprimés entre la date de début d'une opération de production et la date de début de l'opération de transport associée. Le poids de cet arc

doit prendre en compte la durée de l'opération de production p_i, la durée de vie du produit B, la durée de l'opération de transport t_i et la durée du trajet entre le dernier client à livrer et le lieu de production $\tau_{.,0}$.

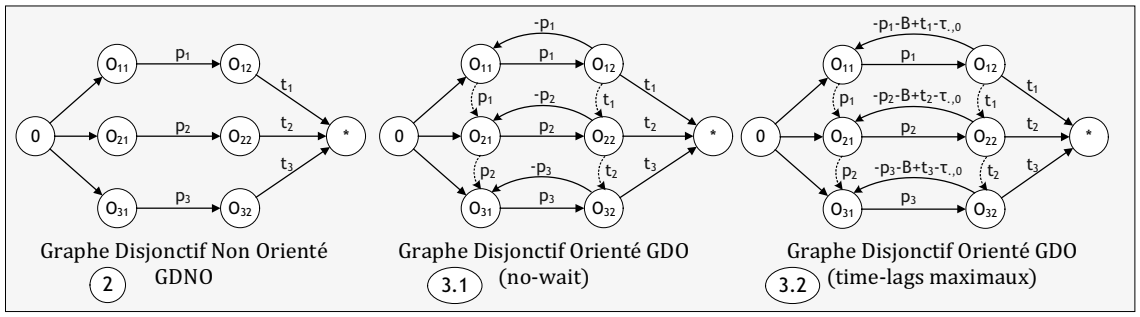


Figure 2-9. Les modélisations du flow shop à deux machines sous forme de graphe disjonctif (étapes 2, 3.1 et 3.2).

Le passage d'une solution du flow shop en no-wait, à une solution du flow shop avec la contrainte de durée de vie consiste à décaler les opérations de production et de transport vers la gauche sur le diagramme de Gantt, dans la limite des contraintes imposées par la durée de vie du produit (voir Figure 2-10).

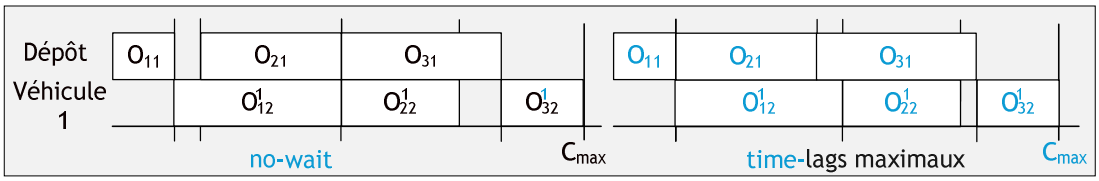


Figure 2-10. Comparaison entre une solution en no-wait et une solution avec les time-lags maximaux

Le problème du PTSP à un véhicule présenté dans cette partie a révélé plusieurs points caractéristiques avec notamment des time-lags maximaux particuliers et l'utilisation d'un graphe disjonctif. La partie suivante permet de définir le problème du PTSP sous forme de programme linéaire permettant d'exprimer toutes les contraintes du problème.

2.3 Formalisation linéaire : proposition

La formalisation linéaire présentée ici repose sur 4 étapes permettant de définir complètement le problème du PTSP. La première étape est la constitution des jobs (opérations de production et de transport), l'ensemble des clients est divisé en sous-ensembles. La deuxième étape garantie le respect des contraintes liées au problème de tournées de véhicules. La troisième étape définit les disjonctions sur la production et le transport. Enfin la quatrième étape complète la définition du problème avec la durée de vie du produit, les contraintes conjonctives entre la production et le transport ainsi que la définition du makespan.

Les données du problème dans la formulation reprennent les notations présentées précédemment. On introduit plusieurs variables binaires :

- y_{bi} qui vaut 1 si le client i appartient au job b ;

- h_b qui vaut 1 si le job b existe, c'est-à-dire si il contient au moins un client ;
- x_{ij}^b qui vaut 1 si le client i et le client j appartiennent au job b et si le client i précède immédiatement le client j ;
- z_{bc} qui vaut 1 si le job b précède le job c.

On introduit également la variable q_i^- qui est égale à la quantité de produit dans le véhicule après la livraison du client i. Enfin, on considère une constante suffisamment grande que l'on note H.

Le programme linéaire P (Figure 2-11) fait apparaître les contraintes suivantes :
(1) objectif ;

Étape 1 :
(2) cette contrainte assure que chaque client i appartient à un job b ;
(3) cette contrainte assure que les demandes clients dans un job b ne dépassent pas la capacité des véhicules ;
(4) permet de définir l'existence d'un job b ;
(5) permet de définir la durée de l'opération de production associée à un job b ;

Étape 2 :
(6) cette contrainte assure l'existence des transports d'un job b ;
(7) et (8) ces contraintes assurent que chaque client i soit desservi une et une seule fois ;
(9) et (10) ces contraintes assurent que pour chaque job b existant, l'opération de transport associée commence et se termine au dépôt ;
(11) et (12) ces contraintes assurent que chaque client i appartenant à un job b appartienne à l'opération de transport du job b ;
(13) et (14) ces contraintes empêchent les sous-tours lors de la création des tournées (contrainte classique MTZ) ;
(15) permet de définir la durée de l'opération de transport associé à un job b ;
(16) permet de définir la durée de l'opération de transport associé à un job b du dépôt au dernier client (le retour au dépôt n'est pas pris en compte) ;

Étape 3 :
(17) et (18) ces contraintes assurent le bon ordonnancement des opérations de production avec le respect des disjonctions ;
(19) et (20) ces contraintes assurent le bon ordonnancement des opérations de transport avec le respect des disjonctions ;

Étape 4 :
(21) cette contrainte assure le bon ordonnancement des opérations de production et de transport entre elles avec le respect des conjonctions ;
(22) et (23) ces contraintes assurent que l'ordonnancement respecte la durée de vie du produit ;
(26) correspond à l'expression classique du makespan : C_{max} qui donne la date à laquelle toutes les opérations de transport sont terminées ;

Définition des variables :
(25), (26), (27) et (28) les variables y_{bi}, h_b, x_{ij}^b et z_{bc} sont binaires ;
(29) et (31) les durées p_b, t_b, tt_b et les dates d_{j1}, d_{j2}sont des réels positifs ou nuls ;
(30) les quantités q_i^- sont des entiers positifs ou nuls.

$$
P:\begin{cases}
Minimiser\ C_{max} & & (1)\\
\forall i \in E & \sum_{b\in J} y_{bi} = 1 & (2)\\
\forall b \in J & \sum_{i\in E} y_{bi}\,.\,q_i \le C & (3)\\
\forall b \in J, \forall i \in E & y_{bi} - h_b \le 0 & (4)\\
\forall b \in J & p_b - \sum_{i\in E} y_{bi}\,.\,q_i.r = 0 & (5)\\
\forall (i,j) \in E^2, i \neq j, \forall b \in J & x_{ij}^b - h_b \le 0 & (6)\\
\forall j \in E & \sum_{b\in J} \sum_{\substack{i\in E\cup\{0\}\\ i\neq j}} x_{ij}^b = 1 & (7)\\
\forall i \in E & \sum_{b\in J} \sum_{\substack{j\in E\cup\{0\}\\ j\neq i}} x_{ij}^b = 1 & (8)\\
\forall b \in J & h_b - \sum_{j\in E} x_{0j}^b = 0 & (9)\\
\forall b \in J & h_b - \sum_{i\in E} x_{i0}^b = 0 & (10)\\
\forall i \in E, \forall b \in J & y_{bi} - \sum_{\substack{j\in E\cup\{0\}\\ j\neq i}} x_{ij}^b = 0 & (11)\\
\forall i \in E, \forall b \in J & y_{bi} - \sum_{\substack{j\in E\cup\{0\}\\ j\neq i}} x_{ji}^b = 0 & (12)\\
\forall b \in J, \forall (i,j) \in E^2, i \neq j & q_j^- - q_i^- + x_{ij}^b.C \le C - q_j & (13)\\
\forall b \in J, \forall (i,j) \in E^2, i \neq j & q_i^- \le C - q_i & (14)\\
\forall b \in J & t_b - \sum_{i\in E\cup\{0\}} \sum_{\substack{j\in E\cup\{0\}\\ j\neq i}} x_{ij}^b\,.\,\tau_{ij} = 0 & (15)\\
\forall b \in J & tt_b - t_b + \sum_{i\in E} x_{i0}^b.\tau_{i0} = 0 & (16)\\
\forall (b,c) \in J^2, b \neq c & d_{b1} + p_b - d_{c1} + z_{bc}.H \le H & (17)\\
\forall (b,c) \in J^2, b \neq c & d_{c1} + p_c - d_{b1} - z_{bc}.H \le 0 & (18)\\
\forall (b,c) \in J^2, b \neq c & d_{b2} + t_b - d_{c2} + z_{bc}.H \le H & (19)\\
\forall (b,c) \in J^2, b \neq c & d_{c2} + t_c - d_{b2} - z_{bc}.H \le 0 & (20)\\
\forall b \in J & d_{b1} + p_b - d_{b2} \le 0 & (21)\\
\forall b \in J & d_{b2} - d_{b1} - p_b - tt_b \le B & (22)\\
\forall b \in J & t_{tb} \le B & (23)\\
\forall b \in J & d_{b2} + t_b - C_{max} \le 0 & (24)\\
\forall b \in J, \forall i \in E & y_{bi} \in \{0,1\} & (25)\\
\forall b \in J & h_b \in \{0,1\} & (26)\\
\forall b \in J, \forall (i,j) \in E^2, i \neq j & x_{ij}^b \in \{0,1\} & (27)\\
\forall (b,c) \in J^2, b \neq c & z_{bc} \in \{0,1\} & (28)\\
\forall b \in J & p_b \in \mathbb{R}, t_b \in \mathbb{R}, tt_b \in \mathbb{R} & (29)\\
\forall i \in E & q_i^- \in \mathbb{N} & (30)\\
\forall b \in J & d_{b1} \in \mathbb{R}, d_{b2} \in \mathbb{R} & (31)
\end{cases}
$$

Figure 2-11. Programme linéaire pour le PTSP

2.4 Résolution du problème avec un véhicule : proposition

2.4.1 Codage indirect des solutions

La résolution du problème du PTSP à un véhicule repose sur plusieurs points-clés liés à l'introduction de deux espaces de recherche représentés sur la Figure 2-12.

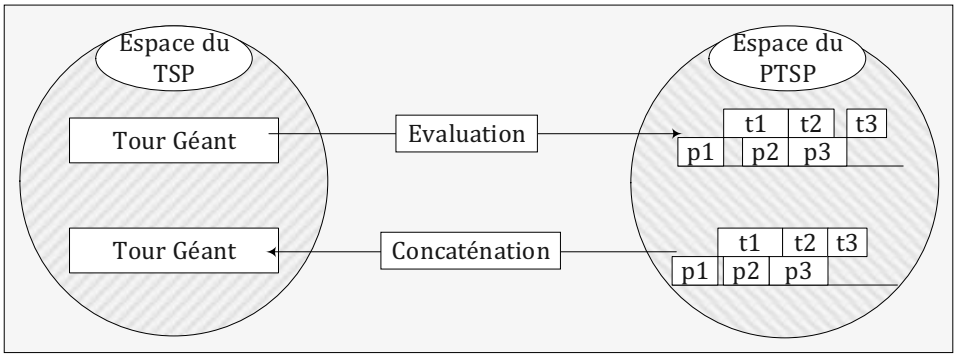


Figure 2-12. Construction d'une solution pour le PTSP à un véhicule

L'approche de résolution est basée sur un codage indirect des solutions. Le codage des solutions est réalisé à partir de tours géants, c'est-à-dire une suite ordonnée de clients. Classiquement, ceci correspond à des solutions du Traveling Salesman Problem (TSP) représentées sous forme de vecteurs d'entiers de taille n, représentant les clients du problème. Un tour géant peut alors être transformé via une fonction « d'évaluation » en solution du PTSP. Le deuxième espace de recherche contient les solutions du PTSP avec à la fois l'ordonnancement de la production et des sous-tournées du véhicule. La fonction d'évaluation permet d'obtenir une solution du PTSP à partir d'un tour géant. Cette fonction d'évaluation définit une application surjective mais non injective, puisqu'il est tout à fait possible qu'une solution du PTPSP puisse être obtenue à partir de plusieurs tours géants différents au sens lexicographique.

Dans le cas général, les fonctions de codage peuvent se décliner en trois types (Figure 2-13) comme le souligne (Cheng *et al.*, 1996) :

- le codage 1-to-n, où un élément de codage est associé à plusieurs solutions ;
- le codage n-to-1, où plusieurs éléments de codage sont associés à une solution ;
- le codage 1-to-1, où chaque élément de codage est associé à une unique solution, ce type de codage est le plus recherché.

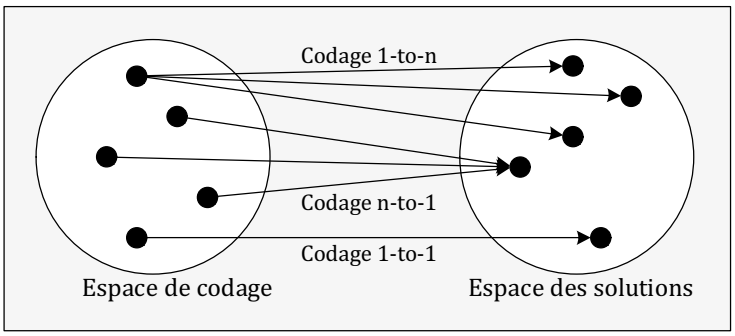


Figure 2-13. Les types de codages.

Le cas idéal est de disposer d'une fonction de type 1-to-1. Pour les problèmes d'ordonnancement, les vecteurs de Bierwirth classiquement utilisés sont de type n-to-1, et les tours géants utilisés sur les problèmes de tournés sont aussi de cette famille. La fonction proposée n'échappe pas à la règle et appartient aux fonctions de type n-to-1.

2.4.2 La fonction de codage

La fonction de codage qui permet d'associer à un tour géant une solution du PTSP est composée de trois étapes principales (Figure 2-14). Une première étape qui résout un problème de « routing » pour déterminer à la fois les caractéristiques des opérations de transport et des opérations de production, une étape d'intégration qui permet de modéliser le problème en un flow shop à deux machines et une troisième étape dans laquelle des algorithmes spécifiques du flow shop à deux machines sont utilisés pour obtenir une résolution intégrée du problème.

La résolution du PTSP repose sur six sous-étapes :

- une méthode SPLIT qui permet d'obtenir les sous-tournées du véhicule à partir d'un tour géant (étape 1.1, Figure 2-14) ;
- une recherche locale sur les sous-tournées pour améliorer la tournée du véhicule (étape 1.2, Figure 2-14) ;
- un outil de modélisation permettant de modéliser le problème en $F2(1,1)|tl_{max}|C_{max}$ (étape 2, Figure 2-14) ;
- une méthode d'ordonnancement permettant d'obtenir une solution du $F2(1,1)||C_{max}$ ou du $F2(1,1)|no\text{-}wait|C_{max}$ (étape 3.1, Figure 2-14) ;
- l'ajout de la contrainte sur la durée de vie du produit, dans le but d'obtenir une solution du PTSP (étape 3.2, Figure 2-14) ;
- une méthode de concaténation pour transformer une solution du PTSP en tour géant (étape 4, Figure 2-14).

Ces différentes étapes sont développées dans les sous-parties suivantes de cette section.

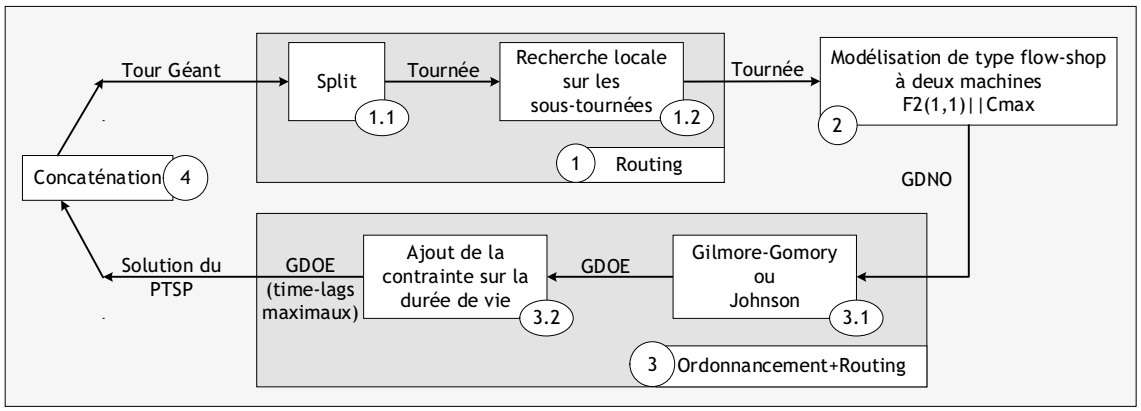


Figure 2-14. Schéma de résolution du PTSP

2.4.3 Construction des sous-tournées du PTSP : le SPLIT

Les tours géants sont évalués grâce à un découpage en sous-tournées via une procédure SPLIT. L'originalité de cette approche réside dans le fait qu'en déterminant les sous-tournées, on détermine aussi simultanément les opérations de production, puisque la définition du problème impose que les clients affectés à une même opération de production doivent également être affectés à une même opération de transport.

Cette procédure minimise la durée totale des sous-tournées sous les contraintes liées à la capacité du véhicule et à la durée de vie des produits transportés. Dans cette optique, la méthode SPLIT proposée ici s'inspire fortement du SPLIT introduit par (Beasley, 1983) en ajoutant la contrainte sur la durée de vie du produit, ce qui va limiter la taille des tournées.

Le SPLIT utilise un algorithme de plus court chemin appliqué sur un graphe auxiliaire $G = (X, A)$ où X représente les $n + 1$ nœuds numérotés de 0 à n. Le nœud 0 est un nœud factice, tandis que les nœuds $1 \dots n$ correspondent au tour géant $TG = (\sigma_1, \dots, \sigma_n)$. L'arc (i, j) appartient à A, l'ensemble des tournées faisables, si la tournée entre σ_{i+1} et σ_j inclus respecte la capacité du véhicule et la durée de vie du produit. C'est-à-dire, si $\sum_{k=i+1}^{j} q_{\sigma_k} \leq Q$ et si $\tau_{0,\sigma_{i+1}} + \sum_{k=i+1}^{j-1} \tau_{\sigma_k,\sigma_{k+1}} \leq B$.

Un label initial est placé sur le nœud *0*, les labels sont ensuite propagés de nœud en nœud dans le graphe G en utilisant les arcs A, le meilleur label sur le nœud n est gardé comme étant la meilleure solution de notre sous-problème. Un label est composé d'une valeur, pour le coût de la solution partielle, et d'un couple pour définir le label père.

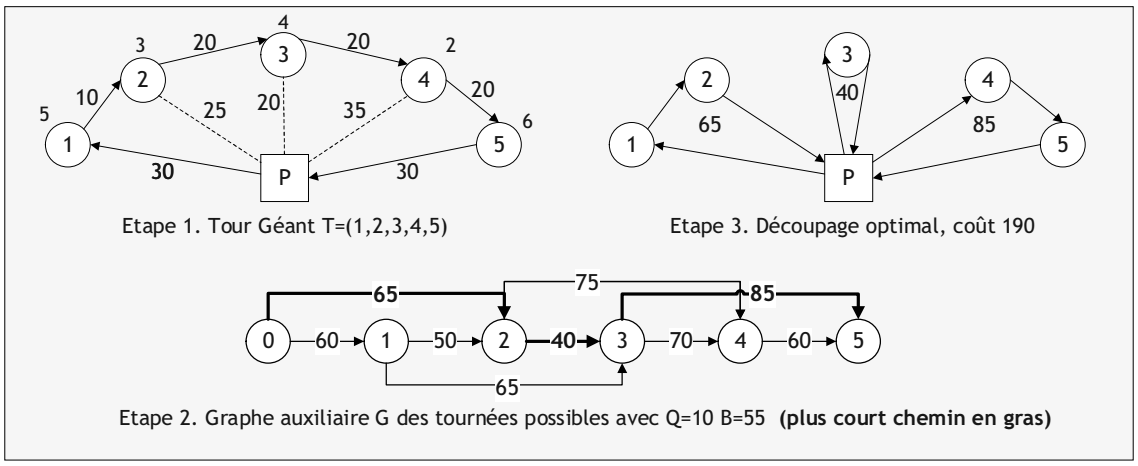


Figure 2-15. Exemple de découpage d'un tour géant en tournées

2.4.4 Recherche locale sur les sous-tournées

La recherche locale sur les sous-tournées est effectuée en utilisant plusieurs voisinages classiques du VRP pour améliorer la solution initiale du PTSP, à savoir le mouvement 2-Opt sur une sous-tournée, le mouvement 2-Opt entre deux sous-tournées, un échange au sein d'une sous-tournée ou entre deux sous-tournées (voir Lacomme *et al.*, 2001 et Prins 2004). Pour chaque mouvement effectué, la faisabilité des nouvelles sous-tournées doit être vérifiée à la fois par rapport à la capacité du véhicule et à la durée de vie.

2.4.5 Modélisation

L'étape 1 (Figure 2-14) détaillée dans les deux sous-parties précédentes a permis la création des sous-tournées associées au problème du PTSP. Or, grâce à la définition du problème, chaque sous-tournée permet de définir une opération de transport. Le nombre d'opérations de transport est donc déjà connu. De plus, étant donné les caractéristiques du PTSP, les clients livrés dans une même opération de transport vont avoir leurs demandes produites dans la même opération de production. Le nombre d'opérations de production est égal au nombre d'opérations de transport. Par conséquent, le nombre de jobs à ordonnancer est totalement défini.

L'utilisation d'un graphe disjonctif (Roy et Sussmann, 1964) peut été utilisé pour modéliser les jobs avec les opérations de production et de transport. La Figure 2-16 illustre le cas d'utilisation d'un graphe disjonctif pour modéliser trois jobs, et donc trois opérations de production et trois opérations de transport, un job étant constitué d'une opération de production et d'une opération de transport.

De plus, la durée de l'opération de transport est égale à la durée de la sous-tournée associée, cette information est par conséquent connue dès la fin de l'étape 1 sur le routing. La durée de l'opération de production est également connue à l'issue de cette étape, puisqu'elle est donnée par la somme des temps de production des demandes des clients livrés dans la sous-tournée associée. Toutes ces informations permettent la modélisation du problème à l'aide d'un graphe disjonctif non-orienté pondéré. Les nœuds du graphe modélisent les opérations de production et les opérations de transport avec deux nœuds supplémentaires pour modéliser le début et la fin du problème (notés respectivement 0 et $*$). Les arcs permettent de modéliser les contraintes temporelles entre les opérations.

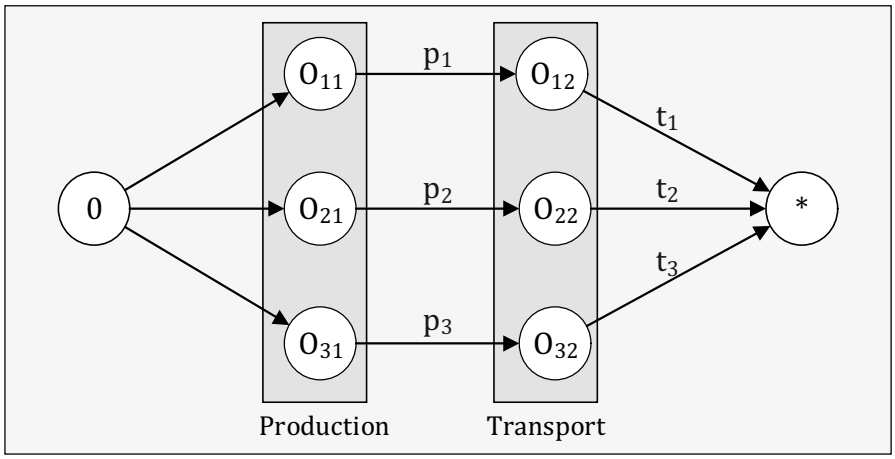


Figure 2-16. Modélisation de trois jobs sous forme de graphe disjonctif non orienté pondéré

Pour obtenir une solution du PTSP à partir de ces premières informations, il faut arbitrer les disjonctions sur les opérations de production et sur les opérations de transport. Il faut donc rechercher une séquence d'opérations de production et une séquence d'opérations de transport. Dans le cas du PTSP, les deux séquences doivent être les mêmes, puis les opérations de production et de transport doivent être réalisées dans le

même ordre. Dans le cas général des problèmes d'ordonnancement d'atelier le problème du PTSP peut donc être modélisé comme un flow shop à deux machines, puisqu'au sein de chaque job l'ordre de visite des deux machines est identique.

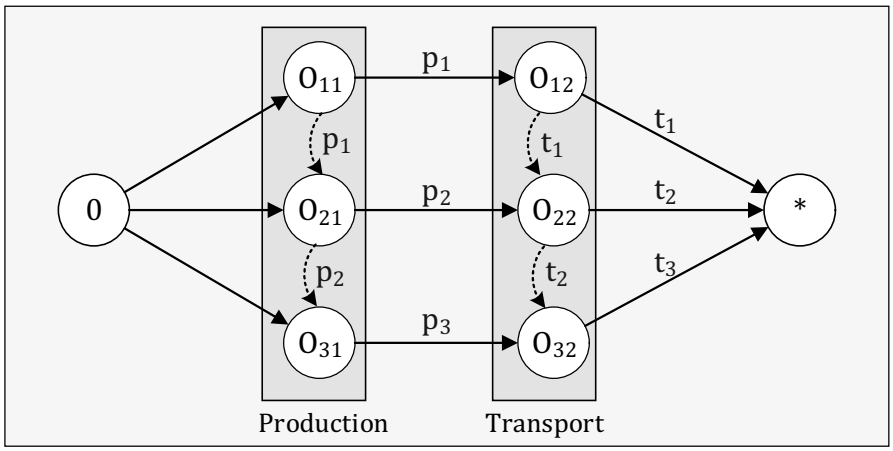


Figure 2-17. Modélisation de trois jobs sous forme de graphe disjonctif orienté pondéré

La Figure 2-16 illustre un choix d'arbitrage des disjonctions sur la production et le transport. Dans cet exemple les opérations sont réalisées dans l'ordre croissant, *i.e.* les opérations de production sont réalisées dans l'ordre (O_{11}, O_{21}, O_{31}) et les opérations de transport dans le même ordre (O_{12}, O_{22}, O_{32}).

Pour définir une solution du PTSP, on impose la contrainte des time-lags maximaux dans la modélisation sous forme de graphe disjonctif. Les time-lags maximaux sont représentés par de nouveaux arcs entre les opérations de production et de transport d'un même job. Le poids de ces arcs permet le respect de la durée de vie du produit pour chaque job Figure 2-18.

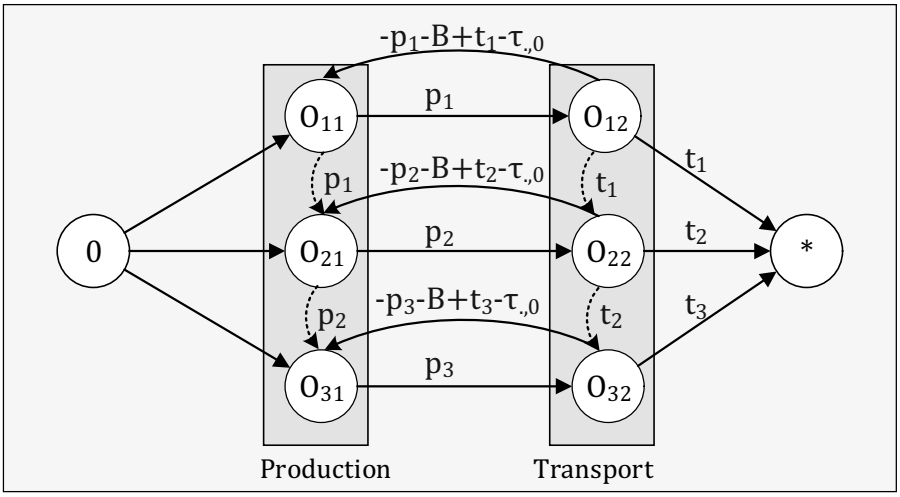


Figure 2-18. Modélisation de trois jobs sous forme de graphe disjonctif orienté pondéré avec des time-lags maximaux

Pour prendre en compte les times-lags maximaux dans la résolution du flow shop à deux machines, trois alternatives peuvent être explorées.

Dans le premier cas, on ramène le $F2(1,1)|tl_{max}|C_{max}$ à un $F2(1,1)|no\text{-}wait|C_{max}$, puis on conserve les disjonctions obtenues pour résoudre le $F2(1,1)|tl_{max}|C_{max}$. Dans ce premier cas, la contrainte associée aux time-lags maximaux est sur-contrainte.

Dans le deuxième cas, la contrainte associée aux time-lags maximaux est totalement relaxée, on résout alors le $F2(1,1)||C_{max}$. De la même façon, en conservant les disjonctions obtenues et en ajoutant les time-lags maximaux, une solution du PTSP peut être définie.

Ces deux premiers cas ont pour avantage d'utiliser des algorithmes exacts pour la première partie de la résolution. Ces algorithmes sont présentés dans la partie 2.4.6.

Dans le troisième cas, on envisage de résoudre directement le problème du $F2(1,1)|tl_{max}|C_{max}$. Cette alternative est développée dans le chapitre 3.

2.4.6 Construction d'une solution du PTSP à partir des sous-tournées

Un fois les sous-tournées du PTSP constituées, la durée et la composition des opérations de production et de transport sont alors connues. Le problème peut alors être modélisé comme un flow shop à deux machines avec time-lags maximaux. Ce problème étant NP-difficile au sens fort, une résolution en deux étapes peut être proposée, avec une étape optimale suivie d'une étape sous-optimale. Deux alternatives peuvent été proposées pour effectuer ces étapes.

Dans un premier cas, on peut résoudre le problème du flow shop sur deux machines en no-wait puis ajouter à cette solution la contrainte sur les time-lags maximaux. L'algorithme de (Gilmore et Gomory 1964), qui a également été utilisé par (Geismar *et al.*, 2008) et détaillé par (Pinedo, 2008) peut être utilisé pour résoudre cette première étape. Cet algorithme résout de façon optimale le problème du flow shop à deux machines en no-wait, avec une complexité $O(n.\log(n))$.

On peut également résoudre le problème classique du flow shop sur deux machines pour ensuite ajouter à la solution la contrainte des time-lags maximaux. L'algorithme de (Johnson, 1954) permet d'obtenir l'ordonnancement optimal pour un flow shop à deux machine et cet algorithme a également une complexité en $O(n.\log(n))$.

Ces deux algorithmes présentent un grand intérêt car leur complexité est très bonne mais ont l'inconvénient d'être dédiés à un problème particulier. L'algorithme de (Gilmore et Gomory 1964) adapté au $F2(1,1)|no\text{-}wait|C_{max}$ peut sembler adapté dans le cas où la durée de vie du produit est très petite. À l'opposé, lorsque la durée de vie du produit est très grande, le problème du $F2(1,1)|tl_{max}|C_{max}$ tend alors vers le problème $F2(1,1)||C_{max}$.

2.4.6.1 L'algorithme de Gilmore et Gomory

Le but de l'algorithme Gilmore-Gomory (Gilmore et Gomory 1964) (Pinedo, 2008) est de minimiser le makespan en ordonnant les jobs, pour que le temps de l'opération de transport du job i soit proche du temps de l'opération de production du job $i+1$. L'algorithme débute avec les jobs triés par ordre croissant sur les temps des opérations de transport. À partir de cet ordre, des permutations soigneusement choisies sont

effectuées pour trouver le meilleur ordonnancement des jobs. L'algorithme a une complexité en $O(n \log n)$, puisqu'il consiste avant tout à exploiter des algorithmes de tri.

Algorithme 1 . Gilmore-Gomory

```
1.  procedure Gilmore-Gomory
2.  input parameters
3.    nj: the number of jobs
3.    P[]: array of the production operations duration of all jobs
3.    T[]: array of the transport operations duration of all jobs
4.  output parameters
5.    S: PTSP solution
6.  global parameter
7.    OP[]: order of the production operations array
8.    OT[]: order of the transport operations array
9.    M[]: permutation mappings array
10.   C[]: interchange costs array
11.   SC[]: smallest interchange costs array
12.   I[]: array to save the sequence of interchange
13.  begin
14.  | call initialization(P[], T[], nj, OP[], OT[], M[]) ;
15.  | if optimal_order(M[])=false then
16.  | | call compute_costs(OP[], OT[], nj, C[]);
17.  | | call select_costs(M[], C[], nj, SC[]);
18.  | | call find_interchange_sequence(M[],SC[],P[],T[],OP[],OT[], nj, I[]);
19.  | endif
20.  | S := call build_solution(P[], T[], nj, M[], I[], S);  //save the solution
21.  end
```

La procédure est détaillée dans Algorithme 1 et utilise plusieurs procédures :

- `initialization(P[], T[], nj, OP[], OT[], M[])`, permet de trier les opérations de production et les opérations de transport par durée croissante. Les indices des jobs correspondant aux opérations sont sauvegardés dans le tableau $OP[]$ pour les opérations de production et dans le tableau $OT[]$ pour les opérations de transport. Cette procédure permet de définir une fonction pour modéliser toutes les permutations entre les deux ordres. Grace à cette fonction, si $M[k] = l$, alors p_l est la $k^{ième}$ plus petite durée des opérations de production ;
- `optimal_order(M[])`, permet de tester l'optimalité de la solution initiale liée à la fonction de modélisation des permutations. L'ordre des jobs triés est optimal si le graphe $G(J, M)$ formé des nœuds modélisant les jobs et des arcs reliant un job j au job $k = M[j]$ est un cycle ;
- `compute_costs(OP[], OT[], nj, C[])`, calcul le coût lié à l'échange du début des deux arcs consécutifs $(j, M[j])$ et $(j + 1, M[j + 1])$. Ce coût est calculé pour $j = 1, \dots, nj$ grâce à la formule suivante : $C[j] = 2 \times max\big(0, min\big(t_{T[j+1]}, p_{M[j+1]}\big) - max\big(t_{T[j]}, t_{T[j]}\big)\big)$;
- `select_costs(M[], C[], nj, SC[])`, sélectionne les coûts les plus petits qui permettent de former un cycle ;
- `find_interchange_sequence(M[], SC[], P[], T[], OP[], OT[], nj, I[])`, cette procédure divise les coûts sélectionnés $SC[]$ en deux groupes. Les coûts appartiennent au groupe 1 si, $p_{M[j]} \geq t_{T[j]}$ et au groupe 2 sinon. Le groupe 1 est ensuite trié par indice décroissant et le groupe 2 par indice croissant. Les indices des deux groupe sont ensuite concaténés pour former le tableau $I[]$;

- `build_solution(P[], T[], nj, M[], I[], S)`, cette procédure permet de construire une solution du PTSP S à partir du tableau des permutations $I[]$ et de la fonction de modélisation des permutations $M[]$.

2.4.6.2 L'algorithme de Johnson

Le but de l'algorithme de Johnson est de minimiser le makespan.

Théorème de l'algorithme de Johnson

Dans un problème de flow shop à deux machines, un job i doit précéder un job j dans une séquence optimale si $min(A_i, B_j) \leq min(A_j, B_i)$, A_j et B_j étant les temps d'exécution du job j respectivement sur la machine A et B.

En d'autres termes, les jobs ayant les plus petits temps d'exécution sur la première machine seront exécutés en premier, et ceux qui ont les plus petits temps sur la deuxième machine seront exécutés à la fin. L'algorithme a une complexité en $O(n \log n)$, puisqu'il consiste avant tout à exploiter des algorithmes de tri.

Algorithme 2 . Johnson

```
1.  procedure Johnson
2.  input parameters
3.    nj: the number of jobs
3.    P[]: array of the production operations duration of all jobs
3.    T[]: array of the transport operations duration of all jobs
4.  output parameters
5.    S: PTSP solution
6.  global parameter
7.    F[]: fist jobs array
8.    L[]: last jobs array
9.    J[]: array to save the sequence of jobs to obtain the optimal solution
10.  begin
12.  | for i from 1 to nj do
13.  | | call compute_groups(P[], T[], nj, F[], L[]);
14.  | endfor
15.  | call sort_jobs(nj, F[], L[], J[]);
16.  | S := call build_solution(P[], T[], nj, J[], S);  //save the solution
17.  end
```

L'algorithme de Johnson est détaillé dans Algorithme 2 et utilise plusieurs procédures :

- `compute_groups(P[], T[], nj, F[], L[])`, permet de diviser les jobs en deux groupes. Les jobs i tel que $P[i] \leq T[i]$ sont ajoutés au tableau $F[]$, tandis que les jobs i tel que $P[i] > T[i]$ sont ajoutés au tableau $L[]$;
- `sort_jobs(nj, F[], L[], J[])`, effectue un tri sur les deux sous-tableaux, $F[]$ et $L[]$. Les jobs dans le tableau $F[]$ sont ordonnancés dans l'ordre croissant des durées des opérations de production. Les jobs dans le tableau $L[]$ sont ordonnancés dans l'ordre décroissant des durées des opérations de transport. La séquence optimale est stockée dans le tableau $J[]$. Elle est constituée de la concaténation des deux tableaux $F[]$ et $L[]$ précédemment triés ;
- `build_solution(P[], T[], nj, J[], S)`, cette procédure permet de construire une solution du PTSP S à partir du tableau des jobs $J[]$ donnant directement la séquence à ordonnancer.

2.4.7 Ajout de la contrainte sur la durée de vie

En supplément de l'algorithme utilisé pour l'étape 3.1, Figure 2-14, on impose la contrainte sur la durée de vie du produit.

L'algorithme de Gilmore-Gomory permet d'obtenir une solution du $F2(1,1)|no\text{-}wait|C_{max}$, cette solution est également une solution du $F2(1,1)|tl_{max}|C_{max}$, puisque la contrainte de no-wait est plus forte que la contrainte sur la durée de vie. Par conséquent, en ajoutant la contrainte sur la durée de vie la solution verra son makespan diminuer ou rester identique.

D'autre part, notons que la solution du $F2(1,1)||C_{max}$ obtenue grâce à l'algorithme de Johnson n'est pas toujours une solution pour le problème du PTSP. L'ajout de la contrainte de durée de vie est alors nécessaire pour obtenir une solution du PTSP, cette solution aura donc un makespan supérieur ou identique à celui de la solution sans contrainte.

2.4.8 Concaténation d'une solution du PTSP en tour géant

La procédure de concaténation (Concat.) permet de construire un tour géant à partir d'une solution du PTSP. En supprimant le dépôt de chaque tournée puis en concaténant celles-ci dans un vecteur, il est possible d'obtenir un nouveau tour géant.

2.5 Proposition d'un schéma général de résolution du PTSP : GRASP×ELS

Le schéma général de résolution est basé sur une métaheuristique de type GRASP×ELS permettant l'intégration des procédures présentées précédemment pour résoudre le problème du PTSP à un véhicule.

Le GRASP×ELS permet l'intégration des points-clés mentionnés précédemment :

- une recherche locale sur les tournées avec des opérations de type 2-Opt ou échange dans une ou plusieurs sous-tournées ;
- une méthode SPLIT pour la création des sous-tournées (opérations de transport) ;
- une méthode d'ordonnancement (Johnson ou Gilmore-Gomory) suivi de l'ajout de la contrainte de durée de vie ;
- une procédure de concaténation pour obtenir un tour géant à partir d'une solution du PTSP.

Le GRASP×ELS est une métaheuristique hybridée (Prins, 2009) à partir d'un GRASP (Greedy Randomized Adaptive Search Procedure) (Feo et Resende, 1995) avec une ELS (Evolutionary Local Search) (Wolf *et al.,* 2007, Prins, 2004) tirant avantage des deux méthodes. Le GRASP permet de fournir np solutions initiales, ces solutions sont ensuite améliorées grâce à une recherche locale. L'heuristique ELS quant à elle est une extension de ILS (Iterated Local Search) (Lourenço *et al.*, 2003) et permet l'exploration du voisinage de la solution courante. L'exploration de l'espace des solutions est effectuée par le GRASP permettant la diversification, et par l'ELS permettant l'intensification de l'exploration.

Comme cela a été montré dans de nombreux travaux (Duhamel *et al.*, 2013) (Lacomme *et al.*, 2013) ou encore (Chassaing *et al.*, 2014), la définition des différents éléments tels que les recherches locales, les heuristiques de construction, *etc*. dans l'un ou l'autre des espaces de recherche est un point très important. Ces définitions permettent à la fois d'obtenir des algorithmes simples mais aussi de réaliser une alternance entres les espaces. Pour le CARP, par exemple, cette alternance a été soulignée comme un élément fort dans l'efficacité des méthodes.

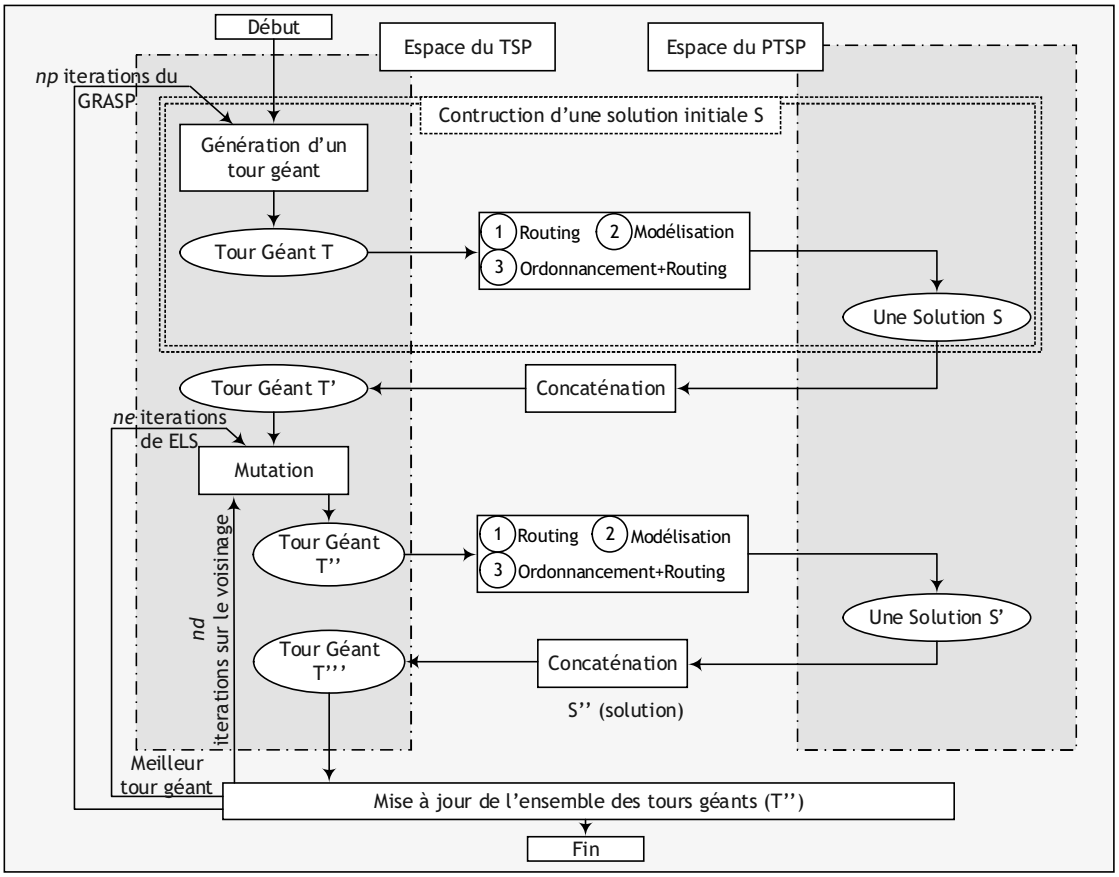


Figure 2-19. GRASP×ELS sur deux espaces de recherche.

Comme représenté sur la Figure 2-19, le GRASP×ELS est défini sur deux espaces de solution : les solutions du TSP avec des tours géants et les solutions du PTSP intégrant production et transport. L'efficacité de la méthode repose sur différents opérateurs (fonction d'évaluation, concaténation et mutation) défini sur des espaces solutions spécifiques pour favoriser l'exploration de la métaheuristique.

Algorithme 3 . GRASP× ELS for the PTSP

```
1.  procedure GRASP×ELS
2.   global parameters
3.    np: number of GRASP iterations (initial solutions)
4.    ne: maximal number of iterations per ELS
5.    nr: maximal number of iterations without improvement per ELS
6.    nd: number of diversifications (mutations)
7.    nl: number of local search on the routing
8.    ng: number of local search on the scheduling
9.   output parameters
10.   S*: best PTSP solution found
11.  begin
12.  | f* := ∞; O := Ø
13.  | for p := 1 to np do                    // GRASP loop
14.  | | S := call Generation_of_initial_solution ()
15.  | | T := call Concat (S)
16.  | | S := call Split (T)
17.  | | S := call Local_Search_on_Routing (S, nl)
18.  | | S := call Scheduling (S)
29.  | | T := call Concat (S)
20.  | | if (f(S) < f*) then f* := f(S); S* := S; // f : the cost of a solution
21.  | | endif
22.  | | i, r := 0
23.  | | while (i < ne) and (r < nr) do   // ELS loop
24.  | | | i := i + 1; f" := ∞
25.  | | | for j := 1 to nd do             // mutation loop
26.  | | | | T' := call Mutation (T)
27.  | | | | S' := call Split (T')
28.  | | | | S' := call Local_Search_on_Routing (S', nl)
29.  | | | | S' := call Scheduling (S')
30.  | | | | T' := call Concat (S')
31.  | | | | if (f(S') < f") then f" := f(S'); T" := T'; S" := S'; endif
32.  | | | endfor
33.  | | | if (f" < f*) then S*:= S"; endif // if a new best solution update S*
34.  | | | T := T";  // best ELS solution becomes the new initial solution
35.  | | endwhile
36.  | endfor
37.  end
```

L'Algorithme 3 présente en détail le schéma de résolution et est composé d'une boucle principale (entre les lignes 13 et 36) permettant la génération des *np* solutions initiales du GRASP. Chaque solution initiale est générée par la procédure `Generation_of_initial_solution()`. Cette solution est ensuite concaténée pour pouvoir appliquer la procédure SPLIT (line 16) et obtenir une solution avec les tournées du problème, cette solution est améliorée (line 17) par une procédure de recherche locale sur les tournées, `Local_Search_on_Routing(S, nl)`. Sur cette solution améliorée, la procédure `Scheduling()` génère une solution du PTSP (line 18) grâce à l'utilisation de l'algorithme de Gilmore-Gomory ou de Johnson suivi de l'ajout des contraintes liées à la durée de vie du produit. Si le coût $f(S)$ de cette solution S est inférieur au coût f^* de la meilleure solution trouvée S^* (line 20), S devient alors la meilleure solution. La boucle while entre les lignes 23 et 35 caractérise l'ELS et la boucle entre les lignes 25 et 32 permet la génération d'un voisinage de tours géants.

L'heuristique permettant de générer les solutions initiales du GRASP, `Generation_of_initial_solution()` utilise en fait deux heuristiques.

La première heuristique est utilisée avec la probabilité de 0.9, elle est basée sur une heuristique gloutonne. Cette heuristique permet de constituer les sous-tournées une par une. Chaque sous-tournée commence au dépôt, on vient ajouter un client dans cette sous-tournée suivant deux critères :

- on ajoute le client le plus proche de la localisation actuelle, celui minimisant le temps de transport ;
- si le chargement du véhicule est supérieur ou égal à $Q/2$ alors le prochain client à servir est sélectionné pour minimiser la distance au dépôt ;

La deuxième heuristique est utilisée avec la probabilité de 0.1, elle effectue un classement décroissant des clients sur les temps de transport. Cette procédure renvoie directement un tour géant.

2.6 Résultats numériques

Les instances introduites par (Geismar *et al.*, 2008) sont utilisées pour tester notre méthode par rapport à celles proposées par (Geismar *et al.*, 2008) et par (Karaoğlan et Kesen, 2017). Les instances et les résultats présentés dans ce chapitre sont disponibles sur la page :

http://fc.isima.fr/~vinot/Research/PTSP_Results.html

Les instances proposées par (Geismar *et al.*, 2008) sont caractérisées par plusieurs paramètres. Les caractéristiques des instances varient en fonction du nombre de clients, entre 40 et 50, et de la localisation des clients qui sont distribués uniformément dans des zones rectangulaire de taille 200×200, 300×300 ou 400×400. Cela permet la création de six jeux de données (DS). De plus, trois autres paramètres interviennent, le taux de production $r \in \{1,2,3\}$, la capacité du véhicule $Q \in \{300,600\}$ et la durée de vie du produit $B \in \{300,600\}$. Au total, cela définit 12 jeux de paramètres (SP) pour chaque jeu de données. L'ensemble représente 72 instances (Tableau 2-3).

Tableau 2-3
Caractéristiques des instances de (Geismar et al. 2008)

Nombre de clients	Demande des clients	Zones de répartition	Taux de production	Capacité du véhicule	Durée de vie du produit
40 / 50	U(100;300)	200 / 300 / 400	1 / 2 / 3	300 / 600	300 / 600

Dans les tableaux de résultats, deux temps sont donnés : le temps publié par les méthodes dans les articles ainsi que les temps normalisés *(scaled time)* par rapport à la machine utilisée dans ce chapitre. Cette normalisation est basée sur le nombre de Mflop/s des différents processeurs utilisés. Pour trouver le nombre de Mflop/s, les travaux de (Dongarra, 2014) ainsi que les travaux réalisés sur le LINPACK, disponibles sur http://www.roylongbottom.org.uk/linpackresults.htm, ont été utilisés. Elle reste imparfaite dans le sens où elle ne prend pas en compte la mémoire de chaque machine, ni d'autre facteurs comme le système d'exploitation, mais elle permet toutefois d'avoir des comparaisons plus objectives que celles obtenues en reportant directement les temps fournis par les auteurs.

Le Tableau 2-4 regroupe le nom des différents processeurs ainsi que le *speed factor* qui correspond au coefficient multiplicatif pour normaliser les temps CPU correspondant à chaque méthode.

Tableau 2-4
Performance relative des ordinateurs utilisés dans la littérature

	Geismar *et al.* (2008)	Karaoğlan et Kesen (2017)	Notre proposition
Ordinateur	Intel Pentium 4, 3.2GHz	Intel Xeon 3.16 GHz	Intel Core i7 3.4 GHz
OS	Windows XP Pro.	Windows 7	Windows 7
Langage	BASIC	C++ IBM ILOG CPLEX 12.6	Visual C++
Mflops	1573	1892	2671
Speed factor	1.0	1.2	1.7

Le Tableau 2-8 contient la moyenne de la valeur des solutions obtenues sur un jeu de paramètres et sur les six jeux de données ainsi que le temps CPU moyen associé à l'obtention de ces solutions. Les paramètres utilisés pour le GRASP×ELS sont donnés dans le Tableau 2-5.

Tableau 2-5
Paramétrage du GRASP×ELS pour les instances de Geismar et al.

Paramètre	Définition	Valeur
np	Nombre d'itérations pour le GRAPS	120
ne	Nombre d'itérations pour le ELS	50
nd	Nombre de voisins générés	25
lr	Nombre d'itérations pour la recherche locale sur les tournées	10
NBmax	Nombre maximal de label par nœud	5

Dans le tableau résultat, les notations suivantes sont utilisées :

C	Nombre de clients
S	Taille de la zone de répartition des clients
Avg	Valeur moyenne
%Gap	Ecart en pourcentage entre la meilleure solution et la borne inférieure
N/A	Données non disponibles
Nb. Opt	Nombre d'instances où la borne inférieure est atteinte. Dans ce cas, la solution obtenue est une solution optimale du PTSP.
LB_i^k	Borne inférieure du problème avec *SP=k* et *DS=i*
h_i^k	Meilleure solution trouvée pour *SP=k*, *DS=i*
tt_i^k	Temps CPU total couplé à h_i^k
$\overline{LB_n^k} = \underset{1 \le i \le n}{\text{avg}}\ LB_i^k$	Borne inférieure moyenne sur les n jeux de données et pour *SP=k*
$\overline{h_n^k} = \underset{1 \le i \le n}{\text{avg}}\ h_i^k$	Moyenne des meilleures solutions sur les n jeux de données et pour *SP=k*
$\overline{tt_n^k} = \underset{1 \le i \le n}{\text{avg}}\ tt_i^k$	Temps CPU total moyen sur les n jeux de données et pour *SP=k*

Le Tableau 2-6 donne la définition des douze jeux de paramètres, caractérisés par le taux de production r, la capacité Q du véhicule et la durée de vie B du produit . Le Tableau 2-7 contient la définition des six jeux de données, caractérisés par le nombre de clients C et la taille de la zone S de répartition des clients .

Tableau 2-6
Définition des 12 jeux de paramètres SP

SP	r	Q	B
1	1	300	300
2	2	300	300
3	3	300	300
4	1	300	600
5	2	300	600
6	3	300	600
7	1	600	300
8	2	600	300
9	3	600	300
10	1	600	600
11	2	600	600
12	3	600	600

Tableau 2-7
Définition des 6 jeux de paramètres DS

DS	C	S
1	40	100
2	40	200
3	40	300
4	50	100
5	50	200
6	50	300

Chaque ligne du Tableau 2-8 est caractérisée par un jeu de paramètre SP. Chaque valeur du tableau représente une valeur moyenne sur les six jeux de données (DS). Les valeurs reportées dans les tableaux pour (Geismar *et al.*, 2008), (Karaoğlan et Kesen, 2017) et le GRASP×ELS ont été arrondies à l'entier le plus proche pour assurer une comparaison équitable des solutions.

Les résultats obtenus sur les instances de (Geismar *et al.*, 2008) en utilisant la méthode présentée dans la partie 2.2.3 sont reportés dans le Tableau 2-8. Ce tableau permet de comparer les performances de plusieurs méthodes, celle de (Geismar *et al.*, 2008), celle de (Karaoğlan et Kesen, 2017) et enfin notre proposition avec deux variantes, soit avec l'algorithme de Gilmore et Gomory, soit avec l'algorithme de Johnson, pour la méthode d'ordonnancement.

Les résultats du Tableau 2-8, montrent que la méthode du GRASP×ELS proposée est beaucoup plus rapide, avec un temps moyen de 103 ou 102 secondes (avec respectivement l'utilisation de Gilmore-Gomory ou Johnson) contre 169 secondes pour (Geismar *et al.*, 2008) et plus de 3000 secondes pour (Karaoğlan et Kesen, 2017).

De plus, le makespan moyen des solutions sur l'ensemble des instances vaut 7888 par la méthode GRASP×ELS + Gilmore-Gomory, alors qu'il vaut 7891 par la méthode de (Karaoğlan et Kesen, 2017), 7902 par la méthode GRASP×ELS + Johnson et 8045 par la méthode de (Geismar *et al.*, 2008).

Tableau 2-8
Résultats avec les méthodes de résolution pour un véhicule.

SP	DS	LB_6^{SP}	Geismar *et al.* (2008)		Karaoğlan et Kesen (2017)		Notre proposition			
							Gilmore-Gomory		Johnson	
			$\overline{h_{6,1}^{SP}}$	$\overline{tt_{6,1}^{SP}}$	$\overline{h_{6,1}^{SP}}$	$\overline{tt_{6,1}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$
1	1..6	9976	10040	N/A	10049	1853	10156	22	10291	24
2	1..6	9121	9157	N/A	9179	3600	9146	24	9145	25
3	1..6	9103	9125	N/A	9153	3600	9120	25	9119	24
4	1..6	9968	10041	N/A	10048	2484	10038	25	10131	26
5	1..6	9108	9171	N/A	9180	3089	9144	28	9143	27
6	1..6	9091	9153	N/A	9152	3154	9118	28	9117	27
7	1..6	8781	8781	N/A	8782	1000	8781	88	8781	89
8	1..6	4793	5744	N/A	5346	3233	5353	92	5336	92
9	1..6	4181	5421	N/A	4887	3600	4927	94	4930	93
10	1..6	8781	8781	N/A	8782	573	8781	98	8781	97
11	1..6	4759	5724	N/A	5276	3006	5259	100	5237	98
12	1..6	4147	5403	N/A	4861	3304	4836	101	4823	101
	Avg.	5383	8045		7891		7888		7902	
	Avg.time			169		2708		60		60
	Avg.scaled time			169		3249		103		102
	Nb. Opt		16/72		13/72		16/72		16/72	

Comme indiqué dans le Tableau 2-9, notre méthode (avec les deux variantes) obtient des solutions strictement meilleures que l'algorithme de (Geismar *et al.*, 2008) dans ~40% des cas (30 instances pour Gilmore-Gomory et 28 pour Johnson), des solutions identiques dans ~42% des cas et des solutions moins bonnes que (Geismar *et al.*, 2008) dans ~17% des cas. Les deux algorithmes (Johnson et Gilmore-Gomory) pour l'ordonnancement ont produit des résultats similaires et aucune dominance ne peut être observée.

Tableau 2-9
Comparaison des solutions avec Gilmore-Gomory ou Johnson et Geismar et al. (2008).

Algorithme	<	=	>
Gilmore-Gomory	30/72	29/72	13/72
Johnson	28/72	32/72	12/72

2.7 Conclusion du chapitre

Une bonne coordination entre la production et le transport est un point-clé dans la chaîne logistique, en permettant de réduire les coûts et de proposer un meilleur service aux clients. Nous nous sommes donc intéressé à un problème intégrant ces deux composantes avec le problème du PTSP (Production and Transportation Scheduling Problem) à un véhicule.

Ce chapitre propose une présentation détaillée du problème, initialement introduit par (Geismar *et al.*, 2008), avec notamment une formulation linéaire et une analyse poussée sur la modélisation du problème en flow shop à deux machines avec des time-lags maximaux.

L'efficacité du schéma de résolution repose sur une métaheuristique de type GRASP×ELS avec une alternance entre deux espaces de recherche, l'espace des solutions et l'espace de codage, avec une représentation indirecte des solutions sous forme de tours géants. La fonction d'évaluation permettant de transformer un tour géant en solution du PTSP repose sur plusieurs méthodes, dont une procédure de type SPLIT ainsi que deux algorithmes pour la résolution intégrée du problème. Notre méthode a prouvé son efficacité sur le cas à un véhicule grâce à une comparaison avec les résultats obtenus, par (Geismar *et al.*, 2008) avec un algorithme génétique et ceux de (Karaoğlan et Kesen, 2017) obtenus avec un algorithme de type branch-and-cut.

Plusieurs extensions peuvent être proposées pour ce problème, avec par exemple, plusieurs lieux de production et/ou plusieurs véhicules. Le taux de production déjà présent dans le PTSP permet de faire varier le temps de production, cependant aucun paramètre ne permet réellement de faire varier le temps de transport. Pour répondre à cette problématique, et mesurer l'interaction entre la production et le transport dans les problèmes intégrés, une généralisation de ce problème pour plusieurs véhicules est développée dans le chapitre 3.

CHAPITRE 3

Le problème d'ordonnancement de la production et du transport avec plusieurs véhicules

Ce chapitre traite du problème d'ordonnancement de la production et du transport (Production and Transportation Scheduling Problem - PTSP) avec plusieurs véhicules. Ce problème intégré se différencie du problème précédent par la gestion d'une flotte homogène de véhicules. Cette généralisation du problème défini par (Geismar *et al.,* 2008) oblige à repenser très largement le schéma de résolution car le sous-problème se modélise désormais comme un flow shop hybride à deux machines avec time-lags maximaux. Les algorithmes utilisés dans le chapitre 2, propres au flow shop à deux machines, ne sont plus applicables. Nous nous sommes intéressé à la définition d'une méthode de résolution pour le problème à plusieurs véhicules. Celle-ci est capable de fournir, pour le cas particulier à un véhicule, des résultats proches de ceux obtenus avec des méthodes dédiées. De nouveaux jeux de données sont définis pour le cas à plusieurs véhicules et sont proposés comme benchmark pour de futures études.

Nous proposons dans ce chapitre une nouvelle méthode de résolution du PTSP à l'aide d'un GRASP×ELS avec plusieurs contributions. La première contribution concerne la définition d'un algorithme de type SPLIT, adapté au PTSP, avec une flotte homogène de véhicules qui résout le problème de détermination des tournées en prenant également en compte la production. La deuxième contribution repose sur la définition d'un graphe disjonctif qui prend explicitement en compte la durée de vie des produits grâce à des times-lags maximaux. Une recherche locale est définie sur ce graphe et tire profit du chemin critique.

Les expérimentations numériques montrent que la nouvelle méthode de résolution est meilleure que les méthodes précédentes pour le cas à un véhicule. Des expérimentations plus poussées sur le cas à plusieurs véhicules mettent en évidence un seuil, à partir duquel, l'augmentation du nombre de véhicules n'a plus d'impact sur l'amélioration de la qualité des solutions.

3.1 Introduction et état de l'art du problème

Les travaux réalisés dans ce chapitre concernent les problèmes intégrés de production et de transport, dans le cas de produits périssables avec une flotte homogène de véhicules. Le problème étudié dans ce chapitre correspond plus particulièrement à une extension du problème défini par (Geismar *et al.,* 2008), le PTSP, caractérisé par un seul lieu de production et un seul type de produit.

(Chen, 2010) mentionne deux articles traitant de problèmes similaires au PTSP, avec des contraintes de périssabilité, l'article de (Armstrong *et al.*, 2008) cité dans le chapitre précédent et celui de (Devapriya*,* 2006).

(Devapriya, 2006) se sont intéressés à un problème avec un ou plusieurs lieux de production et une flotte homogène de véhicules. La livraison d'un produit doit commencer dès que la production de celui-ci est achevée. L'objectif est de minimiser le coût associé au transport, c'est-à-dire, le coût associé au temps de livraison et au nombre de véhicules utilisés pour satisfaire l'ensemble des demandes.

(Devapriya *et al.,* 2017) ont proposé une extension du problème introduit par (Geismar *et al.,* 2008) avec comme différence majeure la taille de la flotte de véhicules qui devient une variable de décision et une contrainte sur l'horizon de planification. Cette extension a été nommée Integrated Production and Distribution Scheduling Problem (IPDSP).

De nombreux autres articles ont été publiés sur les problèmes intégrés de production et de transport ces dernières années. Le Tableau 3-1 regroupe l'ensemble des articles traitant des problèmes de production avec des problèmes de tournées de véhicules intégrés, dans le cas où la production est réalisée par une seule machine et le transport par une flotte homogène de véhicules (Moons, 2017).

Tableau 3-1
Caractéristiques des problèmes intégrés de production et de transport, avec une machine de production et plusieurs véhicules.

	Production				Distribution										Objectif		
					Véhicules												
	Un seul lieu de production	Production en lots	Un seul produit	Temps de production	Un seul véhicule	Flotte homogène	Nombre illimité	Nombre limité	Temps de transport	Plusieurs tournées	Ordre des clients fixé	Fenêtres de temps	Dates limites de livraison	Produits périssables	Makespan	Coût	Demandes satisfaites
Chang et Lee (2004)	●	●	●	●	●				●	●							
Li *et al.* (2005)	●	●	●	●	●				●	●							
Geismar *et al.* (2008)	●	●	●	●	●				●	●				●	●		
Armstrong *et al.* (2008)	●	●	●	●	●				●	●	●	●		●			●
Cheref *et al.* (2016)	●	●	●	●	●				●				●			●	
Karaoğlan et Kesen (2017)	●	●	●	●	●				●					●	●		
Chen et Vairaktarakis (2005)	●	●	●	●		●	●		●							●	
Devapriya (2006)	●	●	●	●		●	●		●					●		●	
Li *et al.* (2016)	●	●	●	●		●	●		●							●	
Jamili *et al.* (2016)	●	●	●	●		●	●		●							●	
Devapriya *et al.* (2017)	●	●	●	●		●		●	●	●				●		●	
Notre proposition	●	●	●	●		●		●	●	●				●	●		

Trois articles cités dans le Tableau 3-1 ne traitent pas les cas des produits périssables, mais concernent des problèmes proches : il s'agit des articles de (Chen et Vairaktarakis, 2005), (Li *et al*., 2016) et (Jamili *et al*., 2016).

(Chen et Vairaktarakis, 2005), ont étudié le problème intégré de production et de transport avec une unique machine de production mais également avec le cas de machines parallèles. Le transport est pris en compte avec un coût fixe lié à l'utilisation d'un véhicule

et un coût variable qui dépend de la taille de la tournée. Leur objectif est de maximiser la qualité de service pour les clients grâce aux dates de livraison, tout en minimisant les coûts de livraison. (Li *et al*., 2016) n'ajoutent aucune contrainte supplémentaire au problème intégré de production et de livraison, mais ils s'intéressent au cas multi-objectifs avec la minimisation : 1) d'un coût lié aux véhicules, avec la minimisation du nombre de véhicules utilisés ainsi que la minimisation de la distance totale parcourue, et 2) du temps d'attente des clients. Dans le problème étudié par (Jamili *et al*., 2016), la production des produits ne peut débuter avant une certaine date, imposée par le fournisseur. L'objectif de ce problème est de minimiser deux critères, la moyenne des dates de livraison et le coût associé au transport.

3.2 Définition du problème

Le problème étudié dans ce chapitre est une extension du problème traité par (Geismar *et al.,* 2008), défini dans le chapitre précédent. En effet, le problème étudié est caractérisé par une flotte homogène (et limitée) de véhicules, et non plus par un seul véhicule. Cette section propose une description du problème avec plusieurs véhicules, en mettant l'accent sur les différences et les similitudes avec le problème à un seul véhicule.

3.2.1 Définition générale

Le PTSP avec plusieurs véhicules peut être défini grâce aux notations présentées dans le chapitre 2 sur le problème à un véhicule :
0 : le lieu de production situé en $(x, y) = (0, 0)$
r : le taux de production, $r \in \{1, 2, 3\}$
B : la durée de vie du produit
n : le nombre de clients
E : l'ensemble des clients, $i \in \{1, \dots, n\}$
q_i : la quantité demandée par le client i, $i \in \{1, \dots, n\}$
(x_i, y_i) : les coordonnées géographiques du client i, $i \in \{1, \dots, n\}$
$\tau_{i,j}$: le coût du transport associé au trajet entre le client i et le client j, cette matrice respecte l'inégalité triangulaire

À ces notations viennent s'ajouter les notations suivantes, permettant la prise en compte de la flotte homogène de véhicules :
N : le nombre de véhicules
V : l'ensemble des véhicules k disponibles, $k \in \{1, \dots, N\}$
Q : la capacité des véhicules

Le coût du transport associé à la durée du trajet entre le client i et le client j, noté $\tau_{i,j}$, correspond au plus court chemin entre les deux clients, ce coût peut donc être pré-calculé. Ce coût ne dépend pas du véhicule utilisé puisque tous les véhicules ont les mêmes caractéristiques (vitesse, capacité, etc.).

Comme pour le problème à un véhicule, une solution est composée d'un ensemble de $n_J \leq n$ jobs. Chaque job $j \in [\![1, n_J]\!]$ est défini par deux opérations O_{j1} et O_{j2}^k. Seule la définition des opérations O_{j2}^k correspondant aux opérations de transport diffère du

problème à un seul véhicule. Ces opérations peuvent désormais être affectées aux différents véhicules disponibles. Chaque opération de transport permet de décrire une sous-tournée, l'ensemble des sous-tournées affectées au même véhicule permet de définir la tournée d'un véhicule.

Pour caractériser une solution, la notation suivante est introduite :
O_{j2}^{k} : opération de transport associée au job *j* et affectée au véhicule *k*

Certaines contraintes doivent être ajoutées, d'autres mises à jour pour répondre au cas à plusieurs véhicules et définir une solution :

- l'ordre des opérations de production et de transport est identique pour un véhicule donné ;
- un véhicule réalise au plus une tournée.

3.2.2 Illustration des problématiques autour du PTSP avec un exemple

Dans ce paragraphe, le problème du PTSP avec plusieurs véhicules est présenté en reprenant les jobs créés dans la solution du chapitre précédent (voir Figure 2-1 du chapitre 2) pour illustrer les points-clés de la résolution intégrée. Pour résoudre ce problème, trois sous-problèmes doivent être résolus conjointement : l'affectation des véhicules à des tournées, le transport et l'ordonnancement.

La Figure 3-1 présente une solution composée des mêmes jobs (opérations de production et de transport) que la Figure 2-1 du chapitre 2 mais avec deux véhicules. En ajoutant un véhicule, on remarque que le makespan est égal à 19 contre 21 pour un seul véhicule. Le véhicule 1 réalise deux sous-tournées modélisées par les deux opérations de transport O_{12}^{1}, O_{32}^{1} et le véhicule 2 réalise la sous-tournée associée à l'opération de transport O_{22}^{2}. Il faut remarquer que les deux opérations de transport O_{12}^{1}, O_{32}^{1} ont lieu séquentiellement puisqu'elles sont affectées au même véhicule 1. En effet, la date de début au plut tôt de O_{32}^{1} est contrainte par la date de fin de O_{12}^{1}. Comme l'opération de transport O_{22}^{2} est affectée au véhicule 2, il n'y pas de contrainte de précédence à respecter entre O_{22}^{2} et aucune autre opération de transport affectée à un autre véhicule. Ceci se visualise sur la Figure 3-1, par un chevauchement des deux opérations de transport O_{12}^{1} et O_{22}^{2}, entre les dates 8 et 10.

On rappelle que les opérations O_{11} et O_{12}^{1}, formant le job 1, sont rattachées aux clients 1 et 2. Les opérations O_{21} et O_{22}^{2} (job 2) permettent de modéliser la production et le transport des demandes liées au client 3, et les clients 4 et 5 sont regroupés dans les opérations O_{31} et O_{32}^{1} (job 3).

En utilisant les mêmes jobs, on peut remarquer que l'ajout d'un troisième véhicule ne réduirait pas le makespan de la solution puisqu'il n'y a aucun temps d'attente entre les opérations de production et les opérations de transport d'un même job. En effet, toutes les opérations sont ordonnancées au plus tôt et cet ordonnancement correspond au cas no-wait.

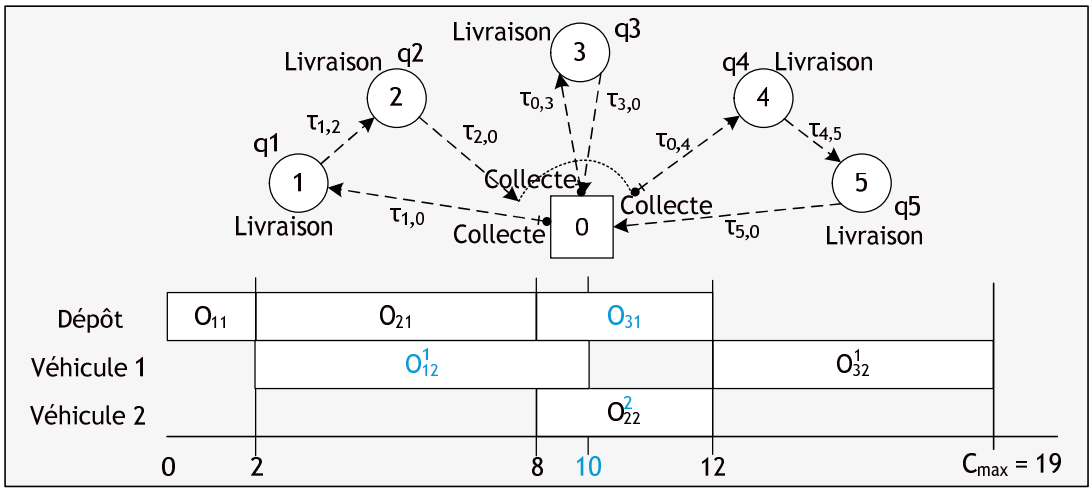


Figure 3-1. Exemple de solution du PTSP avec deux véhicules

Une solution détaillée illustrant plus précisément l'impact de l'ajout d'un véhicule est fournie sur la Figure 3-2. Le graphe de cette figure permet de faire apparaître les dépendances temporelles entre les opérations de transport affectées à un même véhicule. On peut noter que le début de l'opération de transport O_{32}^1 dépend à la fois de la fin de l'opération de transport O_{12}^1 et de la fin de l'opération de production O_{11}.

D'un autre côté, aucune dépendance temporelle n'existe entre les opérations de transport de deux véhicules différents, comme dans l'encadré de la Figure 3-2. La date de début de l'opération de transport O_{22}^2 dépend uniquement de la date de fin de l'opération de productionO_{21}.

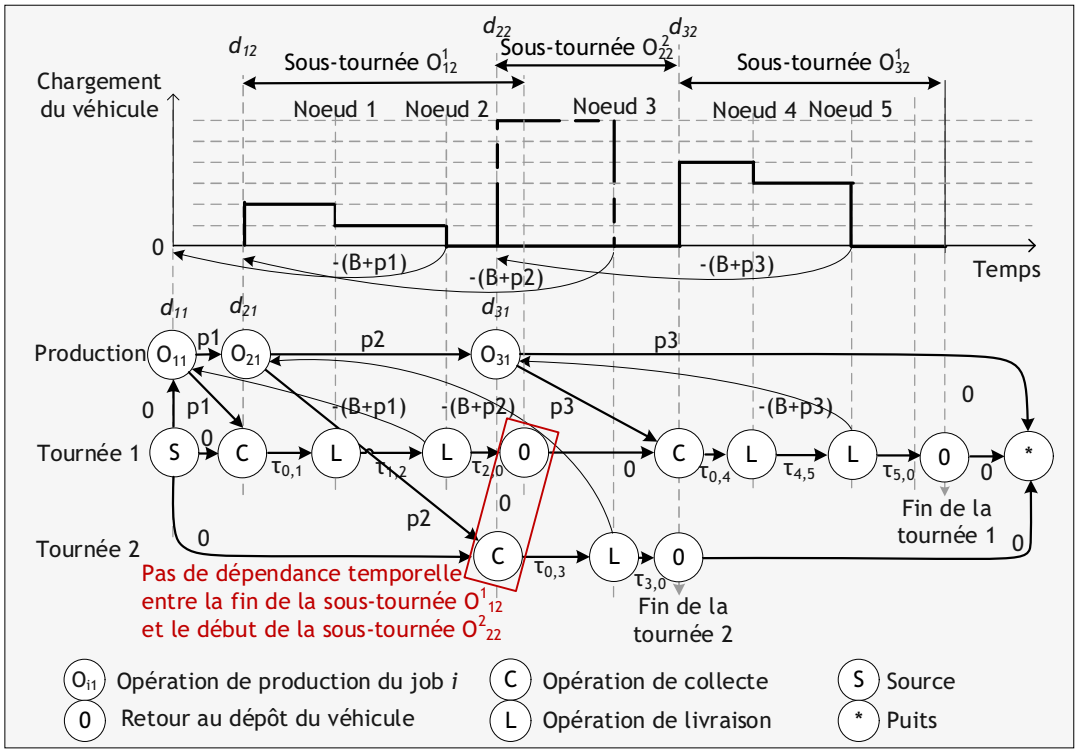


Figure 3-2. Représentation détaillée de la solution à deux véhicules

Chaque sous-tournée est toujours composée d'une opération de collecte au niveau du dépôt suivie d'une ou plusieurs opérations de livraison sur les nœuds clients (Figure 3-2). Les time-lags maximaux sont également présents entre la dernière opération de livraison d'une sous-tournée et l'opération de production de cette sous-tournée, permettant ainsi de modéliser la contrainte de durée de vie du produit.

Dans cet exemple, la sous-tournée modélisée par l'opération transport O_{22}^2 faisant partie de la tournée 2 (véhicule 2), débute avant la fin de la sous-tournée O_{12}^1, les opérations sont effectuées en parallèle (voir Tableau 3-2). En effet, l'opération de transport O_{22}^2 est planifiée entre la date 8 et 12 alors que l'opération de transport O_{12}^1 est planifiée entre 2 et 10. Pendant l'intervalle de temps [8, 10] les deux opérations de transport s'exécutent donc en parallèle.

Tableau 3-2
Exemple de solution du PTSP avec deux véhicules

	Sous-tournée O_{12}^1				Sous-tournée O_{22}^2				Sous-tournée O_{32}^1				
	O_{11}	0	1	2	0	O_{21}	0	3	0	O_{31}	0	4	5	0
Date de début	0					2				8				
Fenêtre de temps			]−∞;9]				]−∞;15]					]−∞;19]		
Date de départ		2	5	7			8	10			12	15	18	
Date d'arrivée			5	7	10			10	12			15	18	19

L'intérêt de disposer d'une flotte de véhicules est donc de permettre la réalisation de plusieurs tournées simultanément.

3.3 Formalisation linéaire : proposition

La formalisation linéaire présentée ici repose sur 5 groupes de contraintes permettant de définir complètement le problème du PTSP. Le premier groupe de contraintes permet de constituer les jobs en définissant la répartition des clients dans des jobs. Le deuxième groupe de contraintes assure l'affectation d'une opération de transport, correspondant à un job, à un véhicule. Le troisième groupe de contraintes garanti le respect des contraintes liées au problème de tournées de véhicules. Le quatrième groupe de contraintes définit les disjonctions sur la production et le transport. Enfin le cinquième groupe de contraintes complète la définition du problème avec la contrainte sur la durée de vie du produit ainsi que les contraintes conjonctives entre la production et le transport, et la définition du makespan.

Les données du problème dans la formulation reprennent les notations présentées précédemment. On introduit plusieurs variables binaires :

- y_{bi} qui vaut 1 si le client i appartient au job b ;
- h_b qui vaut 1 si le job b existe, c'est-à-dire s'il contient au moins un client ;
- x_{ij}^b qui vaut 1 si le client i et le client j appartiennent au job b et si le client i précède immédiatement le client j ;
- z_{bc} qui vaut 1 si le job b précède le job c ;
- a_{bk} qui vaut 1 si le job b est affecté au véhicule k.

On introduit également la variable q_i^- qui est égale à la quantité de produit dans le véhicule après la livraison du client i. Enfin, on considère une constante suffisamment grande que l'on note H.

Le programme linéaire P, donné sur la Figure 3-3, fait apparaître les contraintes suivantes :

(1) objectif ;

Étape 1 :
(2) cette contrainte assure que chaque client i appartient à un job b ;
(3) cette contrainte assure que les demandes clients dans un job b ne dépassent pas la capacité des véhicules ;
(4) permet de définir l'existence d'un job b ;
(5) permet de définir la durée de l'opération de production associé à un job b ;

Étape 2 :
(6) cette contrainte assure l'affectation de chaque job b existant à un et un seul véhicule ;

Étape 3 :
(7) cette contrainte assure l'existence d'un job b ;
(8) et (9) ces contraintes assurent que chaque client i soit desservi une et une seule fois ;
(10) et (11) ces contraintes assurent que pour chaque job b existant, l'opération de transport associée commence et se termine au dépôt ;
(12) et (13) ces contraintes assurent que chaque client i appartenant à un job b appartienne à l'opération de transport du job b ;
(14) et (15) ces contraintes empêchent les sous-tours lors de la création des tournées (contrainte classique MTZ) ;
(16) permet de définir la durée de l'opération de transport associé à un job b ;
(17) permet de définir la durée de l'opération de transport associée à un job b du dépôt au dernier client (le retour au dépôt n'est pas pris en compte) ;

Étape 4 :
(18) et (19) ces contraintes assurent le bon ordonnancement des opérations de production avec le respect des disjonctions ;
(20) et (21) ces contraintes assurent le bon ordonnancement des opérations de transport avec le respect des disjonctions ;

Étape 5 :
(22) cette contrainte assure un ordonnancement correct entre les opérations de production et les opérations de transport entre elles avec le respect des conjonctions ;
(23) et (24) ces contraintes assurent que l'ordonnancement respecte la durée de vie du produit ;
(25) correspond à l'expression classique du makespan : C_{max} donne la date à laquelle toutes les opérations de transport sont terminées ;

Définition des variables :
(26), (27), (28), (29) et (30) les variables $y_{bi}, h_b, x_{ij}^b, z_{bc}$ et a_{bk} sont binaires ;
(31), (32) et (33) les durées p_b, t_b et tt_b sont des réels positifs ou nuls ;
(34) les quantités q_i^- sont des entiers positifs ou nuls ;
(35) et (36) les dates d_{j1} et d_{j2} sont des réels positifs ou nuls ;

$$P:\begin{cases}
Minimiser\ C_{max} & & (1)\\
\forall i \in E & \sum_{b\in J} y_{bi} = 1 & (2)\\
\forall b \in J & \sum_{i\in E} y_{bi}\,.\,q_i \le C & (3)\\
\forall b \in J & y_{bi} - h_b \le 0 & (4)\\
\forall b \in J & p_b - \sum_{i\in E} y_{bi}\,.\,q_i.r = 0 & (5)\\
\forall b \in J & h_b - \sum_{k\in V} a_{bk} = 0 & (6)\\
\forall (i,j) \in E^2, i \ne j, \forall b \in J & x_{ij}^b - h_b \le 0 & (7)\\
\forall j \in E & \sum_{b\in J} \sum_{\substack{i\in E\cup\{0\}\\ i\ne j}} x_{ij}^b = 1 & (8)\\
\forall i \in E & \sum_{b\in J} \sum_{\substack{j\in E\cup\{0\}\\ j\ne i}} x_{ij}^b = 1 & (9)\\
\forall b \in J & h_b - \sum_{j\in E} x_{0j}^b = 0 & (10)\\
\forall b \in J & h_b - \sum_{i\in E} x_{i0}^b = 0 & (11)\\
\forall i \in E, \forall b \in J & y_{bi} - \sum_{\substack{j\in E\cup\{0\}\\ j\ne i}} x_{ij}^b = 0 & (12)\\
\forall i \in E, \forall b \in J & y_{bi} - \sum_{\substack{j\in E\cup\{0\}\\ j\ne i}} x_{ji}^b = 0 & (13)\\
\forall b \in J, \forall (i,j) \in E^2, i \ne j & q_j^- - q_i^- + x_{ij}^b.C \le C - q_j & (14)\\
\forall b \in J, \forall (i,j) \in E^2, i \ne j & q_i^- \le C - q_i & (15)\\
\forall b \in J & t_b - \sum_{i\in E\cup\{0\}} \sum_{\substack{j\in E\cup\{0\}\\ j\ne i}} x_{ij}^b\,.\,\tau_{ij} = 0 & (16)\\
\forall b \in J & tt_b - t_b + \sum_{i\in E} x_{i0}^b.\tau_{i0} = 0 & (17)\\
\forall (b,c) \in J^2, c \ne b & d_{b1} + p_b - d_{c1} + z_{bc}.H \le H & (18)\\
\forall (b,c) \in J^2, c \ne b & d_{c1} + p_c - d_{b1} - z_{bc}.H \le 0 & (19)\\
\forall (b,c) \in J^2, c \ne b & d_{b2} + t_b - d_{c2} + (z_{bc} + a_{bk} + a_{ck}).H \le 3.H & (20)\\
\forall (b,c) \in J^2, c \ne b & d_{c2} + t_c - d_{b2} + (z_{bc} - a_{bk} - a_{ck}).H \le 2.H & (21)\\
\forall b \in J & d_{b1} + p_b - d_{b2} \le 0 & (22)\\
\forall b \in J & d_{b2} - d_{b1} - p_b - tt_b \le B & (23)\\
\forall b \in J & t_{tb} \le B & (24)\\
\forall b \in J & d_{b2} + t_b - C_{max} \le 0 & (25)\\
\forall b \in J, \forall i \in E & y_{bi} \in \{0,1\} & (26)\\
\forall b \in J & h_b \in \{0,1\} & (27)\\
\forall b \in J, \forall (i,j) \in E^2, i \ne j & x_{ij}^b \in \{0,1\} & (28)\\
\forall (b,c) \in J^2, c \ne b & z_{bc} \in \{0,1\} & (29)\\
\forall b \in J, \forall k \in V & a_{bk} \in \{0,1\} & (30)\\
\forall b \in J & p_b \in \mathbb{R} & (31)\\
\forall b \in J & t_b \in \mathbb{R} & (32)\\
\forall b \in J & tt_b \in \mathbb{R} & (33)\\
\forall i \in E & q_i^- \in \mathbb{N} & (34)\\
\forall b \in J & d_{b1} \in \mathbb{R} & (35)\\
\forall b \in J & d_{b2} \in \mathbb{R} & (36)
\end{cases}$$

Figure 3-3. Programme linéaire pour le PTSP

3.4 Proposition pour la résolution du problème avec plusieurs véhicules

3.4.1 Approche de résolution : description et analyse du contexte

L'approche de résolution proposée dans ce chapitre, se base : 1) sur une métaheuristique qui va parcourir l'ensemble des tours géants avec un schéma similaire à celui utilisé pour le cas à un véhicule, 2) sur une nouvelle fonction « d'évaluation » qui permet à partir d'un tour géant d'obtenir une solution du PTSP à plusieurs véhicules.

La première étape de l'évaluation d'un tour géant, porte sur l'ordonnancement et le transport (routing) dans le PTSP et permet la création de la tournée du véhicule avec les sous-tournées qui la composent. Suite à cela, la deuxième étape effectue une recherche locale sur les tournées précédemment générées pour améliorer celles-ci. Une troisième étape permet de modéliser le problème du PTSP à plusieurs véhicules en problème de flow shop hybride à deux machines avec des time-lags maximaux. Cette modélisation favorise l'utilisation d'algorithmes et d'outils, tels que le graphe disjonctif dans la quatrième étape.

Une fois l'évaluation terminée, une procédure de concaténation permet de transformer une solution du PTSP en tour géant. Cette cinquième étape est également représentée sur la Figure 3-4.

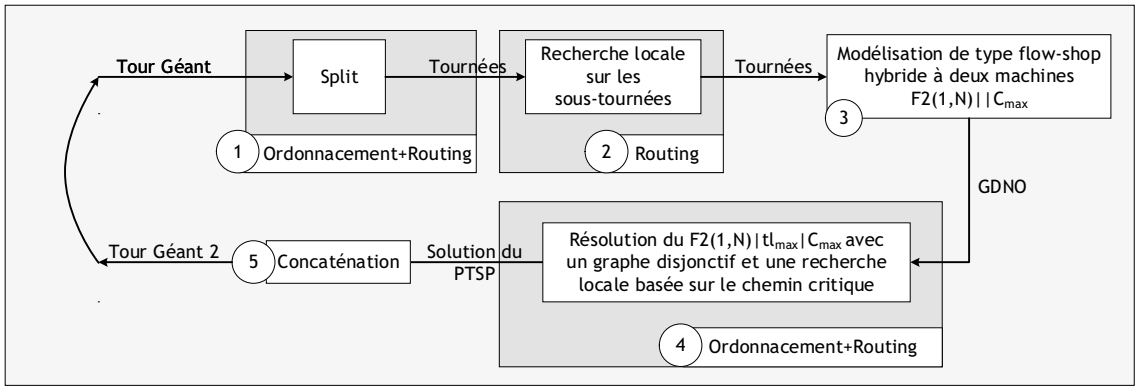


Figure 3-4. Construction d'une solution pour le PTSP à plusieurs véhicules

La résolution du problème du PTSP à plusieurs véhicules présentée dans cette partie révèle plusieurs points spécifiques, la différenciant de la méthode de résolution énoncée dans le chapitre précédent pour le problème à un véhicule. Le Tableau 3-3 résume les caractéristiques des deux approches.

Tableau 3-3
Différences entre les étapes de résolution du PTSP à un ou plusieurs véhicules

	Résolution du PTSP à un véhicule	Résolution du PTSP à plusieurs véhicules
SPLIT	Minimisation du temps de transport	Minimisation du makespan (production et transport)
Modélisation du problème	Flow shop à deux machines avec time-lags maximaux	Flow shop hybride à deux machines avec time-lags maximaux
Résolution intégrée	Plusieurs étapes avec la résolution de sous-problèmes grâce à des algorithmes exactes.	Une seule étape basée sur la résolution du problème complet grâce à un graphe disjonctif

3.4.2 Contributions

La résolution du problème du PTSP à plusieurs véhicules repose également sur plusieurs points-clés inspirés du cas à un véhicule et toujours liés à l'introduction de deux espaces de recherche représentés sur la Figure 2-12 du chapitre 2.

Le premier espace de recherche contient l'ensemble des tours géants associés au problème du TSP et le deuxième espace de recherche contient les solutions du PTSP avec plusieurs véhicules. La fonction de codage qui permet d'associer à un tour géant une solution du PTSP se compose de 4 étapes principales (Figure 3-4). Une première étape qui résout un problème intégré pour déterminer à la fois les caractéristiques des opérations de transport et des opérations de production, une étape de recherche locale sur le routage, une étape d'intégration (modélisation comme un flow shop hybride à deux machines) et une quatrième étape dans laquelle un graphe disjonctif est utilisé pour obtenir une résolution intégrée du problème.

La résolution du PTSP avec plusieurs véhicules repose sur cinq étapes :
- une méthode SPLIT dédiée qui permet d'obtenir les sous-tournées du véhicule à partir d'un tour géant (étape 1, Figure 3-4) ;
- une recherche locale sur les sous-tournées pour améliorer la tournée d'un véhicule (étape 2, Figure 3-4) ;
- un outil de modélisation permettant de modéliser le problème en $F2(1,N)|tl_{max}|C_{max}$ (étape 3, Figure 3-4) ;
- une méthode d'ordonnancement permettant d'obtenir directement une solution du $F2(1,N)|tl_{max}|C_{max}$ suivie d'une recherche locale (étape 4, Figure 3-4) ;
- une méthode de concaténation pour transformer une solution du PTSP en tour géant (étape 5, Figure 3-4).

Ces différentes étapes sont présentées dans les sections suivantes.

La méthode de construction du tour géant, la recherche locale sur les tournées et la méthode de concaténation ne sont pas affectées par la généralisation à plusieurs véhicules, les définitions proposées pour le cas à un véhicule restent valables. Les principaux changements concernent la méthode SPLIT et la méthode d'ordonnancement.

L'algorithme SPLIT proposé pour le problème à plusieurs véhicules diffère du cas à un véhicule dès la définition du label. Le label doit maintenant fournir des informations sur l'ensemble des véhicules du problème. Cette modification engendre la définition de

nouvelles règles de propagation et de dominance. La définition d'une nouvelle procédure SPLIT est une contribution principale pour le cas à plusieurs véhicules et permet en plus de traiter le cas 1 véhicule comme un cas particulier.

Les méthodes d'ordonnancement proposées pour le cas à un véhicule (Gilmore-Gomory et Johnson) ne sont plus utilisables pour résoudre le problème à plusieurs véhicules. La définition d'une nouvelle modélisation et d'une nouvelle recherche locale constituent deux contributions importantes pour la partie ordonnancement du cas à plusieurs véhicules.

3.4.3 Construction des tournées : proposition d'une nouvelle procédure SPLIT pour le cas à plusieurs véhicules

Cette nouvelle procédure SPLIT utilise également un algorithme de plus court chemin appliqué sur un graphe auxiliaire $G = (X, A)$ identique à celui présenté dans le chapitre 2 de par sa structure.

Le problème de tournées possède, cette fois, une contrainte supplémentaire due au nombre de véhicules disponibles, chaque véhicule pouvant être assimilé à une machine du système. L'algorithme SPLIT du PTSP à plusieurs véhicules est une extension du SPLIT introduit pour le Heterogeonous fleet Vehicle Routing Problem (HVRP) par (Duhamel *et al.*, 2012). Dans le problème à plusieurs véhicules la date de fin au plus tôt de la machine de production doit être prise en compte pour chaque opération, ainsi que les dates de fin au plus tôt de chaque sous-tournée pour chaque véhicule. Le calcul d'un plus court chemin à contrainte de ressource dans un graphe est généralement effectué par un algorithme à labels (Desrocher, 1988). Ici, plusieurs labels sont générés sur chaque nœud du graphe auxiliaire, et le point-clé de l'algorithme repose donc sur la définition efficace d'un label contenant le coût d'une solution ainsi que l'état des ressources (ici la machine de production et les véhicules).

La solution obtenue à l'issue du SPLIT, est une solution du PTSP avec plusieurs véhicules en no-wait. La contrainte de durée de vie est cependant prise en compte pour vérifier la faisabilité des sous-tournées. En effet, les arcs présents dans le graphe auxiliaire $G = (X, A)$ modélisent des sous-tournées réalisables. Par conséquent, un arc $(i, j) \in A$ si la somme des demandes de la sous-tournée n'excède pas la capacité du véhicule, *i.e.* $\sum_{k=i+1}^{j} p_{\sigma_k} \leq C$, et si la durée nécessaire pour effectuer la sous-tournée sans le retour au dépôt n'excède pas la durée de vie du produit, *i.e.* $\tau_{0,\sigma_{1+1}} + \sum_{k=i+1}^{j-1} \tau_{\sigma_l,\sigma_{l+1}} \leq B$.

(Duhamel *et al.*, 2012) propose une description générique des algorithmes SPLIT, celle-ci fait apparaître quatre points-clés :

- la définition d'un label basée sur la disponibilité des ressources ;
- une règle de dominance sur les labels pour supprimer des labels et ne garder que ceux non dominés, c'est-à-dire ceux susceptibles d'aboutir au découpage optimal ;
- une règle de propagation pour créer un label sur le nœud j à partir d'un label sur le nœud i, en fonction de la séquence fixée par le tour géant.

Définition du label

Soit $L_i^p = (l_{0,i}, l_{1,i}, \dots, l_{N,i})$ le $p^{ième}$label du nœud i avec $V = N$ le nombre de véhicules. La valeur $L_i^p(j) = l_{j-1,i}$ pour $j \in [\![2, N+1]\!]$ correspond à la date de fin au plus tôt de la dernière sous-tournée (date de disponibilité) du véhicule $k = j - 1$ pour $k \in [\![1, N]\!]$ et $L_i^p(1) = l_{0,i}$ est égale à la date de fin au plus tôt de la production pour les clients $(\sigma_1, \dots, \sigma_i)$.

Le label initial sur le nœud 0 est égal à $L_0^1 = (0,\ 0, \dots, 0)$. Il représente une solution où tous les véhicules sont disponibles à la date 0 ainsi que la machine de production.

Règle de propagation du label

Étant donné un arc $(i, j) \in A$, un véhicule k pour effectuer la sous-tournée associée et un label L_i^p, un nouveau label $L_j^q = \left(l_{0,j},\ l_{1,j}, \dots, l_{N,j}\right)$ peut-être généré grâce à la règle de propagation suivante :

- $l_{0,j} = \max\left(\mathrm{l_{0,i}} + \sum_{\mathrm{l=i+1}}^{\mathrm{j}} \mathrm{p_{\sigma_l}}\ ;\ \mathrm{l_{k,i}}\right) = max\left(l_{0,i} + \sum_{l=i+1}^{j} \frac{q_{\sigma_l}}{r}\ ;\ l_{k,i}\right)$
- $l_{k,j} = l_{0,j} + \tau_{0,\sigma_{1+1}} + \sum_{l=i+1}^{j-1} \tau_{\sigma_l,\sigma_{l+1}} + \tau_{\sigma_j,0}$.

Les solutions partielles construites au cours de la procédure SPLIT permettent la construction de solution en no-wait, c'est-à-dire que la date de fin d'une opération de production doit être égale à la date de début de l'opération de transport du même job. Cette contrainte est traduite par l'opérateur max dans la règle de propagation pour calculer la date de fin au plus tôt de l'opération de production. Le maximum permet de prendre en compte deux cas.

Dans le premier cas, illustré sur la Figure 3-5, l'opération de production O_{j1} à ajouter, correspondant à l'arc (i, j) du graphe auxiliaire, a une durée égale à $\sum_{l=i+1}^{j} p_{\sigma_l}$. La date de fin au plus tôt de la production est alors égale à $d_{j1} = f_{i1} + \sum_{l=i+1}^{j} p_{\sigma_l}$ puisque cette date est plus grande que la date de fin de l'opération de transport $f_{i2} = l_{1,i}$ associée au même véhicule que celui du job i (véhicule 1 dans l'exemple de la Figure 3-5). Dans ce cas, on a $d_{j1} = f_{i1}$.

Dans le deuxième cas, illustré sur la Figure 3-5, l'opération de production O_{j1} à ajouter correspondant à l'arc (i, j) du graphe auxiliaire, a aussi une durée égale à $\sum_{l=i+1}^{j} p_{\sigma_l}$. Cependant, la date de fin au plus tôt de la production n'est pas égale à $f_{i1} + \sum_{l=i+1}^{j} p_{\sigma_l}$ mais à $f_{i2} = l_{1,i}$ puisque cette date est supérieure. Pour respecter la contrainte de no-wait, on a donc $f_{j1} = f_{i2}$, on pourra alors observer une période d'inactivité sur la machine de production (Figure 3-5).

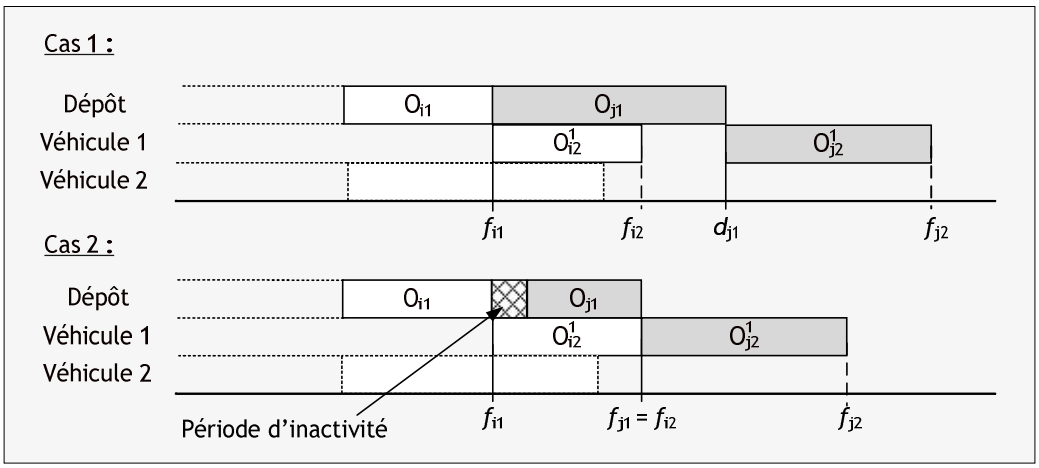


Figure 3-5. Illustration des deux cas pour la propagation d'un label.

Le nombre de labels créés à partir d'un label est égal au nombre de véhicules, et par conséquent, un grand nombre de labels peuvent être générés au cours de la procédure SPLIT. L'introduction d'un critère de dominance pour supprimer des labels et améliorer la performance de l'algorithme est nécessaire pour obtenir une implémentation efficace et doit en plus permettre de conserver l'optimalité de la solution trouvée.

Proposition d'une règle de dominance entre les labels

Le label $L = (l_0,\ l_1, \dots, l_N)$ domine le label $P = (l_0,\ l_1, \dots, l_N)$ si après avoir trié chaque label par valeurs croissantes (on a alors respectivement $L = (l_{\sigma_0}^{\ l}, \dots, l_{\sigma_N}^{\ l})$ et $P = (l_{\sigma_0}^{\ p}, \dots, l_{\sigma_N}^{\ p})$), on respecte les conditions suivantes :

- $\exists i \in \{0, \dots, N\}, \mathrm{l}_{\sigma_i}^{\ l} < \mathrm{l}_{\sigma_i}^{\ p}$;
- $\forall j \in \{0, \dots, N\}, \mathrm{l}_{\sigma_j}^{\ l} \leq \mathrm{l}_{\sigma_j}^{\ p}$.

La règle de dominance permet de limiter le nombre de labels stockés sur chaque nœud. Cependant, plusieurs auteurs dont (Duhamel *et al*., 2012) ont noté que malgré la règle de dominance, un très grand nombre de labels peuvent être générés et stockés au niveau des nœuds du graphe SPLIT. Dans leur article de 2012, Duhamel *et al.* ont par ailleurs montré que limiter le nombre de labels sauvegardés sur chaque nœud est une option intéressante, même si cela ne permet plus de garantir l'optimalité du découpage. En effet, l'étude réalisée par (Duhamel *et al*., 2011), montre que pour le HVRP, le fait de réduire fortement le nombre de labels sauvegardés sur chaque nœud n'engendre pas nécessairement une forte détérioration de la qualité des solutions. Dans la plupart des cas, l'écart entre les solutions finales et le découpage optimal reste très modéré. Ce type de considérations a permis, par exemple, d'utiliser des algorithmes SPLIT dans des métaheuristiques avec l'utilisation d'une méthode SPLIT particulièrement efficace en temps de calcul.

Ce type de considération peut s'appliquer au problème traité ici, et on peut considérer que le nombre de labels dépend d'un paramètre NB_{max} offrant un compromis entre, le temps de calcul que l'on souhaite allouer à la procédure de découpage et la qualité du découpage. La détermination d'une valeur de compromis « acceptable » (dans un sens à définir) pour NB_{max} doit faire l'objet d'une évaluation en fonction des ensembles d'instances à traiter.

Algorithme 1 . SPLIT

```
1.  procedure Split
2.  input parameters
3.    T: giant tour
4.  output parameters
5.    S: VRP-PTSP solution
6.  global parameter
7.    Q : maximal vehicle weight capacity
1.    B : lifespan of the product
8.    q_i : total items order by customer i
9.    τ_ij : cost from customer i to j
10.   n : number of customers
11.   N : number of vehicles
12.   NB_max : maximal number of labels in each node
13.  begin
14.  | L_0^1:=(0,0,…,0), S := ∅
15.  | for i := 1 to n do NB_i := 0 endfor
16.  | for j := 0 to n do
17.  | | j := i
18.  | | stop := false
19.  | | while (j < n and stop=false)
20.  | | | customer := T_j
21.  | | | if (j = i) then
22.  | | | | production_cost := d_customer/r
23.  | | | | transport_cost := c_depot,customer + c_customer,depot
24.  | | | else
25.  | | | | production_cost += d_customer/r
26.  | | | | transport_cost += c_customer-1,customer +c_customer,depot −c_customer-1,depot
27.  | | | endif
28.  | | | if ((production_cost*r < Q) and (transport_cost -c_customer,depot ≤ B))then
29.  | | | | for p := 1 to NB_i do
30.  | | | | | if (d_Pi-1 = 0) then
31.  | | | | | | insertion of the label in first position with the first vehicle
32.  | | | | | else
33.  | | | | | | v:=0
34.  | | | | | | do
35.  | | | | | | | v:=v+1
36.  | | | | | | | L:=Propagation_label_to_vehicle (L_i^p,v,i)
38.  | | | | | | | if (NB_i=0)
39.  | | | | | | | | insertion of L in first position
40.  | | | | | | | else
41.  | | | | | | | | CD:=CheckDomination(L, j, N_j)
42.  | | | | | | | | | call InsertLabel(L, j, CD, N_j, NB_max)
43.  | | | | | | | endif
44.  | | | | | | while (v < N)
45.  | | | | | endif
46.  | | | | endfor
47.  | | | else
48.  | | | | stop := true
49.  | | | endif
50.  | | | j := j + 1
51.  | | endwhile
52.  | endfor
53.  | S := call Extract_trips ()  //save the best solution
54.  end
```

La procédure détaillée dans *Algorithme 1* utilise plusieurs sous-procédures :

- `Propagation_label_to_vehicle(L,v,i)`, propage le label *L* en utilisant le véhicule v sur le noeud *i* et retourne le label créé ;

- `CheckDomination(L, i, Ni)`, applique la règle de dominance entre le label L et tous les labels stockés sur le nœud i, N_i. Cette fonction retourne 0 si L est dominé par au moins un label sur le nœud i. Dans ce cas, L n'est pas stocké sur le nœud. Elle retourne 1 si L domine au moins un label sur le nœud i. Si les deux labels sont incomparables la fonction retourne 2 ;
- `InsertLabel(L, i, CD, Ni, NBmax)`, ajoute le label L sur le nœud i si $CD \in \{1,2\}$ et si le nombre de label sur le nœud N_i n'excède pas le nombre maximal de labels sur un nœud imposé par NB_{max}. Si CD=1 tous les labels sur le nœud i dominés par L sont supprimés. Cette liste de label est triée par coût décroissant ;
- `Extract_trip()`, parcours le plus court chemin du graphe grâce au label du dernier nœud au premier et retourne la meilleure solution avec les tournées correspondantes.

L'algorithme est composé de deux parties : l'initialisation où les variables locales sont initialisées (lignes 14 et 15) et une boucle while de la ligne 16 à 52 qui permet de parcourir les clients du tour géant T. La deuxième boucle while de la ligne 19 à 51 permet l'évaluation des séquences partielles $(\sigma_{i+1}, \dots, \sigma_j)$. Si $i = j$ une nouvelle sous-tournée est créée, les coûts sont initialisés (lignes 22 et 23) et mis à jour dans les lignes 25 et 26.

La condition exprimée sur la ligne 28 permet de vérifier si les contraintes (capacité des véhicules et durée de vie du produit) sont respectées. La boucle principale peut être prématurément stoppée grâce à un booléen (stop=true) pour ne considérer que les sous-tournées réalisables. La boucle *for* (line 29) permet le parcours de tous les labels stockés sur le nœud *i* et permet la propagation des labels L_i^P.

3.4.4 Exemple d'exécution de la procédure SPLIT

Considérons une instance composée de 6 clients, 3 véhicules de capacité 10, un produit avec une durée de vie égale à 20 et un taux de production $r = 1$. Le Tableau 3-4 fournit la demande de chaque client et la matrice des distances entre les clients avec le dépôt noté 0. Le tour géant considéré dans cet exemple est $TG = (4,3,2,6,5,1)$.

Tableau 3-4
Demande des clients et matrice des distances

	0	1	2	3	4	5	6	q_i
0	0	10	15	5	10	5	10	
1	10	0	10	10	30	20	20	3
2	15	10	0	25	30	20	10	4
3	5	10	25	0	10	15	20	2
4	10	30	30	10	0	10	15	6
5	5	20	20	15	10	0	10	5
6	10	20	10	20	15	10	0	3

Le label initial sur le nœud 0 est égal à $L = (0,0,0,0)$, ce qui modélise un système dans lequel la production peut débuter à la date 0, et tous les véhicules sont également disponibles au dépôt à la date 0.

L'arc (0,1) dans le graphe auxiliaire sur la Figure 3-8 correspond à la sous-tournée comprenant seulement le client 4 *i.e.* $C_1 = \{4\}$, avec $p_1' = q_4 = 6$. Les véhicules étant identiques, on ne génére qu'un seul label à partir du label initial en utilisant l'arc (0,1). On obtient alors le label suivant sur le nœud 1 : $P = (6,26,0,0)$. Ce label modélise une solution dans laquelle la production de la demande du client 4 se termine à la date 6 et le transport associé se termine à la date $f_{21} = 6 + 20 = 26$.

La Figure 3-6 représente la solution modélisée par le label (6,26,0,0) obtenu après la propagation du label (0,0,0,0) entre le nœud 0 et le nœud 1. Cette solution correspond à la production et la livraison de la demande associée au client $C_1 = \{4\}$. La Figure 3-7 représente la solution où les clients 4 et 3 sont regroupés dans les opérations de production et de transport d'un même job. Cette solution est modélisée par le label (8,33,0,0) qui a été créé grâce à la règle de propagation entre le nœud 0 et le nœud 2.

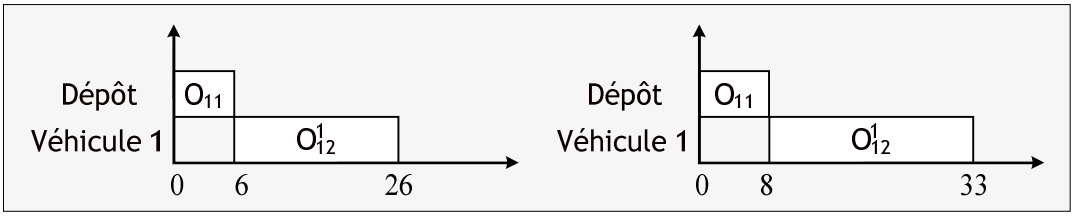


Figure 3-6. Première propagation du label initial (0,0,0,0) *avec* $C_1 = \{4\}$

Figure 3-7. Deuxième propagation du label initial (0,0,0,0) *avec* $C_1 = \{4, 3\}$

Le label initial a été propagé pour obtenir le label (6,26,0,0) sur le nœud 1 et le label (8,33,0,0) sur le nœud 2. À la prochaine itération, le label (6,26,0,0) est propagé du nœud 1 au nœud 2 et permet l'obtention du label (26,36,0,0) en affectant le véhicule 1 au nouveau job et du label (8,26,18,0) en affectant le véhicule 2 au nouveau job. Les labels créés sont propagés et permettent la création de 7 labels sur le nœud 3, chaque label représentant une solution partielle différente (Figure 3-8).

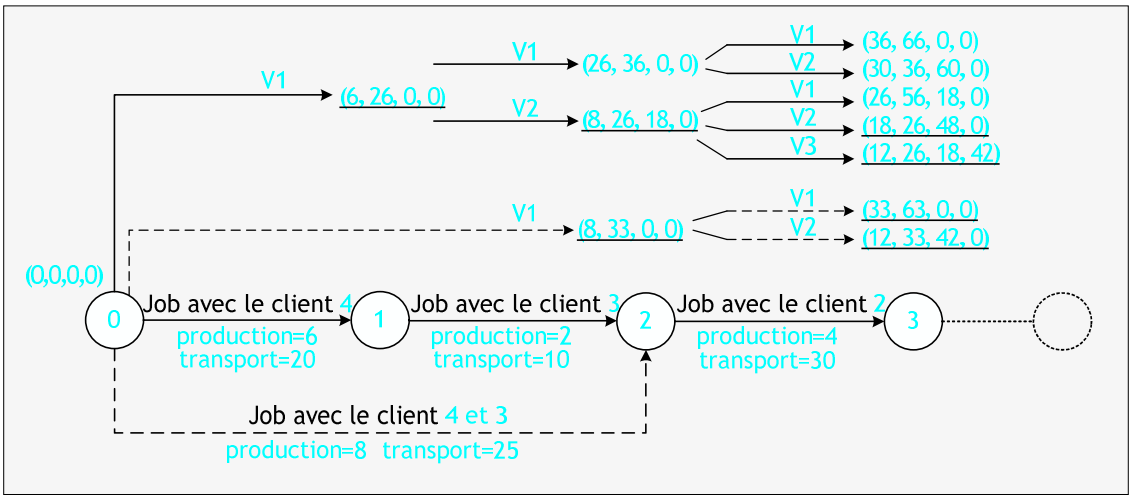


Figure 3-8. Exemple des labels générés durant le SPLIT dans le graphe auxiliaire

Un label peut générer deux labels modélisant des solutions très différentes. Une telle situation est représentée sur la Figure 3-9 qui contient deux solutions associées aux labels $(33,63,0,0)$ et $(12,33,42,0)$. Ces deux labels ont été générés à partir du label $(8,33,0,0)$ et modélisent les deux solutions partielles représentées sur la Figure 3-9.

Dans les deux solutions représentées sur la Figure 3-9, quatre opérations sont ordonnancées. Ces quatre opérations appartiennent à deux jobs différents :

- La suite ordonnée des opérations O_{11} et O_{12}^* définit le job 1.
  O_{11} : opération de production correspondant à la fabrication des produits des clients $\{4,3\}$;
  O_{12} : opération de transport correspondant à la livraison des produits aux clients $\{4,3\}$;
- La suite ordonnée des opérations O_{21} et O_{22}^* définit le job 2.
  O_{21} : opération de production correspondant à la fabrication des produits du client $\{2\}$;
  O_{22} : opération de transport correspondant à la livraison des produits du client $\{2\}$;

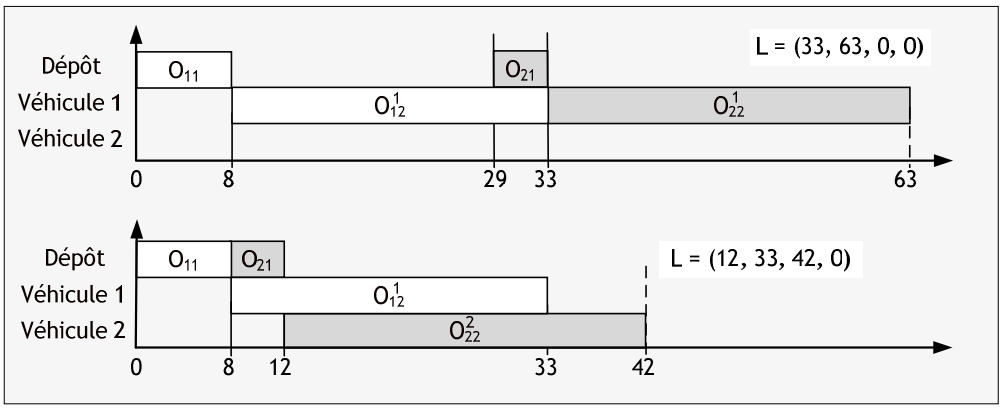


Figure 3-9. Propagation du label $(8,33,0,0)$ *vers le nœud 3 avec les véhicules 1 ou 2*

Cependant, suivant le choix d'affectation des véhicules aux opérations de transport, deux solutions différentes peuvent être obtenues. La solution du haut sur la Figure 3-9 correspond au cas où les deux opérations de transport (O_{12}^*, O_{22}^*) sont affectées au véhicule 1. Par conséquent, les deux opérations de transport sont effectuées séquentiellement. La solution du bas sur la Figure 3-9 correspond au cas où les deux opérations de transport (O_{12}^*, O_{22}^*) sont affectées respectivement au véhicule 1 et au véhicule 2. Cette solution permet la réalisation en parallèle des deux opérations de transport entre les dates 12 et 33.

La meilleure solution obtenue à la fin de l'algorithme est donnée par le label $(26, 46, 45, 42)$ et la solution associée est représentée sur la Figure 3-10. Les véhicules V_1, V_2, V_3, V_2, V_1 sont utilisés dans cet ordre pour réaliser les 5 opérations de transport associées aux 5 jobs créés lors du SPLIT.

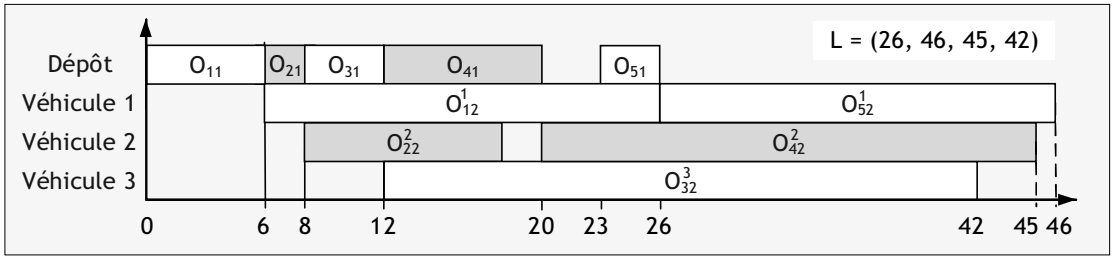


Figure 3-10. Solution finale de l'exemple

Dans la solution représentée sur la Figure 3-10, cinq jobs ont été constitués :

- le job 1 modélise la production et le transport du client {4} ;
- le job 2 modélise la production et le transport du client {3} ;
- le job 3 modélise la production et le transport du client {2} ;
- le job 4 modélise la production et le transport des clients {5,6} ;
- le job 5 modélise la production et le transport du client {1} ;

3.4.5 Modélisation

L'étape 1 (Figure 3-4) détaillée dans la sous-partie précédente a permis la création des différentes opérations de production et de transport associées au problème du PTSP avec plusieurs véhicules.

La nature du problème du PTSP à un véhicule permet de le modéliser comme un flow shop à deux machines avec time-lags maximaux, puisque pour chaque job l'ordre de visite des deux machines est identique. Dans le flow shop à deux machines, chacune des deux machines représente un étage (Figure 3-11). Le problème du PTSP à plusieurs véhicules est également un flow shop, cependant chaque opération de transport peut désormais être réalisée par un véhicule parmi une flotte homogène de véhicules. On parle alors de flow shop hybride à deux étages (Figure 3-12).

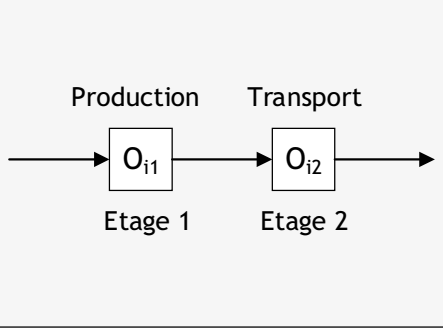


Figure 3-11. Flow shop à deux machines, $F2(1,1)$.

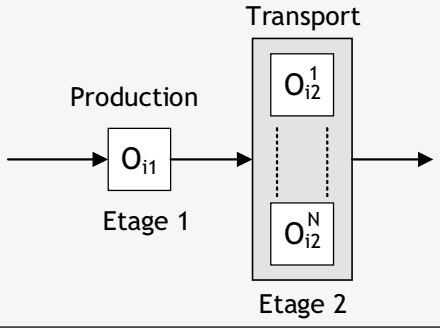


Figure 3-12. Flow shop hybride à deux étages, $F2(1,N)$.

Le problème du PTSP à plusieurs véhicules suit le modèle d'un flow shop hybride à deux étages, dans lequel le premier étage est constitué d'une seule machine pour les opérations de production et le deuxième étage est constitué de N machines identiques pour la flotte homogène de N véhicules pour les opérations de transport.

Un graphe disjonctif (Roy et Sussmann, 1964) peut été utilisé pour modéliser les jobs avec les opérations de production et de transport dans le cadre d'un flow shop. Cette approche a été détaillée dans le chapitre 2, section 2.4.5. La modélisation du PTSP à plusieurs véhicules reprend les mêmes principes que pour le cas à un véhicule.

Pour obtenir une solution du PTSP à partir des jobs, il faut arbitrer les disjonctions sur les opérations de production et sur les opérations de transport effectuées par la même machine. Par conséquent, pour définir une solution du PTSP, les disjonctions doivent être arbitrées pour former : 1) une séquence avec toutes les opérations de production, 2) une séquence d'opérations de transport pour chaque véhicule, et 3) un ordre relatif des opérations de production et de transport, qui doit être le même dans chaque séquence. Une autre contrainte doit être prise en compte dans la modélisation, les time-lags maximaux. Les time-lags maximaux sont représentés par des arcs entre les opérations de production et de transport d'un même job.

La Figure 3-13 illustre un choix d'arbitrage des disjonctions sur la production et le transport. Dans cet exemple les opérations sont réalisées dans l'ordre croissant, c'est-à-dire, les opérations de production sont réalisées dans l'ordre *(O_{11}, O_{21}, O_{31}, O_{41}, O_{51})* et les opérations de transport dans le même ordre sur le véhicule 1 *(O^1_{12}, O^1_{52})*, sur le véhicule 2 *(O^2_{22}, O^2_{42})* et sur le véhicule 3 *(O^3_{32})*.

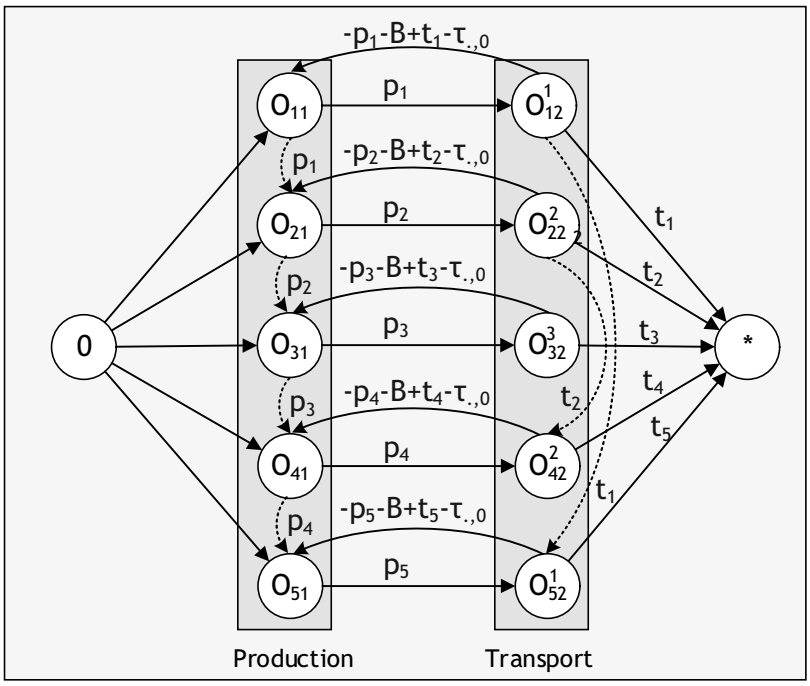


Figure 3-13. Modélisation de cinq jobs sous forme de graphe disjonctif orienté pondéré avec des time-lags maximaux

La résolution du flow shop hybride à deux étages peut être envisagée par à une approche directe avec une recherche locale. Cette méthode est présentée dans la section suivante.

3.4.6 Construction d'une solution du PTSP à partir des sous-tournées

L'algorithme SPLIT utilise comme donnée d'entrée une séquence de client et permet la création d'opérations de transport et de production respectant les contraintes du PTSP. La modélisation présentée dans la section précédente montre que le problème correspond à un flow shop hybride à deux étages. Le premier étage est constitué d'une seule machine pour la production et le deuxième étage est constitué de N machines, sachant qu'une machine de l'étage 2 est en réalité un véhicule dans la terminologie du PTSP.

Ce problème peut être modélisé par un graphe disjonctif (Roy et Sussmann, 1964) $\mathrm{G} = (\mathrm{V}, \mathrm{A}, \mathrm{E_P}, \mathrm{E_T})$, avec :

- V : l'ensemble des nœuds du graphe, chaque opération de production ou de transport est modélisée par un nœud. De plus, un nœud source noté 0 est connecté à l'ensemble des opérations de production, et un nœud puits $*$ est connecté à toutes les opérations de transport, permettant de définir une opération initiale et une opération finale au problème (Figure 3-14);
- A : l'ensemble des arcs conjonctifs du graphe, c'est-à-dire, tous les arcs entre une opération de production et une opération de transport d'un même job ;
- $\mathrm{E_P}$: l'ensemble des arcs disjonctifs entre deux opérations de production ;
- $\mathrm{E_T}$: l'ensemble des arcs disjonctifs entre deux opérations de transport effectuées par le même véhicule.

Les arcs conjonctifs modélisent les contraintes de précédence entre l'opération de production et l'opération de transport d'un même job. Ces arcs ont un coût positif égal à la durée de production des demandes de l'opération de transport, puisque l'opération de transport ne peut commencer que lorsque l'opération de production est terminée.

Deux types d'arcs disjonctifs sont présents dans la modélisation du problème. Chaque arc disjonctif sur la production connecte deux opérations de production O_{i1} et O_{j1}(avec un coût p_i) dans cet ordre, permettant de modéliser le fait que l'opération O_{j1} ne peut débuter que lorsque l'opération O_{i1} est terminée puisque ces deux opérations partagent la même ressource.

Il en est de même pour les opérations de transport O_{i2} et O_{j2}effectuées par le même véhicule et qui sont aussi reliées par des arcs disjonctifs avec un coût égal à t_i pour prendre en compte la durée de l'opération de transport (Figure 3-14). Une solution faisable correspond à un graphe acyclique qui peut être évalué avec un algorithme de plus long chemin pour obtenir les dates de début de toutes les opérations ainsi que le makespan C_{max} obtenu sur l'opération finale $*$.

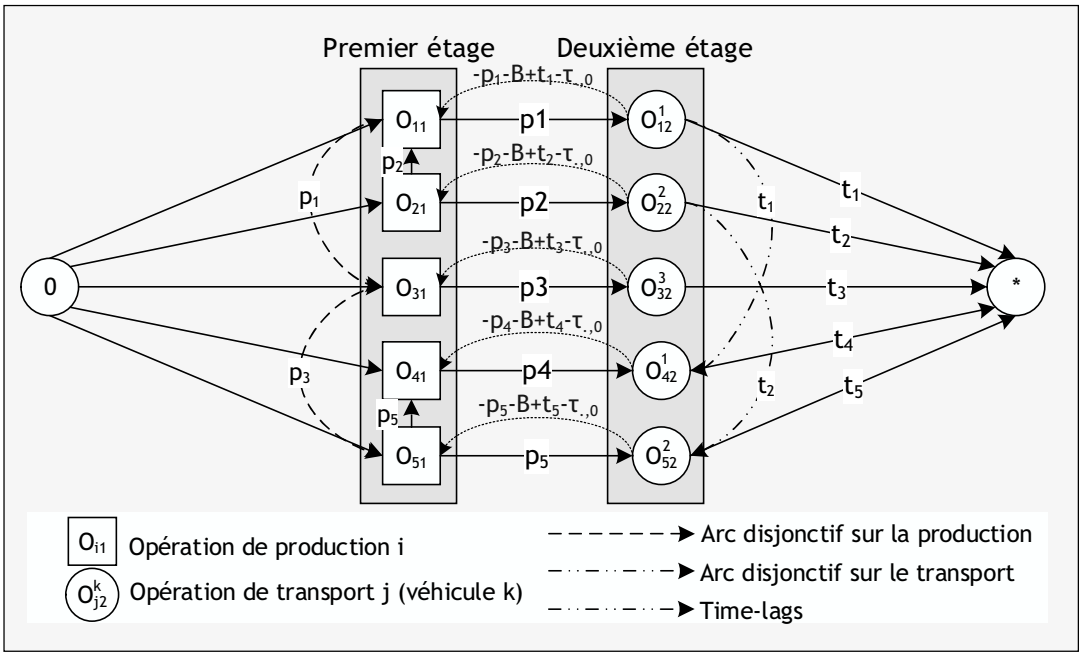


Figure 3-14. Schéma général du graphe disjonctif avec 5 jobs et 3 véhicules.

Dans la Figure 3-14, deux arcs disjonctifs sur le transport sont présents puisque deux opérations de transport sont effectuées par le véhicule 1 et deux autres par le véhicule 2. Le véhicule 1 est affecté aux opérations de transport des jobs 1 et 5 avec une valeur sur l'arc égale à t_1. Cette valeur détermine la durée minimale entre la date de début au plus tôt de l'opération de transport du job 1 et la date de début au plus tôt de l'opération de transport du job 4. La durée de vie intervient entre la date de début d'une opération de production et la date de début de l'opération de transport associée.

Le vecteur de Bierwirth (Bierwirth, 1995) permet une représentation indirecte de la solution et, couplé avec le graphe disjonctif, il permet l'ordonnancement des opérations relatives aux jobs créés dans ce cas par la procédure SPLIT. Un algorithme de recherche locale peut être défini grâce au chemin critique dans le graphe disjonctif avec le voisinage de (Van Laarhoven *et al.,* 1992) et celui de (Grabowski *et al.,* 1986) qui introduisent la notion de blocs.

La recherche locale peut directement agir sur le vecteur de Bierwirth en effectuant des permutations sur le vecteur des jobs. Ces permutations sont déterminées grâce à la définition des blocs et au parcours du chemin critique (de la fin au début). Lorsqu'une permutation permet de générer une meilleure solution, l'exploration du chemin critique recommence à la fin du graphe. Un maximum de *ns* itérations est défini.

3.4.7 Exemple d'exécution de la procédure d'ordonnancement avec recherche locale

Considérons la même instance que celle introduite au début de ce chapitre avec les jobs constitués lors du SPLIT. Le problème est donc composé de 5 jobs avec 3 véhicules, soit un total de dix opérations à ordonnancer. Les temps de production des jobs sont égaux à $p = (6, 2, 4, 8, 3)$ et les temps de transport des jobs $\mathrm{t} = (20, 10, 30, 25, 20)$.

Pour débuter la procédure d'ordonnancement, un vecteur de Bierwirth est généré, par exemple $VB = [1,2,3,4,5]$. Pour cet exemple, l'affectation des véhicules aux jobs dans le vecteur de Bierwirth peut être représentée par le vecteur $[1,2,3,2,1]$, (ici le job 5 est affecté au véhicule 1).

À partir de ce vecteur un graphe disjonctif peut être construit. Plusieurs arcs disjonctifs sur la production doivent être pris en compte. La production étant effectuée sur une seule machine, les arcs disjonctifs vont être situés entre deux opérations de production associées à deux numéros de job consécutifs dans le vecteur de Bierwirth. Par exemple, entre le job 1 en première position dans le vecteur et le job 2 en deuxième position. Cependant, l'opération de transport du job 1 étant affectée au véhicule 1 et l'opération de transport du job 2 étant affectée au véhicule 2, il n'y a pas d'arc disjonctif entre ces deux opérations de transport. Le graphe est également composé de deux arcs disjonctifs sur le transport : l'arc entre l'opération de transport du job 1 et l'opération de transport du job 5 qui sont toutes deux affectées au véhicule 1, et l'arc entre l'opération de transport du job 2 et l'opération de transport du job 4 qui sont toutes deux affectées au véhicule 2. Le graphe peut ensuite être évalué grâce à un algorithme de plus long chemin pour obtenir un makespan égal à 46 (Figure 3-15).

À partir de cette solution, le chemin critique (en gras dans Figure 3-15) peut être utilisé pour améliorer la solution en reprenant une démarche proche de celles classiquement utilisées pour le job shop. Rappelons que les meilleures techniques utilisées sont celles basées sur la notion de blocs avec les propositions de (Van Laarhoven *et al.,* 1992) et de (Grabowski *et al.,* 1986).

Dans cet exemple, on permute les jobs 1 et 5 car l'arc disjonctif sur le transport entre ces deux jobs est sur le chemin critique (Figure 3-15). Cette permutation permet la création d'un nouveau vecteur $VB = [5,2,3,4,1]$ correspondant à une solution de coût 43 (Figure 3-16).

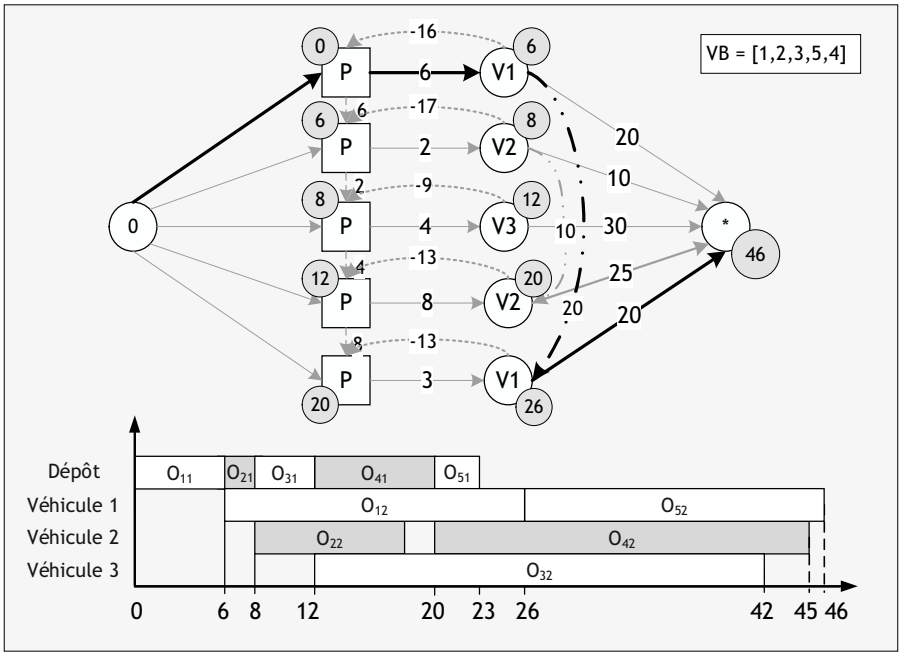


Figure 3-15. Détail de l'étape 1 de la procédure d'ordonnancement sur un exemple

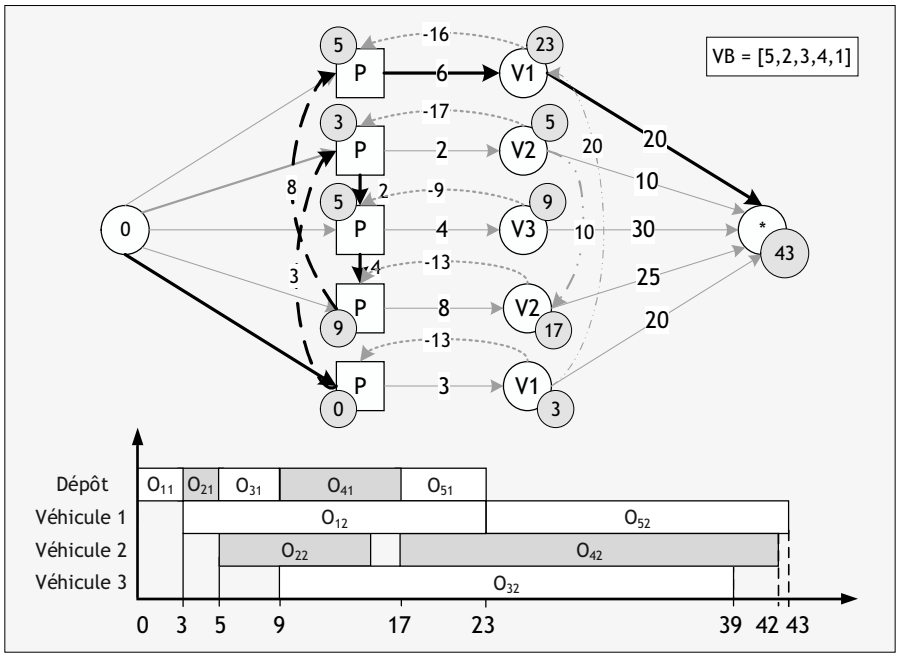


Figure 3-16. Détail de l'étape 2 de la procédure d'ordonnancement sur un exemple

3.4.8 Proposition de nouvelles bornes inférieures

Afin de pouvoir juger de la qualité des solutions du problème du PTSP avec plusieurs véhicules, une nouvelle borne inférieure peut être introduite. Trois bornes inférieures pour le problème du PTSP à un véhicule ont été introduites par (Geismar *et al.*, 2008). Cependant, ces bornes inférieures doivent être étendues pour le cas à plusieurs véhicules. En considérant des sous-tournées composées d'un seul client avec un nombre $N \geq n$ de véhicules , une nouvelle borne inférieure peut être définie en s'inspirant de l'algorithme de Jackson (Jackson, 1955). L'algorithme de Jackson traite d'un problème d'ordonnancement à une machine avec des dates de fin souhaitées pour les jobs. La borne inférieure est donc basée sur l'algorithme de Jackson qui crée un ordonnancement optimal des opérations sur une machine en minimisant le retard maximal. Cet ordre consiste à ordonner les jobs sur les dates de fin souhaitées de façon croissante.

La ligne 18 de l'Algorithme 2 permet de définir la somme des durées des jobs déjà ordonnancés. À la ligne 19, les dates de fin des jobs sont mises à jour comme étant le maximum entre la valeur courante et la somme des durées des jobs déjà ordonnancés plus la date d'échéance du job. Ces deux étapes n'apparaissent pas dans l'algorithme original de Jackson, elles ont été ajoutées pour correspondre aux attentes du problème du PTSP. Pour appliquer cet algorithme sur le problème du PTSP, un parallèle doit être observé entre les deux problèmes. La durée des jobs est égale à la durée des opérations de production et les dates de fin souhaitées correspondent à la durée des opérations de transport. La borne inférieure est donnée par le C_{max} à la fin de l'algorithme.

Algorithme 2 : Borne inférieure basée sur l'algorithme de Jackson

```
1.  procedure A lower bound based on the Jackson Rule
2.  input parameters
3.    J: set of job/sub-trip
4.  global parameter
5.    rj : release date of the job/ sub-trip j
6.    pj : processing time of the job/ sub-trip j
7.    q : sum of the processing time already scheduled
8.    dj : due date of the job/ sub-trip j
9.    n : number of customers/ sub-trip
10.  output parameters
11.   Cmax : lower bound
12.  begin
13.  | L := {1, …, n}, q := 0
14.  | if  (J ≠ ∅) then
15.  | | while (L ≠ ∅) do
16.  | | | u := minj{rj}, j in L
17.  | | | choose i in L such as ri = u and di maximal
18.  | | | q :=  q + pi ;
19.  | | | for all k in L do rk:= max{rk, q + di}, Cmax := rk; endfor
20.  | | | L := L\{i};
21.  | | endwhile
22.  | endif
23.  end
```

3.5 Schéma général de résolution du PTSP : GRASP×ELS

Le schéma général de résolution est basé sur une métaheuristique de type GRASP×ELS permettant l'intégration des procédures présentées précédemment, et permet la résolution du PTSP avec plusieurs véhicules mais peut également résoudre le cas particulier à un véhicule.

Le GRASP×ELS permet l'intégration des points-clés mentionnés précédemment :

- une recherche locale sur les tournées avec des opérations de type 2-Opt ou échange dans une sous-tournée ;
- une méthode SPLIT pour la création des sous-tournées (des jobs) en minimisant le makespan associé au problème intégré ;
- une méthode d'ordonnancement basée sur l'utilisation d'un graphe disjonctif ;
- une procédure de concaténation pour obtenir un tour géant à partir d'une solution du PTSP.

Le GRASP×ELS est défini sur deux espaces de solution : les solutions du TSP avec des tours géants et les solutions du PTSP intégrant production et transport. L'efficacité de la méthode repose sur différents éléments (la fonction d'évaluation et de concaténation et l'opérateur de mutation) définis sur des espaces solutions spécifiques pour favoriser l'exploration de la métaheuristique. La structure générale du GRASP×ELS étend celle présentée dans le chapitre 2 avec des différences majeures dans la procédure d'évaluation, notamment une nouvelle procédure SPLIT et l'ajout d'une recherche locale sur le graphe disjonctif.

3.6 Résultats numériques

3.6.1 Résultats numériques pour le PTSP avec un seul véhicule

Les instances introduites par (Geismar *et al*., 2008) sont utilisées pour tester la méthode par rapport à celles proposées par (Geismar *et al*., 2008) et par (Karaoğlan et Kesen, 2017) pour le cas à un véhicule. Les instances proposées par (Geismar *et al*., 2008) sont détaillées dans le chapitre 2 avec le Tableau 2-3. Les informations relatives aux différentes machines, ainsi que le *speed factor* qui correspond au coefficient multiplicatif pour normaliser les temps CPU correspondant à chaque méthode, sont présentées dans le chapitre 2. Les paramètres utilisés pour le GRASP×ELS sont donné dans le Tableau 3-5.

Tableau 3-5
Paramétrage du GRASP×ELS pour les instances de Geismar et al.

Paramètre	Définition	Valeur
np	Nombre d'itérations pour le GRAPS	150
ne	Nombre d'itérations pour le ELS	30
nd	Nombre de voisins générés	15
lr	Nombre d'itérations pour la recherche locale sur les tournées	40
ns	Nombre d'itérations pour la recherche locale sur l'ordonnancement	500
NBmax	Nombre maximal de labels par nœud	5

Par rapport aux paramètres utilisés pour le cas à un véhicule, le nombre d'itérations du GRASP a été augmenté de 120 à 150 afin d'intensifier l'exploration de l'espace des solutions : ceci se justifie par le fait que le problème traité possède une combinatoire bien supérieure puisqu'il prend en compte l'affectation des véhicules aux opérations de transport. Un nouveau paramètre est ajouté pour la recherche locale sur l'ordonnancement puisque cette étape n'apparaissait pas dans le cas à un véhicule directement résolu par l'algorithme de Johnson ou Gilmore-Gomory.

Dans les tableaux résultats, les notations sont les suivantes :

C	Nombre de clients
S	Taille de la zone de répartition des clients
Avg	Valeur moyenne
%Gap	Ecart en pourcentage entre la meilleure solution et la borne inférieure
N/A	Données non disponibles
Nb. Opt	Nombre d'instances où la borne inférieure est atteinte (au moins une fois parmi les 5 réplications). Dans ce cas, la solution obtenue est une solution optimale du PTSP.
LB_i^k	Borne inférieure du problème avec *SP=k* et *DS=i*
$h_{i,j}^k$	Meilleure solution trouvée pour *SP=k*, *DS=i* et *rep=j*
$tt_{i,j}^k$	Temps CPU total couplé à $h_{i,j}^k$
$t_{i,j}^k$	Temps CPU pour obtenir $h_{i,j}^k$
$\overline{LB_n^k} = \underset{1 \le i \le n}{\text{avg}}\ LB_i^k$	Borne inférieure moyenne sur les n jeux de données et pour *SP=k*

$\overline{h_{n,m}^k} = \underset{\substack{1\le i\le n \\ 1\le j\le m}}{\text{avg}}\ h_{i,j}^k$	Moyenne des meilleures solutions sur les n jeux de données, avec m réplications et pour $SP{=}k$
$\overline{tt_{n,m}^k} = \underset{\substack{1\le i\le n \\ 1\le j\le m}}{\text{avg}}\ tt_{i,j}^k$	Temps CPU total moyen sur les n jeux de données, avec m réplications et pour $SP{=}k$
$\overline{t_{n,m}^k} = \underset{\substack{1\le i\le n \\ 1\le j\le m}}{\text{avg}}\ t_{i,j}^k$	Temps CPU moyen sur les n jeux de données, avec m réplications et pour $SP{=}k$
h_i^k	Meilleure solution parmi les 5 réplications, avec $SP{=}k$ et $DS{=}i$
tt_i^k	Temps CPU total couplé à h_i^k
$t_{i,j}^k$	Temps CPU pour obtenir $h_{i,j}^k$
$\overline{h_n^k} = \underset{1\le i\le n}{\text{avg}}\ h_i^k$	Moyenne des solutions h_i^k sur les n jeux de données et pour $SP{=}k$
$\overline{tt_n^k} = \underset{1\le i\le n}{\text{avg}}\ tt_i^k$	Moyenne des temps tt_i^k sur les n jeux de données et pour $SP{=}k$
$\overline{t_n^k} = \underset{1\le i\le n}{\text{avg}}\ t_i^k$	Moyenne des temps $t_{i,j}^k$ sur les n jeux de données et pour $SP{=}k$

Lors des tests, 5 réplications (rep) de notre algorithme ont été effectuées. Les tableaux résultats (Tableau 3-6 et Tableau 3-8) contiennent la moyenne de la meilleure solution obtenue sur chacune des 5 réplications ainsi que le temps CPU moyen associé à l'obtention de ces solutions.

Chaque ligne du Tableau 3-6 et du Tableau 3-8 est caractérisée par un jeu de paramètre SP. Chaque valeur du tableau représente une valeur moyenne sur les 6 jeux de données (DS) et sur les 5 réplications pour notre algorithme.

Les valeurs reportées dans les tableaux pour (Geismar *et al.*, 2008), (Karaoğlan et Kesen, 2017) et de la méthode GRASP×ELS, ont été arrondis à l'entier le plus proche. Par exemple, dans le Tableau 3-7 à ligne 4, la valeur 9994 a été obtenue à partir de la valeur 9994.06 dans la publication de (Karaoğlan et Kesen, 2017).

Le Tableau 3-6 permet de comparer les résultats obtenus avec chacune des trois méthodes. L'algorithme génétique proposé par (Geismar *et al.*, 2008) est exécuté pendant un temps moyen de 169 secondes, alors que notre algorithme est exécuté pendant 82.7 secondes. Cependant le temps moyen d'exécution du branch-and-cut proposé par (Karaoğlan et Kesen, 2017) est bien plus élevé, avec 3249 secondes. De plus, notre algorithme fournit sur l'ensemble des instances une valeur moyenne des solutions égales à 7884, et montre clairement qu'il est plus performant que les algorithmes de (Geismar *et al.*, 2008) avec une valeur moyenne de 8445, et de (Karaoğlan et Kesen, 2017) avec une valeur moyenne de 7891.

Tableau 3-6
Résultats sur les instances de (Geismar et al., 2008)

SP	DS	$\overline{LB_6^{SP}}$	Geismar *et al.* (2008)			Karaoğlan et Kensen (2017)			Notre proposition					
			$\overline{h_{6,1}^{SP}}$	$\overline{tt_{6,1}^{SP}}$	$\overline{t_{6,1}^{SP}}$	$\overline{h_{6,1}^{SP}}$	$\overline{tt_{6,1}^{SP}}$	$\overline{t_{6,1}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_6^{SP}}$	$\overline{tt_6^{SP}}$	$\overline{t_6^{SP}}$
1	1..6	8781	10040	N/A	N/A	10049	1853.5	N/A	10039	40.7	21.8	10027	40.8	26.7
2	1..6	4412	9157	N/A	N/A	9179	3600.0	N/A	9148	47.8	18.2	9144	47.2	21.3
3	1..6	2956	9125	N/A	N/A	9153	3600.0	N/A	9118	49.7	18.5	9117	49.3	27.2
4	1..6	8781	10041	N/A	N/A	10048	2484.0	N/A	10026	38.6	20.0	10020	38.5	28.5
5	1..6	4412	9171	N/A	N/A	9180	3089.1	N/A	9151	50.1	15.7	9145	49.6	16.4
6	1..6	2956	9153	N/A	N/A	9152	3154.3	N/A	9122	52.2	21.2	9119	52.1	19.2
7	1..6	8781	8781	N/A	N/A	8782	1000.5	N/A	8782	27.2	21.3	8781	18.3	18.3
8	1..6	4412	5744	N/A	N/A	5346	3232.8	N/A	5347	147.7	63.7	5333	145.2	60.7
9	1..6	2956	5421	N/A	N/A	4887	3600.0	N/A	4919	188.7	99.5	4897	188.7	71.7
10	1..6	8781	8781	N/A	N/A	8782	573.3	N/A	8781	4.8	4.8	8781	1.0	1.0
11	1..6	4412	5724	N/A	N/A	5276	3005.7	N/A	5300	150.6	64.8	5287	148.2	83.0
12	1..6	2956	5403	N/A	N/A	4861	3304.1	N/A	4875	194.2	118.7	4858	194.2	137.9
	Avg.	5383	8045			7891			7884			7876		
	Avg.time			169	N/A		2708	N/A		82.7	40.7		81.1	42.7
	Avg. scaled time			169	N/A		3249	N/A		141	69		138	73
	Nb. Opt		16/72			13/72			16/72			18/72		

Tableau 3-7
Résultats pour le jeu de paramètres $SP = 1, (1{,}300{,}300)$

SP	DS	C	S	LB_{DS}^{SP}	Geismar *et al.* (2008)			Karaoğlan et Kensen (2017)			Notre proposition					
					$h_{DS,1}^{SP}$	$tt_{DS,1}^{SP}$	$t_{DS,1}^{SP}$	$h_{DS,1}^{SP}$	$tt_{DS,1}^{SP}$	$t_{DS,1}^{SP}$	$\overline{h_{DS,5}^{SP}}$	$\overline{tt_{DS,5}^{SP}}$	$\overline{t_{DS,5}^{SP}}$	h_{DS}^{SP}	tt_{DS}^{SP}	t_{DS}^{SP}
1	1	40	100	8211	8211	N/A	N/A	8213	20.9	N/A	8211	0.2	0.2	8211	0.1	0.1
1	2	40	200	7658	7915	N/A	N/A	7967	3600.0	N/A	7935	39.7	13.8	7922	39.7	6.5
1	3	40	300	7700	10701	N/A	N/A	10731	36000.0	N/A	10701	45.3	18.1	10701	44.9	16.0
1	4	50	100	9994	9994	162	N/A	9994	0.5	N/A	9994	0.2	0.2	9994	0.2	0.2
1	5	50	200	9891	11017	175	N/A	11030	3600.0	N/A	11055	64.7	41.8	11018	65.3	54.1
1	6	50	300	9231	12401	169	N/A	12357	299.7	N/A	12336	95.5	57.1	12318	95.1	83.7
		Avg.		8781	10040			10049			10039			10027		
		Avg.time				N/A			1853.5	N/A		40.7	21.8		40.8	26.7
		Avg. scaled time				N/A			2224.2	N/A			37.1		69.5	45.5
		Nb. Opt			2/6			1/6			2/6			2/6		

Les résultats obtenus pour le premier jeu de paramètre $SP = 1$, *i.e.* $SP = (1{,}300{,}300)$ sont détaillés dans le Tableau 3-7. Les seules valeurs moyennes restantes dans le Tableau 3-7 sont dues aux 5 réplications pour notre méthode.

L'efficacité du GRASP×ELS est plus importante sur les instances avec $Q = 600$ (correspondant aux instances avec *SP* variant de 7 à 12) et plus particulièrement pour les instances ayant un taux de production élevé ($SP = 9$ et 12)(Tableau 3-8). En effet, pour les instances avec *SP* variant de 7 à 12, le temps de transport « domine » le temps de production, comme souligné par (Geismar *et al.,* 2008). Par conséquent, en améliorant la résolution liée à la construction des tournées, la solution finale peut être nettement améliorée.

Dans le Tableau 3-8, avec $SP = 12$, l'écart est réduit de 45,2% pour (Geismar *et al.*, 2008) à 39.3% pour notre proposition. Avec l'algorithme de (Karaoğlan et Kesen, 2017) l'écart est aussi réduit à 39.2%.

Tableau 3-8
Comparaison des différentes méthodes

SP	$\overline{LB_6^{SP}}$	Geismar *et al.*		Karaoğlan et Kesen		Notre prop.	
		$\overline{h_{6,1}^{SP}}$	%Gap	$\overline{h_{6,1}^{SP}}$	%Gap	$\overline{h_{6,5}^{SP}}$	%Gap
1	8781	10040	12.5	10049	12.6	10039	12.5
2	4412	9157	51.8	9179	51.9	9148	51.7
3	2956	9125	67.6	9153	67.7	9118	67.5
4	8781	10041	12.5	10048	12.6	10026	12.4
5	4412	9171	51.9	9180	51.9	9151	51.7
6	2956	9153	67.7	9152	67.7	9122	67.6
7	8781	8781	0.0	8782	0.0	8782	0.0
8	4412	5744	23.1	5346	17.5	5347	17.5
9	2956	5421	45.4	4887	39.5	4919	39.9
10	8781	8781	0.0	8782	0.0	8781	0.0
11	4412	5724	22.9	5276	16.4	5300	16.7
12	2956	5403	45.2	4861	39.2	4875	39.3
Avg.	5383	8045	33.4	7891	31.4	7884	31.4
Nb. Opt		16/72		13/72		18/72	

Pour le cas de PTSP à un véhicule, notre méthode permet d'obtenir 18 solutions optimales, contre 16 solutions optimales pour (Geismar *et al.*, 2008) (Tableau 3-6). Notre méthode surpasse également la méthode de (Karaoğlan et Kesen, 2017) permettant d'obtenir seulement 13 solutions optimales.

3.6.2 Résultats numériques pour le PTSP avec plusieurs véhicules

Dans cette section, des résultats sur le cas à plusieurs véhicules sont présentés. Une première série de test a été effectuée sur des instances obtenues à partir des instances de (Geismar *et al.*, 2008). Une deuxième série de test a été réalisée sur de nouvelles instances, spécialement conçues pour le problème, afin d'observer les interactions entre les deux problèmes intégrés.

3.6.2.1 Extensions des instances de Geismar et al. (2008)

Dans cette section nous avons choisi d'étendre les instances de (Geismar *et al.*, 2008) à plusieurs véhicules, pour évaluer l'impact du transport sur les solutions du PTSP. L'ensemble des instances, ainsi que les solutions présentées dans le Tableau 3-9 sont disponibles et téléchargeables à l'adresse suivante :

http://fc.isima.fr/~vinot/Research/PTSP_Results.html

Sur le Tableau 3-9, on peut remarquer qu'une augmentation du nombre de véhicules de un à deux permet de réduire la valeur moyenne du makespan des solutions de 25% (de 7884 à 5820). De la même façon, en augmentant le nombre de véhicules de deux à trois, on observe une diminution du makespan moyen de 6% (de 5820 à 5478).

Il faut remarquer qu'une flotte de 6 véhicules est suffisante pour atteindre les bornes inférieures de chacune des instances avec les contraintes les plus fortes, $r \in \{1,2,3\}$, $Q = 300$ et $B = 300$. Ces instances correspondent aux trois premières lignes du Tableau 3-9 et correspondent aux paramètres $SP = \{1,2,3\}$. La valeur moyenne des solutions obtenues pour le cas 6 véhicules sur les instances caractérisées par $SP = \{1,2,3\}$ est égale à la valeur moyenne des bornes inférieures.

Le Tableau 3-10 donne le détaille de l'ensemble des solutions pour les instances avec $SP = 3$ et pour $DS = [\![1,6]\!]$ avec 2, 3, 4, 5 ou 6 véhicules. Pour le cas 6 véhicules, on peut remarquer que toutes les solutions sont optimales puisque la borne inférieure est atteinte à chaque fois.

Tableau 3-9
Résultats avec 2, 3, 4, 5 et 6 véhicules.

SP	DS	$\overline{LB_6^{SP}}$	2 véhicules			3 véhicules			4 véhicules			5 véhicules			6 véhicules		
			$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$
1	1..6	8781	8781	0.4	0.4	8781	0.3	0.3	8781	0.4	0.4	8781	0.4	0.4	**8781**	0.5	0.5
2	1..6	4412	5114	37.6	17.1	4452	22.5	9.6	4412	5.6	2.9	4412	1.4	1.4	**4412**	2.1	2.1
3	**1..6**	**2956**	**4658**	**51.3**	**24.6**	**3461**	**41.4**	**19.2**	**3061**	**37.0**	**22.3**	**2968**	**29.6**	**15.4**	**2956**	**16.4**	**14.0**
4	1..6	8781	8781	0.5	0.5	8781	0.5	0.5	8781	0.5	0.5	8781	0.5	0.5	8781	0.6	0.6
5	1..6	4412	5116	40.2	21.9	4435	27.0	7.9	4412	2.4	2.4	4412	1.8	1.8	4412	2.8	2.8
6	1..6	2956	4674	52.8	21.4	3467	45.5	20.3	3059	31.7	18.3	2969	35.9	15.1	2956	5.5	5.5
7	1..6	8781	8781	2.6	2.6	8781	1.7	1.7	8781	1.7	1.7	8781	1.8	1.8	8781	1.9	1.9
8	1..6	4412	4426	69.5	41.7	4419	50.0	28.1	4413	49.4	30.7	4412	43.6	25.2	4412	47.2	38.5
9	1..6	2956	3175	141.8	85.5	2988	130.5	56.2	2973	102.5	54.1	2968	105.8	51.3	2968	107.7	58.4
10	1..6	8781	8781	1.6	1.6	8781	1.3	1.3	8781	1.3	1.3	8781	1.3	1.3	8781	1.3	1.3
11	1..6	4412	4422	53.6	22.7	4412	52.4	39.0	4414	57.4	26.8	4414	60.8	28.3	4414	61.8	28.7
12	1..6	2956	3137	150.7	92.7	2979	123.1	41.4	2969	114.7	51.5	2968	133.1	65.7	2969	135.7	48.6
Avg.		5383	5820			5478			5403			5387			5385		
Avg.time				50.2	27.7		41.36	18.8		33.71	17.7		34.7	17.4		31.9	16.9
Nb. Opt			41/72			53/72			63/72			63/72			66/72		

Tableau 3-10
Résultats détaillés pour $SP = 3, (3{,}300{,}300)$ avec 2, 3, 4, 5 et 6 véhicules.

SP	DS	$\overline{LB_6^{SP}}$	2 véhicules			3 véhicules			4 véhicules			5 véhicules			6 véhicules		
			$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$	$\overline{h_{6,5}^{SP}}$	$\overline{tt_{6,5}^{SP}}$	$\overline{t_{6,5}^{SP}}$
3	1	2756	3082	20.5	8.6	2756	1.9	1.9	2756	0.5	0.5	2765	2.7	2.7	2756	2.9	2.9
3	2	2589	3968	35.7	17.5	2787	39.4	20.6	2589	11.0	10.2	2589	4.0	4.0	2589	3.4	3.4
3	3	2600	5373	43.6	17.6	3704	46.5	24.7	2939	49.1	38.2	2632	58.8	31.3	2600	8.4	8.4
3	4	3363	3736	55.0	20.2	3363	0.7	0.7	3363	0.7	0.6	3363	0.8	0.8	3363	0.9	0.8
3	5	3305	5551	58.7	44.2	3856	63.1	36.5	3318	62.3	37.4	3305	2.5	2.5	3305	2.1	3.1
3	**6**	**3124**	**6238**	**94.1**	**39.4**	**4301**	**96.9**	**30.8**	**3400**	**98.5**	**47.1**	**3165**	**108.5**	**51.0**	**3124**	**80.6**	**66.6**
Avg.		2956	4658			3461			3061			2968			2956		
Avg.time				51.3	24.6		41.4	19.2		37.0	22.3		29.6	15.4		16.4	14.0
Nb. Opt			0/6			2/6			3/6			4/6			6/6		

3.6.2.2 Analyse d'une instance de (Geismar et al., 2008) : instance où le temps moyen de production est très faible face au temps moyen de transport

Il existe quelques instances pour lesquelles le temps moyen de production est très faible face au temps moyen de transport. Ces instances se caractérisent par :

- un taux de production élevée ($r = 3$), engendrant un ratio moyen entre les temps de production et des temps de transport élevé, parfois supérieur à 4 ou 5.
- des solutions dans lesquelles, le nombre de jobs est quasiment identique au nombre de clients (très peu de clients par tournées).

L'instance caractérisée par $SP = 3$, $DS = 6$ de (Geismar *et al.*, 2008) défini avec un véhicule par les auteurs, possède un taux de production de 3 et entre dans cette catégorie. De plus, le temps moyen pour produire la demande d'un client est de l'ordre de 60 unités de temps alors que le temps moyen de transport est de l'ordre de 300 unités de temps (ceci donne un ratio de 5). Dans cette instance, la plus petite quantité demandée par un client est égale à 101, la capacité des véhicules étant égale à 300, chaque tournée peut transporter au plus deux demandes. De plus, parmi les 50 clients, seuls 32 peuvent être livrés dans la même sous-tournée qu'un autre client (somme des quantités demandées inférieure à 300). La solution peut donc contenir au plus 16 sous-tournées composées de deux clients et au minimum 18 sous-tournées composées d'un seul client. Un autre paramètre accentue ce phénomène de tournées composées d'un seul client : il s'agit de la durée de vie du produit. Dans cette instance ($SP = 3$, $DS = 6$) la durée de vie du produit est égale à 300, ce qui contraint indirectement le nombre de client par sous-tournée.

Nous pouvons donc conclure que cette instance traduit un cas où la production est dominée par le transport, c'est-à-dire que la durée des opérations de production est nettement inférieure à la durée des opérations de transport. L'augmentation du nombre de véhicules disponibles permet, dans une certaine mesure, de contrebalancer ce phénomène en parallélisant les opérations de transport.

Analyse d'une solution avec un nombre « suffisant » de véhicules

À partir d'un certain nombre n^* de véhicules, on obtient des solutions pour lesquelles il n'y a plus aucun temps d'attente sur la production, ce qui permet de retrouver un makespan identique à la valeur de la borne inférieure. Ce nombre de véhicules correspond à la taille « optimale » de la flotte de véhicule, dans le sens où un ajout d'un véhicule supplémentaire ne permet pas d'améliorer la solution (diminuer le makespan). Pour l'instance caractérisée par $SP = 3$ et $DS = 6$, la valeur $n^* = 6$ permet d'atteindre la borne inférieure, on peut dire que ce nombre de véhicule est suffisant.

Lorsque le nombre de véhicules est égal à 6, on obtient une solution optimale de makespan égal à 3124.34 arrondi à 3124 présentée sur le Tableau 3-10 et la Figure 3-17. Cette solution est composée de 47 jobs permettant la réalisation de la production et de la livraison des demandes de 50 clients. On peut donc noter que la grande majorité des sous-tournées ne sont composées que d'un seul client.

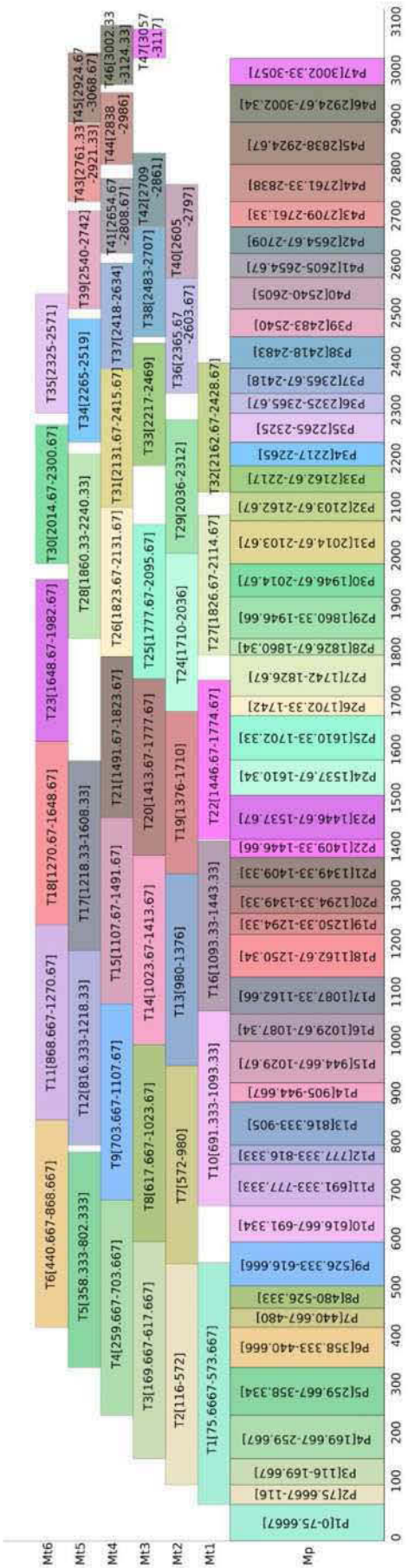


Figure 3-17. Diagramme de Gantt d'une solution optimale pour l'instance avec $SP = 3$, $DS = 6$ et 6 véhicules

La Figure 3-18 présente les 10 premiers jobs (opérations de production et de transport) d'une solution optimale pour l'instance avec SP = 3, $DS = 6$ et 6 véhicules. Sur cette figure, les 6 premières opérations de transport sont en no-wait étant toutes affectées à des véhicules différents. Les opérations de transport 7, 8, et 9 sont, quant à elles, décalées par rapport à la fin des opérations de production. Ce temps d'attente entre la fin de la production et le début du transport est autorisé mais limité par la durée de vie du produit transporté. Par exemple, sur la Figure 3-18, l'opération de production du job 9 est achevée à la date 616.67 et l'unique client de la tournée est livré à la date 905.67. Le produit va donc attendre pendant 289 unités de temps avant d'être livré, ce qui est en accord avec la contrainte sur la durée de vie B du produit qui est fixée à 300 unités de temps pour cette instance.

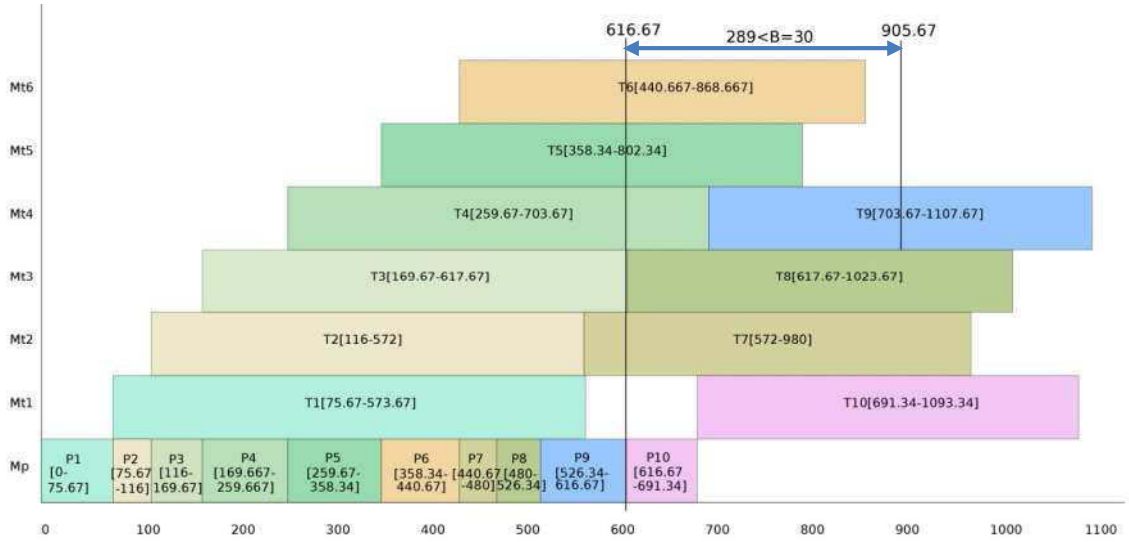


Figure 3-18. Diagramme de Gantt partiel d'une solution optimale pour l'instance avec $SP = 3$, $DS = 6$ et 6 véhicules

Impact du nombre de véhicules

Le *Tableau 3-11* permet d'observer l'impact du nombre de véhicules sur la solution pour l'instance avec SP = 3, $DS = 6$. Ce tableau permet de faire la synthèse des points suivants en fonction du nombre de véhicules :

- C_{max} : le makespan
- nj : le nombre de job
- τ_p : le taux d'utilisation de la machine de production (entre le début et la fin de la production)
- τ_t : le taux d'utilisation moyen des véhicules (entre le début et la fin des tournées)

Les informations regroupées dans le *Tableau 3-11* correspondent à la meilleure solution obtenue pour chaque véhicule, correspondant à une réplication. Les taux sont exprimés en pourcentage.

Tableau 3-11
Comparaison des solutions de l'instance avec $SP = 3$ et $DS = 6$ en fonction du nombre de véhicules

	1 véhicule	2 véhicules	3 véhicules	4 véhicules	5 véhicules	6 véhicules
C_{max}	12204	6215	4276	3383	3147	3124
nj	37	39	37	38	41	74
τ_p	26.1	52.0	76.8	97.8	99.9	100.0
τ_t	100.0	99.9	99.8	98.8	97.6	97.6

Avec un véhicule le makespan est égal à 12 204, et en ajoutant un véhicule le makespan est réduit de 50%. En passant de deux à trois véhicules le makespan est réduit de 31%, de trois à quatre véhicules de 21%, de quatre à cinq véhicules de 7% et de cinq à six véhicules de moins de 1%.

La différence majeure, observable entre les 6 configurations résumées sur le *Tableau 3-11* se trouve au niveau de la production. Avec peu de véhicules, la production est discontinue avec de nombreux temps d'attente, le taux d'utilisation de la machine de production est donc très faible. En effet, avec deux véhicules, la production s'étend de la date 0 à la date 5890.33, pour une durée effective de production égale à 3057. La machine de production est donc utilisée seulement 52% du temps. Ces temps d'attente sont visibles pendant toute la durée de production et le début d'une solution avec deux véhicules est représenté sur la Figure 3-19. On constate que sur les 7 premières opérations de production, entre les dates 136.34 et 226.34 la machine de production est inutilisée (entre les opérations numéro 2 et 3). Entre la première et la septième opération de production, trois périodes d'inactivité apparaissent au niveau de la production. A l'inverse, sur les véhicules aucun temps d'attente ne peut être remarqué entre les sous-tournées.

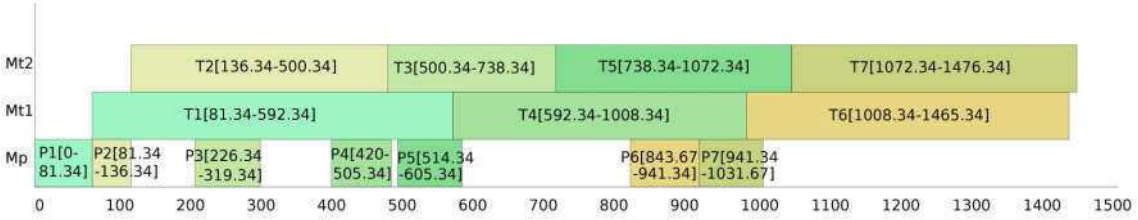


Figure 3-19. Diagramme de Gantt d'une solution pour l'instance avec $SP = 3$, $DS = 6$ et 2 véhicules

Les périodes d'inactivité de la machine de production sont particulièrement importantes dans le cas de l'instance avec SP = 3 et DS = 6 avec deux véhicules car on se trouve dans la situation où, d'une part les temps de transport sont élevés par rapport aux temps de production ($r = 3$) et d'autre part le nombre de véhicule est très faible avec des capacités limitées seulement à 300 unités de production.

Ces périodes d'inactivité sur la production sont fortement réduites dans le cas à quatre véhicules, avec un taux d'utilisation de plus de 97%, et sont totalement absentes dans le cas à 6 véhicules puisque la machine de production est utilisée 100% du temps.

La présence de temps d'attente sur les transports est due à des activités ordonnancées en no-wait par rapport à la production. Or lorsque le transport domine fortement (cas avec un et deux véhicules pour l'instance avec $SP = 3$ et $DS = 6$), très peu d'opérations sont ordonnancées en no-wait, avec un ordonnancement au plus tôt, dû à la différence entre la durée des opérations de production et la durée des opérations de transport (Figure 3-19).

L'exemple de cette instance est particulièrement adapté au cas à plusieurs véhicules. En effet, sur cette instance où le transport domine la production, l'augmentation du nombre de véhicules permet de tendre vers un problème plus équilibré. On peut ainsi mesurer l'efficacité de l'algorithme et observer les caractéristiques des solutions et des instances en fonction du nombre de véhicules.

La majorité des instances de (Geismar *et al.*, 2008) n'ont pas les même caractéristiques que l'exemple étudié dans cette section. Une analyse plus complète et globale des instances de (Geismar *et al.*, 2008) montre que :

1) le ratio entre le temps de moyen de production et le temps moyen de transport est élevé, ce qui caractérise des instances ayant en majorité une dominance des temps de production face aux temps de transport ;
2) le ratio entre la demande moyenne des clients et la capacité des véhicules est environ égal à 0.3 pour toutes les instances, par conséquent le nombre moyen de clients par sous-tournées reste faible, environ deux clients par tournée ;
3) la répartition uniforme des clients sur une zone géographique avec un dépôt localisé au centre de la zone engendre des instances théoriques.

Les instances de (Geismar *et al.*, 2008) sont dont parfaitement adaptées au cas à un véhicule, mais ne permettent pas de faire des tests complets avec un grand nombre de véhicules transportant des produits pour un nombre important de clients. De nouveaux jeux d'instances ont été introduits, permettant notamment la création d'instances de plus grandes tailles.

3.6.2.3 Introduction de nouvelles instances

Pour avoir des configurations plus réalistes, un nouvel ensemble d'instances a été généré avec différents paramètres, dont :

- une distribution des clients en zones, pour modéliser des zones urbaines et rurales ;
- un dépôt localisé aléatoirement (au centre, à la périphérie, etc.) ;
- des véhicules avec des capacités importantes permettant d'effectuer des grandes tournées.

Chaque instance est caractérisée par :

- un nombre de clients, déterminé grâce à une distribution uniforme $U(50,100)$ pour les grandes instances ou $U(100,200)$ pour les très grandes instances ;
- une zone d'intérêt qui contient tous les clients ainsi que le dépôt. La longueur X et la largeur Y de cette zone peut varier entre 100 et 500, créant 25 possibilités (100×100, 100×200, …, 500×400, 500×500). Cette zone est centrée sur les

coordonnées (0,0), elle est divisée en quatre secteurs $[-X, 0] \times [0, Y]$, $[0, X] \times [0, Y]$, $[0, X] \times [-Y, 0]$ et $[-X, 0] \times [-Y, 0]$ (Figure 3-20);
- un nombre de zones client $c \in \{1,2,3\}$.

Avec tous ces paramètres, 75 instances de grande taille et de très grande taille ont été générées (25 possibilités pour la taille de la zone d'intérêt et 3 possibilités pour le nombre de zones client). Une instance est donc caractérisée par un triplet (x, z, c) avec, $x \in \{G, TG\}$ permettant d'indiquer si l'instance est de grande taille G ou de très grande taille TG, $z \in [\![1,25]\!]$ permettant de définir la configuration de la zone d'intérêt et $c \in \{1,2,3\}$ pour définir le nombre de zones client. Lorsque $z = 1$, la zone d'intérêt est de taille 100×100, pour $z = 2$, la zone d'intérêt est de taille 100×200, etc. Pour $z = 25$, la zone d'intérêt est par conséquent de taille 500×500.

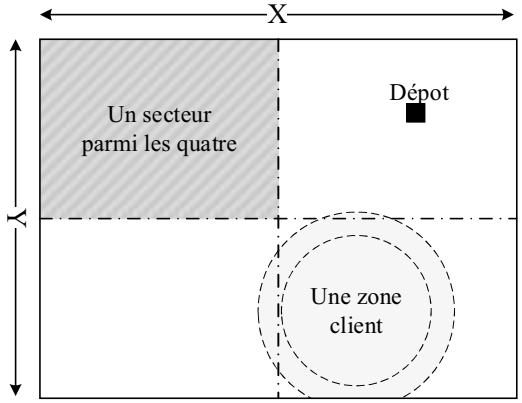
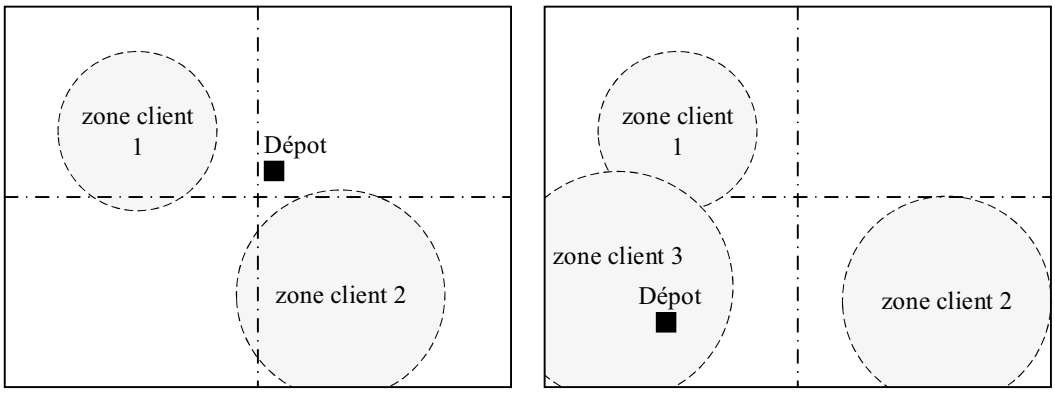


Figure 3-20. Représentation des paramètres d'une instance.

La Figure 3-21 illustre deux exemples de configuration d'instance. Dans le premier exemple (Figure 3-21.a), l'instance est composée de deux zones clients et d'un dépôt centré (avec une petite variation) par rapport à la zone d'intérêt de taille [X ; Y]. Dans le deuxième exemple (Figure 3-21.b), l'instance est composée de trois zones clients de taille et de localisation aléatoires et d'un dépôt centré (avec une petite variation) par rapport à la zone client 3.



(a): Deux zones clients avec dépôt centré (b) : Trois zones clients avec dépôt excentré

Figure 3-21. Deux exemples de configuration d'instances.

D'autres informations sont nécessaires pour définir complètement une instance :

- la capacité des véhicules, égale à 1000 ;
- la demande de chaque client, choisie parmi les valeurs $\{50, 100, 200, 300, 500\}$ avec les probabilités $\{0.1, 0.2, 0.4, 0.2, 0.1\}$;
- la localisation de chaque client, choisie aléatoirement dans une zone client ;
- la localisation de chaque zone client, aléatoirement sélectionné parmi les centres des quatre secteurs avec une variation de 10%;
- la taille de chaque zone client, égale à la distance minimum entre le centre de la zone client et le bord de la zone d'intérêt avec une variation de 20% ;
- la localisation du dépôt peut être soit centré avec une probabilité de 0.3%, soit excentré avec une probabilité de 0.7%. Lorsque le dépôt est excentré, il pourra être localisé au niveau du centre d'un des quatre secteurs avec une variation de 10%. Sinon, il sera au centre de la zone d'intérêt avec une variation de 10%.

À partir de chacune des instances définies précédemment, et pour obtenir des configurations permettant de jouer sur la dominance entre la production et le transport en fonction du nombre de véhicules, deux types d'instances sont créés. Ces deux types sont définis par deux paramètres r et B, le taux de production et la durée de vie du produit, également utilisés par (Geismar *et al.*, 2008).

Le type 1, avec $r = 1$ et $B = 1200$, modélise des situations où les temps de production sont du même ordre de grandeur que les temps de transport, et pour lesquelles la durée de vie du produit est très importante par rapport aux distances entre les clients. Ce qui engendre des situations où avec un faible nombre de véhicules (1 ou 2) on peut obtenir des solutions de coût identique à la borne inférieure. Les solutions de bonne qualité sont des solutions dans lesquelles le nombre de clients par tournée peut être important par rapport aux instances de (Geismar *et al.*, 2008).

Le type 2, avec $r = 4$ et $B = 600$, modélise des situations où les temps de production sont plus grands que les temps de transport, et pour lesquelles la durée de vie du produit contraint la production et le transport. Par conséquent, pour trouver une solution de coût identique à la borne inférieure (rappelons que la borne est obtenue en relâchant la contrainte sur le nombre de véhicules), il faut utiliser un nombre de véhicules plus grand que pour les types 1 et 2 (en général sur nos instances autour de 4 véhicules).

Tableau 3-12
Caractéristiques des deux types d'instances et prévisions des solutions

	Paramètres		Caractéristiques probables des bonnes solutions	
	r	B	Nombre moyen de véhicules	Nombre maximal de clients par tournées
Type 1	1	1200	1	9
Type 2	4	600	3	6

Ces deux types d'instances sont proposés sur les grandes instances (entre 50 et 100 clients). Les très grandes instances (entre 100 et 200 clients) sont générées uniquement pour le type 1.

Au total, 150 instances de grandes tailles sont générées ($25 \times 3 = 75$ de type 1 et $25 \times 3 = 75$ de type 2) et 75 instances de très grande taille sont générées appartenant toutes au type 1. Un schéma récapitulatif en Figure 3-22 illustre les caractéristiques de toutes les instances générées.

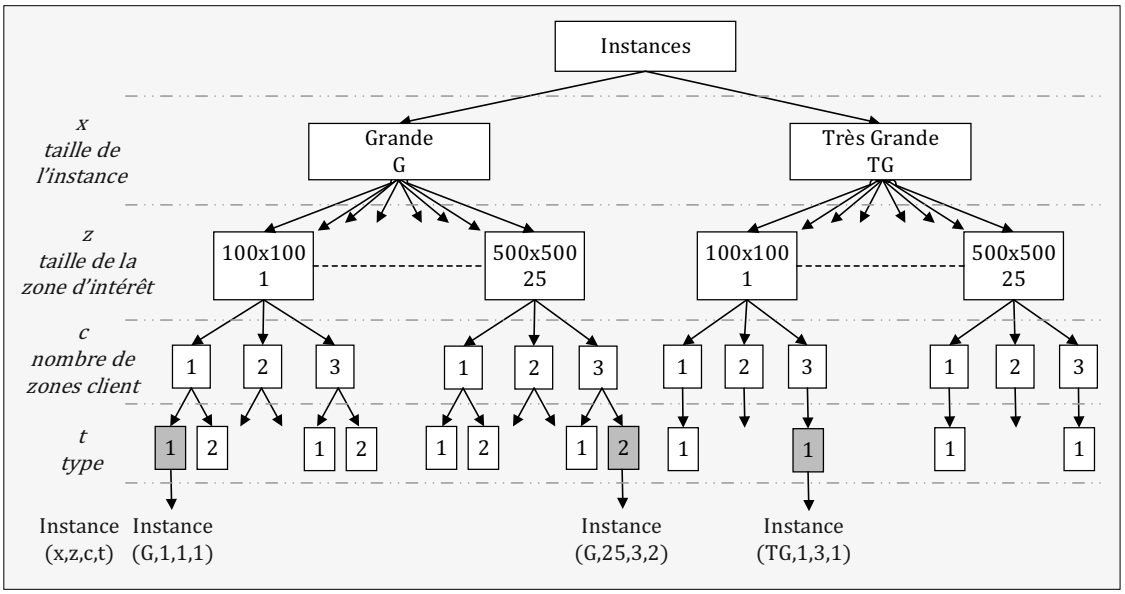


Figure 3-22. Construction des nouvelles instances.

Afin de définir complétement une instance le triplet (x, z, c) caractérisant une instance est tranformé en un quadruplet (x, z, c, t) avec $t \in \{1,2\}$ permettant de donner le type de l'instance. Lorsqu'un indice est remplacé par $*$, comme dans le quadruplet $(x, z, *, t)$, cela indique que l'indice peut prendre n'importe quelle valeur de son ensemble.

3.6.2.4 Résultats numériques sur les nouvelles instances pour le type 1

Le paramétrage du GRASP×ELS est le même pour toutes les instances et ce indépendamment du nombre de zones client ou du nombre de véhicules (Tableau 3-13).

Tableau 3-13
Paramétrage du GRASP×ELS pour les nouvelles instances.

Paramètre	Définition	Valeur
np	Nombre d'itérations pour le GRAPS	50
ne	Nombre d'itérations pour le ELS	10
nd	Nombre de voisins générés	5
lr	Nombre d'itérations pour la recherche locale sur les tournées	40
ns	Nombre d'itérations pour la recherche locale sur l'ordonnancement	500
NBmax	Nombre maximal de labels par nœud	5

Cinq réplications sont effectuées sur chaque instance. Le Tableau 3-14 donne les moyennes des meilleures solutions obtenues sur les cinq réplications et sur les trois types de zone clients (colonne $\overline{h_{3,5}}$), la moyenne des temps CPU (colonne $\overline{tt_{3,5}}$) et la moyenne

des temps CPU pour trouver la meilleure solution (colonne $\overline{t_{3,5}}$), dans le cas à un véhicule. La borne inférieure fait référence à la borne inférieure introduite dans la partie 3.4.8. La moyenne des bornes inférieures est effectuée sur les trois types de zone clients (colonne $\overline{LB_3}$).

Pour le cas à un véhicule, le GRASP×ELS sur les grandes instances fournit des solutions en un temps moyen de 1.13 secondes contre en moyenne 6.63 seconds pour les très grandes instances. Entre les deux types d'instances, le temps de calcul a été multiplié par six. De plus, en moyenne, l'écart entre nos résultats et la borne inférieure est très petit, 0.09% pour les grandes instances et 0.03% pour les très grandes instances. Toutes les tournées ainsi qu'une représentation graphique de l'instance 1 sont disponibles sur :

http://fc.isima.fr/~vinot/Research/PTSP_Results.html

En passant de un à deux véhicules, l'écart entre les résultats et la borne inférieure passe de 0.09% à 0.005% pour les grandes instances et de 0.03% à 0.003% pour les très grandes instances. Les temps de calculs pour le cas à deux véhicules sont nettement inférieurs (environ divisés par dix), ceci étant dû au grand nombre d'instances pour lesquelles la borne inférieure est atteinte.

Les expérimentations numériques effectuées avec trois et quatre véhicules permettent d'affirmer que toutes les bornes inférieures des instances de type 1 sont atteignables. Une flotte de quatre véhicules est suffisante pour les grandes instances et une flotte de trois véhicules est suffisante pour les très grandes instances.

Pour les grandes instances et dans le cas de deux véhicules, toutes les solutions sur les instances avec trois zones clients sont optimales, toutes les bornes inférieures sont atteintes. Le Tableau 3-15 donne le détail des solutions des instances $(G, 25, \{1,2,3\}, 1)$ correspondant aux moyennes obtenues dans la dernière ligne du Tableau 3-14 a.

Le Tableau 3-15 donne les meilleures solutions obtenues sur les cinq réplications et sur les trois types de zone clients (colonne $\overline{h_{1,5}}$), les temps CPU totaux (colonne $\overline{tt_{1,5}}$) et les temps CPU pour trouver la meilleure solution (colonne $\overline{t_{1,5}}$), dans le cas à un ou deux véhicules. La moyenne des bornes inférieures est fournie pour les trois types de zones clients (colonne $\overline{LB_1}$).

Tableau 3-14
Résultats sur les grandes et très grandes instances pour le type 1 avec 1, ou 2 véhicules.

a. Les grandes instances

		1 véhicule			2 véhicules			
Inst.	$\overline{LB_3}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{v_{best}}$
(G,1,*,1)	15119	15119	0.01	0.00	15119	0.00	0.00	1.0
(G,2,*,1)	17572	17572	0.00	0.00	17572	0.01	0.00	1.0
(G,3,*,1)	15929	15929	0.01	0.01	15929	0.00	0.00	1.0
(G,4,*,1)	12396	12437	3.63	0.03	12396	0.00	0.00	2.0
(G,5,*,1)	17458	17459	2.06	0.01	17458	0.00	0.00	1.3
(G,6,*,1)	15173	15173	0.01	0.00	15173	0.00	0.00	1.0
(G,7,*,1)	15985	15985	0.00	0.00	15985	0.00	0.00	1.0
(G,8,*,1)	15851	15857	1.57	0.00	15851	0.00	0.00	1.3
(G,9,*,1)	19815	19815	0.01	0.00	19815	0.00	0.00	1.0
(G,10,*,1)	16333	16346	3.36	0.04	16333	0.00	0.00	1.7
(G,11,*,1)	17774	17774	0.00	0.00	17774	0.00	0.00	1.0
(G,12,*,1)	15607	15608	1.32	0.03	15607	0.01	0.00	1.3
(G,13,*,1)	15235	15331	2.53	0.28	15235	0.03	0.03	1.7
(G,14,*,1)	13497	13501	1.28	0.00	13497	0.01	0.00	1.3
(G,15,*,1)	15425	15431	1.16	0.01	15425	0.01	0.00	1.3
(G,16,*,1)	13687	13688	1.04	0.00	13687	0.00	0.00	1.3
(G,17,*,1)	14273	14273	0.06	0.06	14273	0.00	0.00	1.0
(G,18,*,1)	14449	14452	1.30	0.00	14449	0.00	0.00	1.3
(G,19,*,1)	16579	16579	0.00	0.00	16579	0.01	0.01	1.0
(G,20,*,1)	14415	14442	2.32	0.20	14415	0.00	0.00	1.7
(G,21,*,1)	17012	17012	0.00	0.00	17012	0.00	0.00	1.0
(G,22,*,1)	16300	16312	1.66	0.33	16300	0.01	0.00	1.3
(G,23,*,1)	14421	14530	1.48	0.00	14439	2.48	0.06	1.7
(G,24,*,1)	17364	17364	0.00	0.00	17364	0.01	0.00	1.0
(G,25,*,1)	**14167**	**14188**	**3.33**	**0.74**	**14167**	**0.01**	**0.00**	1.7
Avg.	15673	15687			15674	0.10	0.00	1.3
Gap LB		0.09%			0.005%			
Avg.time			1.13	0.07		0.10	0.00	
Nb. Opt		55/75			74/75			

b. Les très grandes instances

		1 véhicule			2 véhicules			
Inst.	$\overline{LB_3}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{v_{best}}$
(TG,1,*,1)	27859	27859	0.02	0.01	27859	0.02	0.01	1.0
(TG,2,*,1)	30401	30401	0.02	0.01	30401	0.02	0.01	1.0
(TG,3,*,1)	37341	37341	0.03	0.03	37341	0.03	0.03	1.0
(TG,4,*,1)	34229	34229	0.03	0.02	34229	0.02	0.01	1.0
(TG,5,*,1)	23003	23069	8.64	6.57	23018	8.51	0.02	1.7
(TG,6,*,1)	34625	34625	0.02	0.02	34625	0.02	0.02	1.0
(TG,7,*,1)	34923	34923	0.03	0.02	34923	0.02	0.02	1.0
(TG,8,*,1)	32175	32175	0.02	0.02	32175	0.02	0.02	1.0
(TG,9,*,1)	35041	35046	13.59	0.02	35041	0.03	0.02	1.3
(TG,10,*,1)	34739	34751	27.00	0.03	34739	0.02	0.02	1.7
(TG,11,*,1)	31357	31357	0.02	0.02	31357	0.02	0.02	1.0
(TG,12,*,1)	32843	32843	0.03	0.02	32843	0.02	0.02	1.0
(TG,13,*,1)	34371	34371	0.02	0.02	34371	0.02	0.02	1.0
(TG,14,*,1)	32165	32179	32.41	0.02	32165	0.02	0.02	2.0
(TG,15,*,1)	31361	31361	8.27	0.34	31361	0.02	0.02	1.3
(TG,16,*,1)	30529	30529	0.02	0.02	30529	0.02	0.02	1.0
(TG,17,*,1)	30507	30507	0.02	0.01	30507	0.02	0.01	1.0
(TG,18,*,1)	35015	35015	0.03	0.02	35015	0.03	0.02	1.0
(TG,19,*,1)	33901	33901	0.02	0.02	33901	0.02	0.02	1.0
(TG,20,*,1)	29647	29651	6.95	0.58	29647	0.02	0.02	1.3
(TG,21,*,1)	35893	35893	0.03	0.03	35893	0.03	0.02	1.0
(TG,22,*,1)	33157	33175	24.80	0.29	33157	0.02	0.02	1.7
(TG,23,*,1)	31203	31229	11.92	0.02	31203	0.02	0.02	1.3
(TG,24,*,1)	32379	32385	8.58	1.26	32379	0.02	0.02	1.3
(TG,25,*,1)	33524	33576	23.19	11.86	33531	12.60	0.28	2.0
Avg.	32487	32496			32488			1.2
Gap LB		0.03%			0.003%			
Avg.time			6.63	0.85		0.86	0.03	
Nb. Opt		60/75			72/75			

Pour le cas à un véhicule, le GRASP×ELS fournit sur les trois instances $(G, 25,*,1)$ des solutions très proches de l'optimal, avec un écart de 0.15%. Pour le cas à deux véhicules, toutes les solutions trouvées pour les instances $(G, 25,*,1)$ sont optimales.

Tableau 3-15
Résultats sur les grandes instances avec $(25,*)$ pour le type 1 avec 1, ou 2 véhicules.

		1 véhicule			2 véhicules			
Instance	$\overline{\mathrm{LB}_1}$	$\overline{h_{1,5}}$	$\overline{tt_{1,5}}$	$\overline{t_{1,5}}$	$\overline{h_{1,5}}$	$\overline{tt_{1,5}}$	$\overline{t_{1,5}}$	$\overline{v_{best}}$
(G,25,1,1)	16798	16804	61.27	6.18	16798	7.80	7.74	2.0
(G,25,2,1)	14792	14850	61.82	3.73	14792	16.57	16.52	2.0
(G,25,3,1)	10910	10910	25.96	25.92	10910	18.02	17.96	1.0
Avg.	14167	14188			14167			1.7
Gap LB		0.15%			0.00%			
Avg.time			49.69	11.94		14.13	14.07	
Nb. Opt		1/3			3/3			

Les tournées de la solution optimale associée à l'instance $(G, 25,3,1)$ avec deux véhicules sont représentées sur la Figure 3-23, chaque tournée étant représentée par une couleur. Cette instance est caractérisée par :

- 56 clients numérotés et distribués sur une zone de dimension 500×500 ;
- un dépôt situé dans une zone client, modélisé par un carré portant le numéro 0 ;
- deux véhicules de capacité 1000.

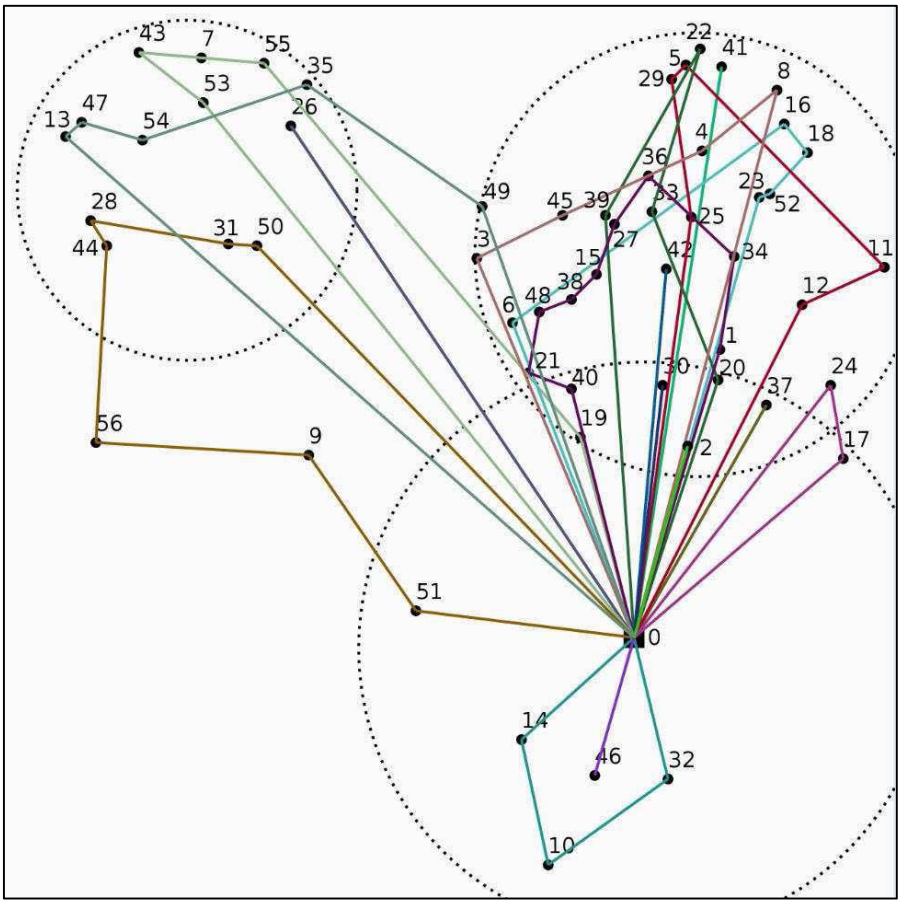
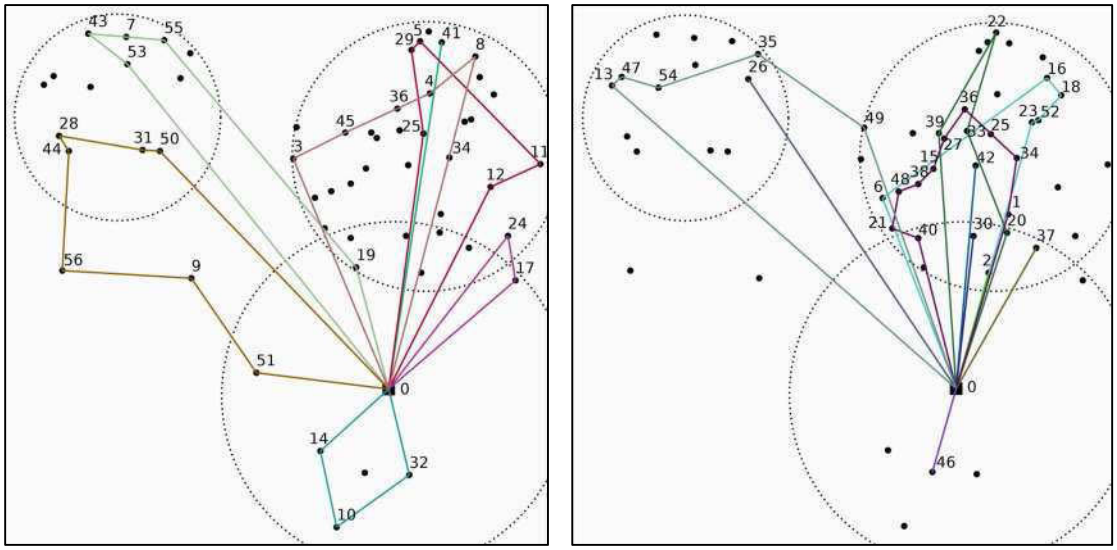


Figure 3-23. Représentation des tournées de la solution associée à l'instance $(G, 25,3,1)$ avec 56 clients et deux véhicules.

Sur la Figure 3-23, la plus grande des tournées est composée de neuf clients (1 / 34 / 36 / 27 / 15 / 38 / 48 / 21 / 40), cette tournée est effectuée par le véhicule 2 (Figure 3-24.b). Tous ces clients appartiennent à la même zone géographique (zone client dans la partie supérieure droite). La quantité totale demandée par l'ensemble de ces clients est égale à 1000, ce qui permet de charger au maximum le véhicule. Le temps de transport de cette tournée est égal à 587.



(a): Tournée du véhicule 1 *(b) : Tournée du véhicule 2*

Figure 3-24. Représentation des tournées de la solution associée à l'instance $(G, 25,3,1)$.

3.6.2.5 Résultats numériques sur les nouvelles instances pour le type 2

Le paramétrage du GRASP×ELS pour le type 2 est le même pour toutes les instances et est identique au paramétrage utilisé pour le type 1 (Tableau 3-13). Sur le Tableau 3-16, on peut remarquer qu'une augmentation du nombre de véhicules de un à deux permet de réduire la valeur du makespan moyen des solutions de près de 30% (de 6211 à 4281). De la même façon, en augmentant le nombre de véhicules de deux à trois, on observe une diminution du makespan moyen de 5% (de 4281 à 4036).

Il peut être noté qu'une flotte de sept véhicules est suffisante pour atteindre les bornes inférieures de chacune des instances. Cependant, le plus petit nombre optimal de véhicules varie d'une instance à l'autre. Pour le groupe d'instances $(G, 1,*,2)$, la borne inférieure est atteinte avec un véhicule (Tableau 3-16), ces instances sont caractérisées par une zone géographique de petite taille (100×100), par conséquent, les temps de production sont plus élevés que les temps de transport, un seul véhicule est donc suffisant. A l'inverse, pour le groupe d'instances $(G, 25,*,2)$ caractérisé par une zone géographique de très grande taille (500×500), un facteur 10 peut être observé entre le temps de production et le temps de transport pour un client. La borne inférieure pour ces instances est atteinte avec en moyenne 5.3 véhicules (Figure 3-25).

Tableau 3-16
Résultats sur les grandes instances pour le type 2 avec 1, 2, 3, 4, 5, 6 ou 7 véhicules

		1 véhicule			2 véhicules			3 véhicules			4 véhicules			5 véhicules			6 véhicules			7 véhicules			
Inst.	$\overline{LB_3}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{h_{3,5}}$	$\overline{tt_{3,5}}$	$\overline{t_{3,5}}$	$\overline{v_{best}}$
(G,1,*,2)	3794	3794	0.0	0.0	3794	0.0	0.0	3794	0.0	0.0	3794	0.0	0.0	3794	0.0	0.0	3794	0.0	0.0	3794	0.0	0.0	1.00
(G,2,*,2)	4412	4445	1.4	1.4	4419	2.4	1.0	4412	0.0	0.0	4412	0.0	0.0	4412	0.0	0.0	4412	0.0	0.0	4412	0.0	0.0	1.67
(G,3,*,2)	4035	5262	4.5	2.1	4067	2.4	0.2	4052	3.5	0.6	4036	4.2	3.3	4036	0.1	0.1	4036	0.0	0.0	4036	0.0	0.0	3.00
(G,4,*,2)	3266	5055	3.0	0.9	3312	2.9	1.4	3270	2.3	0.5	3267	2.6	0.2	3266	0.8	0.8	3266	0.0	0.0	3266	0.0	0.0	3.33
(G,5,*,2)	4434	6263	4.5	3.0	4580	3.9	0.4	4434	1.2	1.2	4434	0.0	0.0	4434	0.0	0.0	4434	0.0	0.0	4434	0.0	0.0	2.33
(G,6,*,2)	3810	3814	1.6	0.1	3810	0.0	0.0	3810	0.0	0.0	3810	0.0	0.0	3810	0.0	0.0	3810	0.0	0.0	3810	0.0	0.0	1.33
(G,7,*,2)	4060	5514	3.8	2.1	4070	4.3	0.8	4061	0.0	0.0	4061	0.0	0.0	4061	0.0	0.0	4061	0.0	0.0	4061	0.0	0.0	2.67
(G,8,*,2)	4039	5041	3.9	2.7	4055	1.8	1.7	4039	0.2	0.2	4039	0.0	0.0	4039	0.0	0.0	4039	0.0	0.0	4039	0.0	0.0	2.33
(G,9,*,2)	4990	7372	4.7	1.1	4990	1.8	1.7	4990	0.0	0.0	4990	0.0	0.0	4990	0.0	0.0	4990	0.0	0.0	4990	0.0	0.0	2.00
(G,10,*,2)	4173	6950	3.9	1.8	4290	3.8	1.2	4179	2.9	0.0	4174	0.0	0.0	4174	0.0	0.0	4174	0.0	0.0	4174	0.0	0.0	3.00
(G,11,*,2)	4474	4932	4.6	1.7	4474	0.0	0.0	4474	0.0	0.0	4474	0.0	0.0	4474	0.0	0.0	4474	0.0	0.0	4474	0.0	0.0	2.00
(G,12,*,2)	3946	4758	3.8	0.8	3962	2.2	0.6	3946	0.0	0.0	3946	0.0	0.0	3946	0.0	0.0	3946	0.0	0.0	3946	0.0	0.0	2.33
(G,13,*,2)	3933	6900	3.7	1.4	4230	3.0	0.3	3974	4.0	1.1	3944	5.0	4.5	3939	3.7	1.5	3939	4.5	3.2	3933	2.2	2.2	4.33
(G,14,*,2)	3437	5358	3.3	1.3	3454	4.3	1.8	3438	2.3	0.0	3437	0.0	0.0	3437	0.0	0.0	3437	0.0	0.0	3437	0.0	0.0	3.00
(G,15,*,2)	3903	8674	3.7	2.0	5087	3.6	3.0	4097	2.9	2.6	3930	3.6	1.6	3916	5.2	5.0	3908	6.8	1.5	3903	0.2	0.2	4.00
(G,16,*,2)	3475	4276	2.1	0.9	3504	1.5	1.0	3475	0.0	0.0	3475	0.0	0.0	3475	0.0	0.0	3475	0.0	0.0	3475	0.0	0.0	2.00
(G,17,*,2)	3623	6516	3.4	0.9	3878	3.4	1.9	3664	2.3	0.8	3638	3.0	1.0	3623	0.4	0.4	3623	0.0	0.0	3623	0.0	0.0	3.33
(G,18,*,2)	3674	6469	3.5	0.8	3839	5.2	2.2	3679	3.1	1.0	3674	0.0	0.0	3674	0.0	0.0	3674	0.0	0.0	3674	0.0	0.0	3.33
(G,19,*,2)	4191	7598	4.1	0.5	4407	6.1	3.9	4192	0.0	0.0	4192	0.0	0.0	4192	0.0	0.0	4192	0.0	0.0	4192	0.0	0.0	3.00
(G,20,*,2)	3721	9508	3.6	2.1	5226	5.1	2.8	3984	6.7	4.1	3766	5.1	1.1	3721	3.0	3.0	3721	0.1	0.1	3721	0.0	0.0	4.67
(G,21,*,2)	4306	4505	3.0	0.8	4316	2.1	0.6	4306	0.0	0.0	4306	0.0	0.0	4306	0.0	0.0	4306	0.0	0.0	4306	0.0	0.0	2.00
(G,22,*,2)	4181	6499	4.1	2.2	4405	2.0	0.2	4183	2.8	2.7	4181	0.1	0.1	4181	0.1	0.1	4181	0.0	0.0	4181	0.0	0.0	2.67
(G,23,*,2)	3809	9173	3.4	1.7	5032	4.9	2.9	4090	2.7	0.4	3822	3.3	1.9	3809	0.1	0.1	3809	0.1	0.1	3809	0.0	0.0	3.67
(G,24,*,2)	4352	7099	5.0	2.6	4593	4.5	3.7	4352	0.2	0.2	4352	0.1	0.1	4352	0.1	0.1	4352	0.0	0.0	4352	0.0	0.0	2.67
(G,25,*,2)	3681	9490	3.6	1.7	5242	4.9	0.7	4007	6.5	5.4	3782	7.8	6.0	3695	3.2	0.8	3682	0.4	0.4	3682	0.2	0.2	5.33
Avg.	3989	6211			4281			4036			3997			3990			3989			3989			2.84
Gap LB (%)		55.69			7.33			1.18			0.21			0.03			0.01			0.00			
Avg. time			3.5	1.5		3.0	1.4		1.7	0.8		1.4	0.8		0.7	0.5		0.5	0.2		0.1	0.1	
Nb. Opt		10/75			35/75			58/75			64/75			72/75			73/75			75/75			

Une tendance apparaît en comparant l'aire de la zone d'intérêt d'une instance au nombre suffisant de véhicules pour retrouver les bornes inférieures. Une bonne solution du PTSP est caractérisée par un équilibre entre les temps de production et les temps de transport. Cet équilibre peut être obtenu en faisant varier le nombre de véhicules. Lorsque les temps de transport sont plus grands que les temps de production, ce qui est le cas pour les zones d'intérêt de grandes tailles, l'augmentation du nombre de véhicules permet de paralléliser les transports dans le temps. Cela a pour effet de rééquilibrer le problème par rapport aux temps de production qui sont constants et sur une seule machine. Le nombre de véhicules nécessaires à l'obtention d'une solution optimale est donc corrélé à la taille de la zone d'intérêt de l'instance (Figure 3-25).

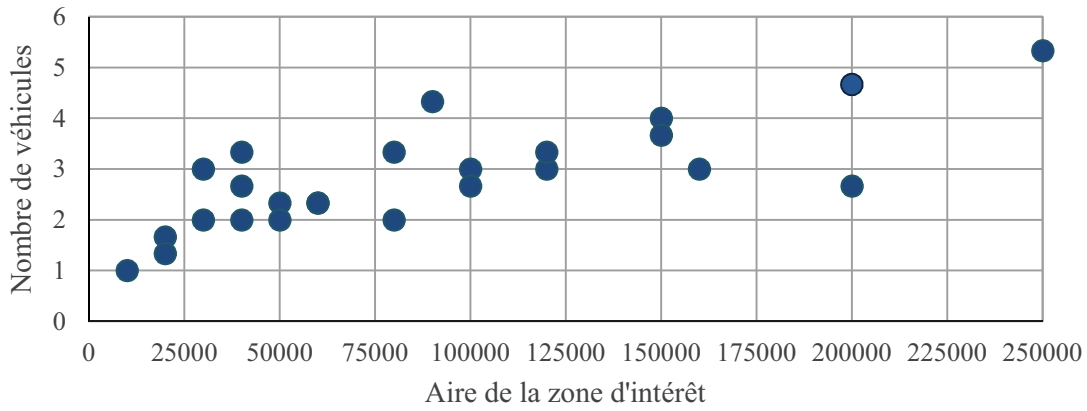


Figure 3-25. Lien entre l'aire de la zone d'intérêt et le nombre de véhicules nécessaires pour résoudre optimalement une instance.

Pour reprendre la même instance que celle détaillée dans la partie précédente, la solution optimale associée à l'instance $(G, 25{,}3{,}2)$ est obtenue avec six véhicules. Cette solution est composée de 27 jobs et le nombre moyen de clients par tournée est égal à deux. Le temps moyen de production est égal à 99 et le temps moyen de transport est égal à 522. Ces données font apparaître un ratio entre les temps de production et de transport strictement supérieur à cinq, cette solution ne peut donc pas être réalisée avec moins de six véhicules.

Dans cette solution, la plus grande des tournées est composée de six clients (3 / 45 / 39 / 27 / 33 / 25), cette tournée est effectuée par le véhicule 2. Tous ces clients appartiennent à la même zone géographique (zone client dans la partie supérieure droite). La quantité totale demandée par l'ensemble de ces clients est égale à 900, ce qui correspond à un temps de production égal à 225. Le temps de transport de cette tournée est égal à 594.

Contrairement au type 1, dans le type 2 on constate un grand nombre de tournées qui ne contiennent qu'un seul client. Cette observation valide les prévisions faites lors de la création du type 2. Dans ce type d'instances avec un taux de production $r = 4$ et une durée de vie du produit $B = 600$, les bonnes solutions sont caractérisées pas des opérations de production et de transport contenant les demandes de très peu de clients.

3.6.3 Conclusion

À travers les tableaux de cette section, nous concluons que cette nouvelle approche de résolution est meilleure que la méthode dédiée à un seul véhicule pour le cas spécifique à un véhicule sur les instances de (Geismar *et al.*, 2008). L'extension des instances de (Geismar *et al.*, 2008) au cas à plusieurs véhicules a permis de réaliser une étude plus poussée et de démontrer qu'à l'aide de six véhicules toutes les bornes inférieures peuvent être atteintes. Une solution optimale pour chaque instance avec six véhicules peut être fournie.

Pour les nouvelles instances de grandes et très grandes tailles, la métaheuristique est très efficace puisqu'elle fournit la solution optimale pour le cas à un véhicule (respectivement deux véhicules), pour le type 1, dans plus de 76% (respectivement 97%) des cas. Une solution optimale pour chaque instance a été obtenue avec au plus quatre véhicules. Les résultats sur les instances de type 2 permettent d'observer le comportement d'instances nécessitant plus de quatre véhicules disponibles.

Ces nouvelles instances permettent de tester des situations plus complexes que celles engendrées par les instances de (Geismar *et al.*, 2008). En effet, ces nouvelles instances contiennent un plus grand nombre de clients/demandes (entre 50 et 200 clients). De plus, les instances de (Geismar *et al.*, 2008) permettent la construction de tournées avec en moyenne 2.75 clients, tandis que les nouvelles instances permettent la construction de tournées avec en moyenne 6.86 clients. Des solutions de même coût que la borne inférieure sont obtenues grâce au GRASP×ELS en considérant jusqu'à sept véhicules.

3.7 Conclusion du chapitre

L'optimisation des coûts et de la satisfaction des clients passe par une plus grande efficacité des systèmes de production et de distribution et constitue un enjeu majeur dans notre société.

Nous nous sommes donc intéressé à un problème intégrant ces deux composantes avec le problème du PTSP à un ou plusieurs véhicules. Ce travail étend le problème initialement proposé par (Geismar *et al.*, 2008) en proposant une approche basée sur une représentation indirecte de la solution avec plusieurs espaces de solutions.

Nous avons proposé une modélisation mathématique du problème ainsi qu'une borne inférieure permettant d'évaluer la qualité d'une solution. L'efficacité du schéma de résolution repose sur une métaheuristique de type GRASP×ELS avec plusieurs outils, dont une procédure de type SPLIT, ainsi qu'un graphe disjonctif, tous deux spécifiques au PTSP avec plusieurs véhicules. L'originalité du SPLIT repose sur la définition de labels permettant, à chaque instant, de connaître l'état global du système, c'est-à-dire l'état de la production et de chacun des véhicules. Pour ce faire, le label contient toutes les dates de disponibilité des ressources (machine de production et machines de transport/véhicules).

Cette méthode a prouvé son efficacité sur le cas à un véhicule grâce à une comparaison avec les résultats obtenus par (Geismar *et al.*, 2008) et par (Karaoğlan et Kesen, 2017).

Cette méthode permet également d'obtenir des solutions dans le cas à plusieurs véhicules, sur les instances de (Geismar *et al.*, 2008) généralisées ainsi que sur un nouvel ensemble d'instances.

Ce nouvel ensemble d'instances permet de modéliser des zones de plus ou moins forte densité, afin de simuler par exemple des villes. De plus ces instances permettent d'obtenir des solutions, dans lesquelles, les tournées passent par un grand nombre de clients, augmentant ainsi la difficulté de résolution liée au problème de tournées de véhicules.

Il faut remarquer que ce problème se caractérise par des transports, qui sont directement liés à la production, dans le sens où, une fois les opérations de production déterminées, les opérations de transport sont connues, ou réciproquement. De ce point de vue, le problème suivant, abordé dans le chapitre 4, qui étend le RCPSP, appartient à une catégorie de problèmes très différents.

CHAPITRE 4

Le problème d'ordonnancement de projet sous contraintes de ressources avec routage

Ce chapitre traite du problème d'ordonnancement de projet sous contraintes de ressources avec routage (Resource-Constrained Project Scheduling Problem with Routing - RCPSPR). Ce problème intégré étend le problème d'ordonnancement de projet sous contraintes de ressources (RCPSP) en introduisant la gestion d'une flotte hétérogène de véhicules permettant le transport des ressources. Dans ce nouveau problème, les véhicules permettent le transport des ressources, d'une activité à une autre, avec un seul type de ressource. Des jeux de données sont introduits et proposés comme benchmark pour de futures études.

Nous proposons dans ce chapitre une méthode de résolution du RCPSP basée sur une représentation indirecte de la solution, et une fonction d'évaluation intégrée au sein d'une métaheuristique de type GRASP×ELS. La contribution majeure de ce chapitre repose sur la proposition d'une fonction d'évaluation composée de plusieurs étapes, permettant de résoudre les sous-problèmes du RCPSPR. Cette fonction comprend : 1) une heuristique permettant de définir un flot qui est une solution du RCPSP ; 2) une méthode pour diviser les ressources transportées afin de respecter la capacité des véhicules ; 3) une heuristique pour définir un tour géant ; 4) la définition d'un algorithme de type SPLIT (adapté au RCPSPR) qui résout le problème de détermination des tournées avec l'affectation des véhicules. Une solution pouvant être modélisée à l'aide d'un graphe disjonctif orienté, une recherche locale peut être définie en tirant profit du chemin critique.

Les expérimentations numériques montrent que cette méthode de résolution est efficace. Les solutions obtenues grâce au GRASP×ELS sont comparées aux solutions optimales obtenues par programmation linéaire.

4.1 Introduction et état de l'art du problème

Dans ce chapitre, le problème d'ordonnancement de projet sous contraintes de ressources avec routage (RCPSPR) est introduit. Ce problème est une extension du RCPSP.

4.1.1 Le RCPSP

Le RCPSP est caractérisé par un ensemble de ressources disponibles en quantités limitées et par des activités, dont les demandes en ressources et les durées sont connues, liées entre-elles par des relations de précédence. Ce problème consiste à trouver un ordonnancement de durée minimale, en définissant une date de début pour chaque activité, de telle sorte que les relations de précédence et les contraintes de ressources

soient respectées. Le RCPSP est un problème d'optimisation combinatoire défini par un sextuplet (V, p, E, R, B, b).

L'ensemble des activités du problème est noté $V = \{A_0, \ldots, A_{n+1}\}$, une durée $p_i \in \mathbb{N}$ est associée à chaque activité A_i, avec $n \in \mathbb{N}$ le nombre d'activités non fictives. Il existe deux activités fictives, A_0 et A_{n+1}, ces activités correspondent respectivement au début du projet (activité qui précède toutes les autres) et à la fin du projet (activité qui succède à toutes les autres). Pour ces activités, $p_0 = p_{n+1} = 0$. L'ensemble des activités non fictives est identifié par $A = \{A_1, \ldots, A_n\}$.

L'ensemble des couples d'activités liées par une relation de précédence est noté E. Un couple $(A_i, A_j) \in E$ signifie que l'activité A_i précède l'activité A_j. Cette relation est transitive, *i.e.* $\left((A_i, A_j) \in E \wedge (A_j, A_k) \in E\right) \Rightarrow (A_j, A_k) \in E$, et antisymétrique, *i.e.* $i \neq j \Rightarrow \neg\left((A_i, A_j) \in E \wedge (A_j, A_i) \notin E\right)$.

L'ensemble des q ressources renouvelables est noté $R = \{R_1, \ldots, R_q\}$. Les quantités disponibles des ressources sont représentées par un vecteur $B \in \mathbb{N}^q$, de telle sorte que B_k désigne la disponibilité de la ressource R_k. Chaque activité A_i nécessite une certaine quantité de chaque ressource R_k, ce qui correspond à une demande notée $b_{ik} \in \mathbb{N}$.

Une solution du RCPSP est caractérisée par un vecteur contenant les dates de début de chaque activité, $S = (S_0, \ldots, S_{n+1}) \in \mathbb{N}^{n+2}$. On peut noter C_i la date de fin de l'activité A_i de telle sorte que $C_i = S_i + p_i$. Une solution du RCSP doit respecter les contraintes de précédence (1) et les contraintes de ressources (2), avec $A_t = \{A_i \in A \mid S_i \leq t \leq C_i\}$ l'ensemble des activités non fictives en cours à la date t.

$$\forall (A_i, A_j) \in E,\ S_j - S_i \geq p_i \tag{1}$$

$$\forall R_k \in R, \forall t \geq 0,\ \sum_{A_i \in A_t} b_{ik} \leq B_k \tag{2}$$

Le makespan associé à une solution du RCPSP est égal à S_{n+1}, la date de début de la dernière activité. Une activité ne peut pas être interrompue, la préemption n'est pas autorisée.

Dans le Tableau 4-1, une instance du RCPSP est donnée avec $n = 6$ activités et $|R| = 1$ ressource avec $B_1 = 12$. Les contraintes de précédence entre les activités sont représentées sur la Figure 4-1 avec un graphe $G(V, E)$ dans lequel les nœuds correspondent aux activités et les arcs correspondent aux relations de précédence. Une solution optimale de ce problème, avec un makespan $S_{n+1} = 36$, est illustrée sur la Figure 4-2 à l'aide d'un diagramme de Gantt, avec en abscisse le temps et en ordonnée l'occupation des ressources. Les activités sont représentées par des rectangles, placés de telle sorte que l'abscisse du coin inférieur gauche soit égale à la date de début de l'activité. Une solution du RCPSP est entièrement définie par les dates de début des activités. Dans cet exemple, on a $S = (0,2,0,2,22,20,22,36)$. Cependant, la position des activités peut être modifiée par translation suivant l'axe des ordonnées. Une translation permet de modifier les quantités de ressources échangées entre deux activités. De plus, le rectangle représentant une activité peut être divisé en plusieurs rectangles de même abscisse mobiles suivant l'axe des ressources. Une solution n'est donc pas associée à une représentation unique.

Tableau 4-1
Données d'un projet composé de 6 activités et d'une ressource.

A_i	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7
p_i	0	20	2	18	6	2	14	0
b_{i1}	0	4	10	4	6	3	4	0

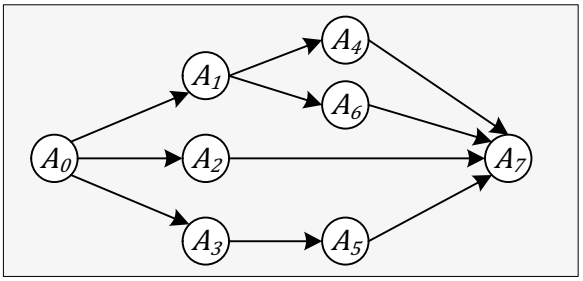


Figure 4-1. Graphe des précédences

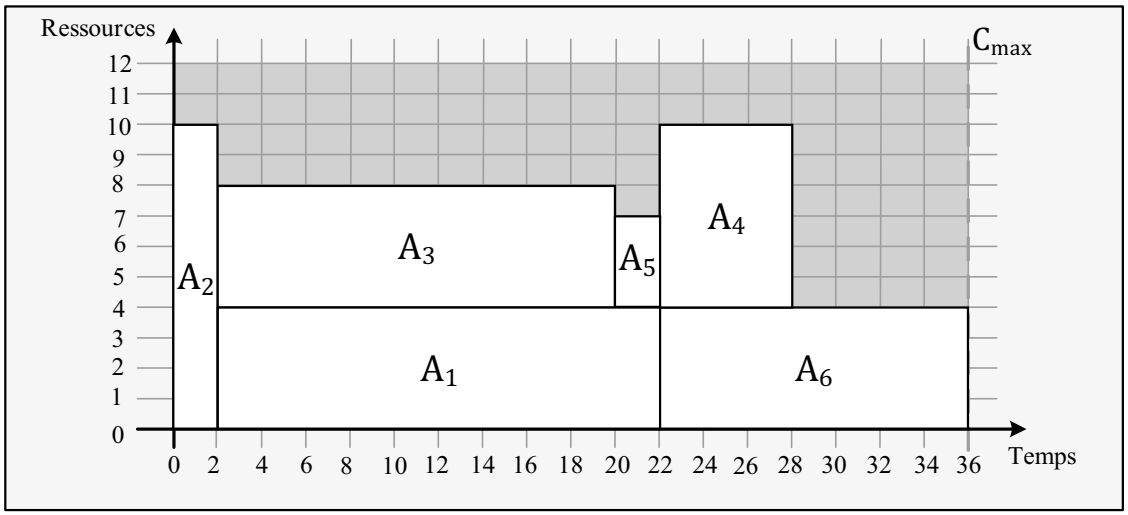


Figure 4-2. Solution optimale du RCPSP avec 6 activités et une ressource

Le RCPSP est un problème NP-difficile (voir Blazewicz *et al.,* (1983) et Brucker *et al.*, (1999) pour plus de détails sur la complexité). La théorie de la complexité définit qu'un problème est NP-difficile si sa version décisionnelle est NP-complet. De nombreuses études et classifications ont été publiées sur le RCPSP, on peut citer par exemple Herroelen *et al.*, (1998), Brucker *et al.*, (2000), Kolisch et Padman (2001), Weglarz (1999), Demeulemeester et Herroelen (2002) et Hartmann et Briskorn (2010).

4.1.2 Le RCPSP avec temps de transfert

De nombreuses extensions du RCPSP ont été étudiées. Elles incluent, par exemple, la préemption des activités (Damay et al. 2007), (Moukrim et Quilliot, 2005), des contraintes temporelles (time-lags) (De Reyck et Herroelen, 1998), des ressources non renouvelables (Kimms, 2001), ou encore des temps de transfert entre les activités. Cette dernière extension peut permettre de modéliser, par exemple, le temps de reconfiguration d'une usine ou encore le temps de transport des ressources entre deux activités. Peu de travaux ont été publiés sur ce problème, mais on peut citer (Krüger et Scholl, 2010) et (Poppenborg et Knust, 2016) ainsi que (Quilliot et Toussaint, 2013).

Dans le problème du RCPSP avec temps de transfert (RCPSPTT – RCPSP with transfer time), lorsqu'une unité de ressource R_k est transférée de l'activité A_i à l'activité A_j, un temps de transfert t_{ij}^k doit être pris en compte. Ce temps de transfert correspond à une durée minimale qui doit séparer la fin de l'activité A_i et le début de A_j.

En reprenant la solution du RCPSP « classique » donnée sur la Figure 4-2 et en ajoutant des temps de transfert, dont les durées sont indiquées sur le Tableau 4-2, on obtient la solution explicitée sur la Figure 4-3. La durée des temps de transfert est représentée par une double flèche horizontale. La solution du RCPSP avec temps de transfert a un makespan égal à 94, date à laquelle l'activité fictive A_7 peut débuter. L'ajout de temps de transfert (qui appartiennent à la famille des time-lags minimaux) a pour effet d'« éloigner » les activités les unes des autres.

Tableau 4-2
Durées associées aux temps de transfert entre deux activités.

t_{02}	t_{16}	t_{21}	t_{23}	t_{24}	t_{34}	t_{35}	t_{47}	t_{54}	t_{67}
8	17	2	5	3	3	21	5	19	9

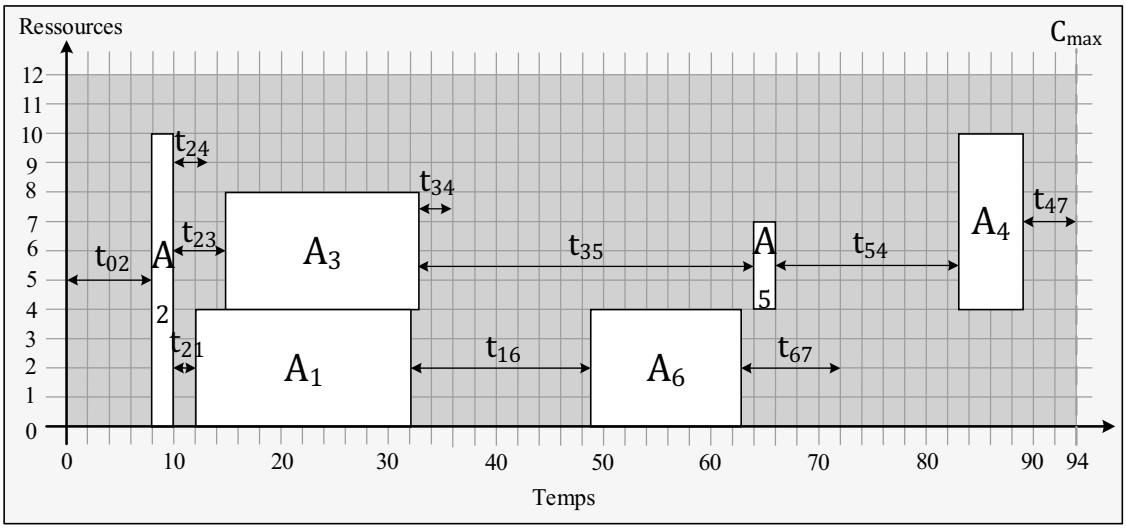


Figure 4-3. Solution du RCPSP avec temps de transfert

4.1.3 La modélisation du transport dans les problèmes d'ordonnancement

Plusieurs extensions de problèmes d'ordonnancement traitent de l'ajout de contraintes de transport, on peut citer le job shop flexible avec transport (Zhang et al, 2012b), le job shop avec transport (Knust, 1999), (Lacomme *et al.*, 2007), (Afsar *et al.*, 2016), les systèmes flexibles de production (FMS) (Caumond *et al.*, 2009), ou encore le RCPSP avec le transport (Quilliot et Toussaint, 2013). Les articles de (Krüger et Scholl, 2010, Poppenborg et Knust, 2016) ont abordé le problème du RCPSP avec des temps de transfert, cependant dans ces études le transport n'est pas explicitement modélisé avec une flotte de véhicules.

Les approches pour les problèmes d'ordonnancement sont majoritairement basées sur une modélisation de type d'un graphe disjonctif (Roy et Sussmann, 1964). Cet outil de modélisation a été étendu aux problèmes d'ordonnancement avec transport (pour le job shop par exemple, par (Lacomme *et al.*, 2007) et (Zhang *et al.*,2012a), où les opérations de transport sont modélisées par des nœuds avec des arcs disjonctifs entre les opérations nécessitant la même machine (ou le même véhicule).

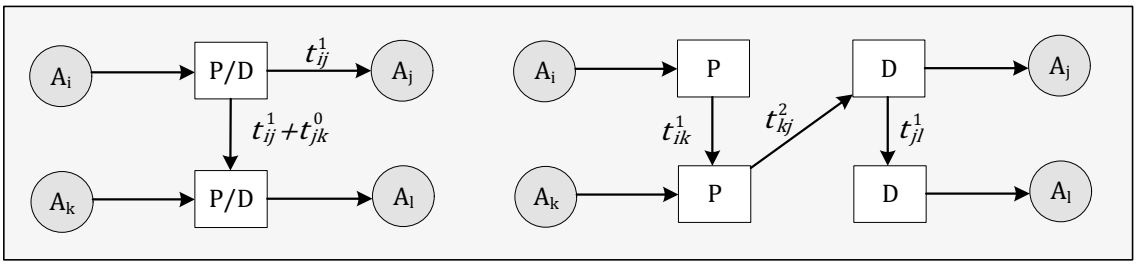
Les contraintes de transport, dans les problèmes d'ordonnancement, peuvent être modélisées implicitement de deux façons. La première consiste à modéliser un

déplacement physique des ressources d'un endroit à l'autre (modélisant le transport de ressources par exemple). La seconde repose sur une modélisation grâce à des time-lags entre les machines (variante des temps de transfert).

Les time-lags minimaux entre les activités sont couramment utilisés pour modéliser un délai minimal et peuvent dépendre ou non des activités. Si la gestion explicite des véhicules n'est pas nécessaire, cette approche offre une modélisation efficace du transport et peut également être pertinente dans les problèmes où le parc automobile est suffisamment grand, pour supposer qu'un transport peut être réalisé sans délai après la fin d'une activité. Le problème étudié par (Poppenborg et Knust, 2016), sur le RCPSP avec les temps de transfert, appartient à cette approche de modélisation. Les time-lags maximaux entre les activités sont utilisés dans les problèmes de collecte et de livraison, lorsque le temps de conduite est limité, ou dans les problèmes avec des contraintes de périssabilité.

La modélisation explicite du transport prend en compte la capacité du véhicule. Dans certaines situations, la capacité du véhicule peut être supposée unitaire (le véhicule ne peut pas réaliser plus d'une requête à la fois). Avec cette hypothèse, les ressources transportées sont acheminées sans détour d'une activité à l'autre, et la livraison (D-Delivery) est donc précédée directement de la collecte (P - Pickup). Dans ce cas, une solution du problème de tournées de véhicules consiste à définir les disjonctions entre les opérations de transport (couple P/D), sur la Figure 4-4 a.

Selon le problème, le temps de transport peut dépendre de la charge du véhicule et peut être désigné par t_{ij}^{x}, définissant le temps de transport requis pour que le véhicule passe de l'activité A_i à A_j avec x ressources chargées (on note parfois $e_{ij} = t_{ij}^{0}$). Si les temps de transport dépendent du véhicule, ils peuvent être notés t_{ij}^{ux} avec T_u le véhicule.



a) Véhicule de capacité unitaire *b) Véhicule de capacité non-unitaire*

Figure 4-4. Représentations explicites du transport avec collecte et livraison

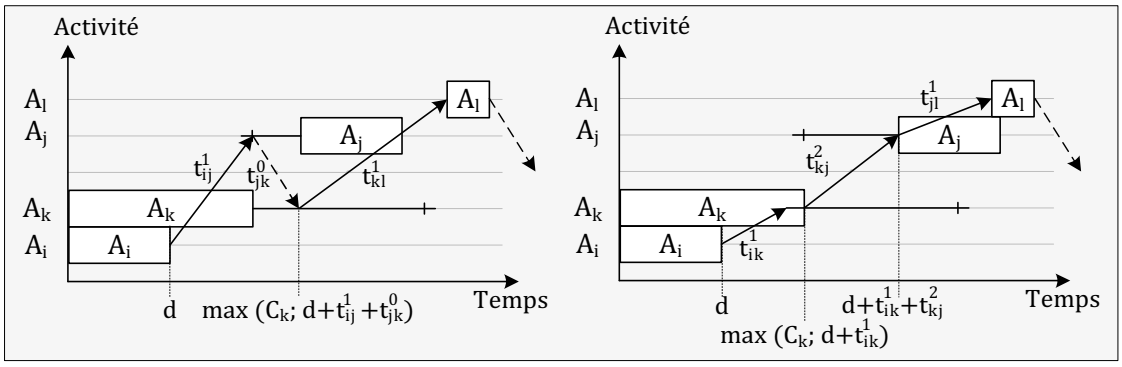
Comme souligné par (Lacomme *et al.*, 2013), l'arc disjonctif entre deux opérations de transport est modélisé par un arc de coût $t_{ij}^{1} + t_{jk}^{0}$ (Figure 4-4 a), correspondant à une opération de transport à charge entre l'activité A_i à l'activité A_j et d'un déplacement à vide de l'activité A_j à l'activité A_k.

Lorsque les véhicules sont de capacités non-unitaires, les disjonctions doivent être définies entre deux opérations de collecte ou de livraison, comme sur l'exemple représenté par la Figure 4-4 b). Le modèle doit donc différencier les nœuds de collecte des nœuds de livraison. Une solution du problème de tournée consiste à définir les disjonctions entre les nœuds de collecte et les nœuds de livraison. Par exemple, la Figure 4-4 b) représente une solution dans laquelle le véhicule collecte une ressource sur

l'activité A_i, puis une seconde ressource sur l'activité A_k, décharge la première ressource sur l'activité A_j et le deuxième sur l'activité A_l.

Ces opérations de transport sont représentées temporellement sur la Figure 4-5. Sur la Figure 4-5 a), la date de début de l'opération de transport dépend à la fois de la date C_k de fin de l'activité A_k, et de la date de disponibilité du véhicule sur l'activité A_k. Le véhicule est disponible sur l'activité A_k lorsque l'opération de transport précédente est achevée et que le véhicule s'est positionné sur l'activité A_k (un déplacement à vide peut être nécessaire). Le véhicule est donc disponible à la date $d + t_{ij}^1 + t_{jk}^0$, d étant la date de début de l'opération de transport précédente. Le chargement d'un véhicule au cours du temps est donc caractérisé par une alternance de valeurs positives (véhicule chargé) et nulles (véhicule vide).

Sur la Figure 4-5 b), correspondant au cas non-unitaire, le véhicule peut effectuer séquentiellement plusieurs opérations de transport à charge. La ressource collectée sur l'activité A_i est transportée vers l'activité A_k avant d'être livrée à l'activité A_j. Le chargement d'un véhicule au cours du temps est donc caractérisé par des suites de valeurs croissantes (opérations de collecte) et des suites de valeurs décroissantes (opérations de livraison).



a) Véhicule de capacité unitaire *b) Véhicule de capacité non-unitaire*

Figure 4-5. Représentations temporelle des opérations de transport

De tels modèles ont été utilisés pour modéliser des problèmes intégrant ordonnancement et transport dans un graphe disjonctif. Une solution est alors entièrement définie par l'arbitrage des disjonctions sur les opérations de production/activités et des disjonctions sur les opérations de transport.

4.2 Définition du problème

Le problème étudié dans ce chapitre est une extension du RCPSP et du RCPSP avec temps de transfert, énoncés dans la partie précédente. En effet, le problème étudié est caractérisé par la disponibilité d'une flotte hétérogène (et limitée) de véhicules et un seul type de ressources. Les véhicules sont considérés de capacité unitaire afin de transporter les ressources directement. Ce problème est nommé RCPSPR (Resource-Constrained Project Scheduling Problem with Routing).

4.2.1 Définition générale

Le RCPSPR est défini grâce aux notations du RCPSP présentées dans la partie précédente, ainsi que des notations liées à la flotte hétérogène de véhicules. Le RCPSPR est un problème d'optimisation combinatoire défini par un nonuplet $(V, p, E, R, B, b, T, C, t)$ qui étend le sextuplet défini précédemment pour le RCPSP.

L'ensemble des véhicules du problème est noté $T = \{T_1, \dots, T_N\}$, une capacité $C_u \in \mathbb{N}$ étant associée à chaque véhicule T_u avec $N \in \mathbb{N}$ le nombre de véhicules disponibles. Les véhicules sont donnés par ordre décroissant de capacités, *i.e.* $C_u \geq C_{u+1}, \forall u \in [\![1, N-1]\!]$. La durée de transport entre deux activités est donnée par la matrice $t \in \mathbb{N}^{(n+2)\times(n+2)}$, la durée de transport entre l'activité A_i et l'activité A_j, est notée $t_{ij} \in \mathbb{N}$. Cette durée ne dépend ni du véhicule, ni du chargement (quantité ou type de ressource) et correspond au plus court chemin entre les deux activités. L'inégalité triangulaire est respectée.

Le problème traité dans ce chapitre est caractérisé par un seul type de ressource. L'indice sur le type de ressource est donc supprimé pour alléger les notations, par conséquent, $b_{i1} = b_i$.

Comme pour le RCPSP, une solution du RCPSPR est caractérisée par un vecteur $S = (S_0, \dots, S_{n+1}) \in \mathbb{N}^{n+2}$ contenant les dates de début de chaque activité, et par un ensemble de $N_t \leq N$ tournées. Chaque tournée est effectuée par un véhicule et est composée d'une ou plusieurs opérations de transport. Une opération de transport $O_{i,j,u,x} = (P_{i,j,u,x}, D_{i,j,u,x})$ est définie par :

- deux activités A_i et A_j entre lesquelles une quantité $0 < x \leq C_u$ de ressource est transportée par le véhicule T_u ;
- une opération de collecte (pickup) $P_{i,j,u,x}$ à laquelle est associée une date de départ $B_{i,j,u,x}$ (départ du véhicule T_u sur l'activité A_i avec x ressource) ;
- une opération de livraison (delivery) $D_{i,j,u,x}$ à laquelle est associée une date d'arrivée $A_{i,j,u,x}$ (arrivée du véhicule T_u sur l'activité A_j avec x ressource).

Pour définir une tournée, seules les opérations de transport (à charge) sont nécessaires puisque les déplacements du véhicule à vide ont lieu entre deux opérations de transports (les déplacements du véhicule sont ordonnancés au plus tôt).

Une solution du RCPSPR doit respecter les contraintes de précédence, les contraintes de ressources précédemment mentionnées, ainsi que les contraintes de capacités des véhicules. Le makespan associé à une solution du RCPSPR est égal à S_{n+1}, la date de début de la dernière activité (activité fictive). Une activité ne peut pas être interrompue, la préemption sur les activités n'étant pas autorisée. Chaque activité A_i nécessite une certaine quantité de la ressource R_1. La demande en ressource b_i d'une activité A_i peut être supérieure à la capacité du plus grand véhicule, et la préemption sur le transport est par conséquent autorisée, puisque les demandes peuvent être fractionnées et traitées par des véhicules différents.

Dans ce chapitre, une hypothèse concernant le transport a été formulée : les ressources sont acheminées directement d'une activité à l'autre. Par conséquent, la tournée d'un véhicule est composée d'une alternance d'opérations de collecte et de livraison. Deux

opérations de collecte (ou de livraison) ne peuvent pas être effectuées consécutivement dans une tournée.

4.2.2 Illustration des problématiques autour du RCPSPR avec un exemple

Dans ce paragraphe, le problème du RCPSPR est présenté en reprenant le problème avec six activités énoncé dans la partie précédente (voir Tableau 4-1 et Figure 4-1) pour illustrer les points-clés de la résolution intégrée. Pour résoudre ce problème, trois sous-problèmes doivent être résolus conjointement :

- le problème du RCPSP, pour définir les échanges de ressources entre les activés ;
- le problème d'affectation des véhicules, pour définir les opérations de transport permettant la réalisation des échanges de ressources entre les activités ;
- l'ordonnancement des opérations de transport, pour définir les tournées de véhicules et les dates de début des activités.

Les deux derniers sous-problèmes à résoudre peuvent être regroupés en un même problème, présentant des similitudes avec le VRPPD (Vehicle Routing Problem with Pickup and Delivery). Une même activité reçoit et envoie des ressources acheminées par une flotte hétérogène de véhicules, ces ressources sont initialement disponibles sur une activité fictive (dépôt) et doivent être restituées à cette même activité à la fin de l'ordonnancement. Cependant, une activité peut être visitée plusieurs fois pour acheminer les ressources. De plus, entre la dernière opération de livraison et la première opération de collecte sur une activité, une durée minimale doit être respectée (durée de réalisation de l'activité).

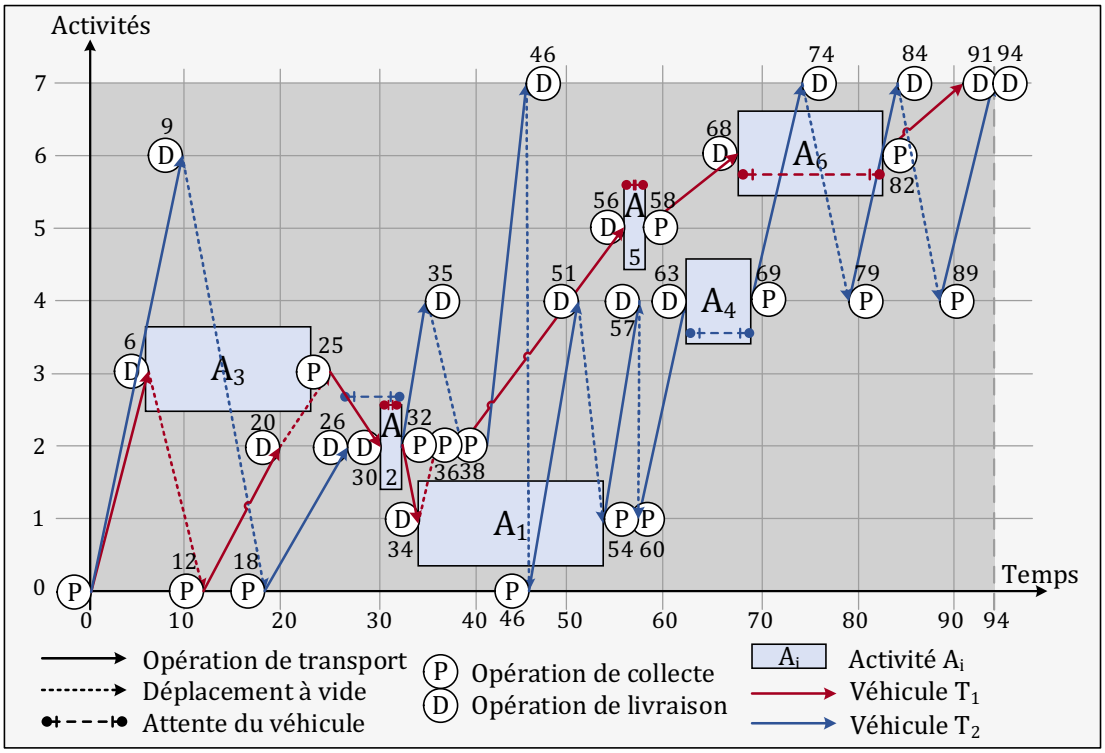


Figure 4-6. Solution optimale du RCPSPR avec deux véhicules

La Figure 4-6 et la Figure 4-7 présentent une solution optimale du problème à deux véhicules, tels que $C_1 = 4$ et $C_2 = 2$. Cette solution est composée de deux tournées :

- la tournée du véhicule T_1 (représenté par des arcs rouges sur la Figure 4-6 et la Figure 4-7), composée de sept opérations de transport (10 déplacements du véhicule au total) ;
- la tournée du véhicule T_2 (représenté par des arcs bleus sur la Figure 4-6 et la Figure 4-7), composée de 10 opérations de transport (16 déplacements du véhicule au total).

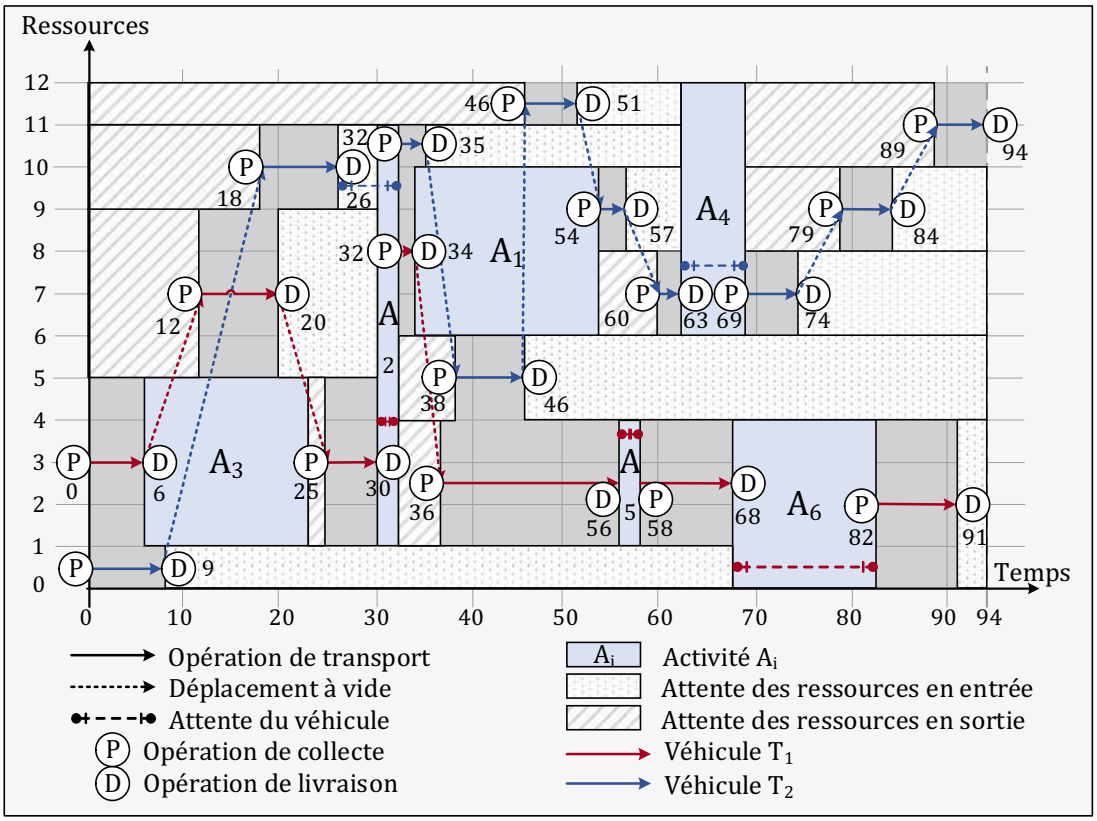


Figure 4-7. Solution optimale du RCPSPR avec deux véhicules avec une représentation explicite de la gestion des ressources

La représentation de la solution illustrée sur la Figure 4-7 n'est pas unique, il existe plusieurs manières de positionner les rectangles liés aux activités par translation verticale. Néanmoins, contrairement au RCPSP, une solution du RCPSPR est caractérisée par les échanges de ressources entre activités, la représentation sous forme de diagramme de Gantt doit tenir compte des échanges de ressources entre activités.

La solution peut également être représentée grâce à un réseau routier, en prenant en compte la localisation des activités et les déplacements des véhicules, comme sur la Figure 4-8. Les opérations de collecte (respectivement de livraison) associées aux opérations de transport sont représentées par la lettre P (respectivement D). Les ressources non transportées et non utilisées pour la réalisation d'une activité sont en attentes, soit avant le début d'une activité, soit après la fin d'une activité. Ces temps d'attentes sont représentés par des rectangles, sur la Figure 4-7, dont la hauteur représente la quantité de ressource en attente et la longueur la durée d'attente.

On peut détailler le début de la tournée du véhicule T_1 représentée sur la Figure 4-6 et la Figure 4-7. Cette tournée débute par l'opération de transport $O_{0,3,1,4}$ avec une opération

de collecte sur l'activité A_0 à la date 0 et une opération de livraison sur l'activité A_1 à la date 6 (puisque $t_{03} = 6$). À l'issue de cette opération de transport l'activité A_3 débute puisqu'elle dispose de l'ensemble des ressources demandées ($b_3 = 4$). Le véhicule effectue ensuite un déplacement à vide vers l'activité A_0 pour effectuer l'opération de transport $O_{0,2,1,4}$ entre les dates 12 et 20. L'activité A_3 se termine à la date 24, à partir de cette date les ressources peuvent donc être récupérées. Le véhicule effectue donc un déplacement à vide vers l'activité A_3 pour effectuer l'opération de transport $O_{3,2,1,4}$ entre les dates 25 et 30. Avant d'effectuer l'opération de transport $O_{2,1,1,4}$, le véhicule T_1 doit attendre pendant la durée de l'activité A_2.

Cet exemple permet d'illustrer deux points importants dans le RCPSPR, démontrant la forte interaction entre le problème d'ordonnancement et de tournées de véhicules :

- une activité ne peut débuter que lorsque toutes les ressources nécessaires à sa réalisation ont été livrées. Cet exemple peut être illustré avec l'activité A_2 sur la Figure 4-7. Trois opérations de transport permettent de livrer les ressources de A_2, $O_{0,2,1,4}$, $O_{0,2,2,2}$ et $O_{3,2,1,4}$ avec les dates respectives, 20, 26 et 30. Par conséquent, la date de début au plus tôt de l'activité A_2 est égale à $S_2 = 30$.
- une opération de collecte sur une activité ne peut avoir lieu que lorsque l'activité est achevée (ressources libérées) et que le véhicule est disponible sur l'activité. Ces deux situations sont illustrées sur la Figure 4-7. Pour l'opération de transport $O_{2,1,1,4}$, celle-ci peut débuter dès la fin de l'activité A_2 puisque le véhicule T_1 est sur l'activité A_2, et donc $B_{2,1,1,4} = 32$. Pour l'opération de transport $O_{2,5,1,3}$, celle-ci ne peut pas débuter dès la fin de l'activité A_2 puisque le véhicule T_1 n'est pas disponible. Le véhicule doit effectuer l'opération de transport $O_{2,1,1,4}$ puis revenir sur l'activité A_2, et donc $B_{2,5,1,3} = A_{2,1,1,4} + t_{1,2} = 34 + 2 = 36$.

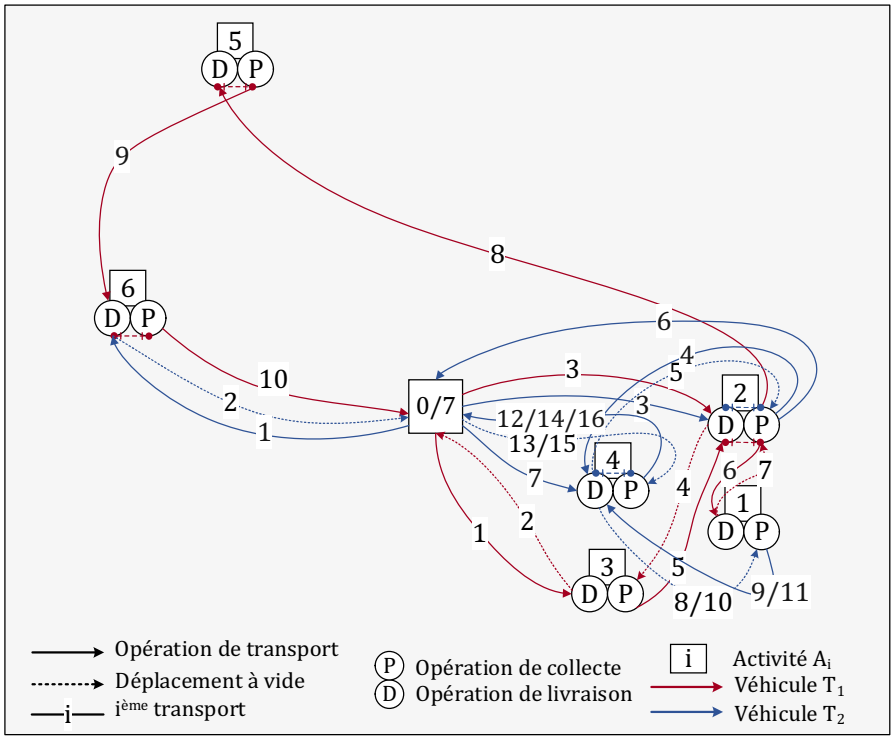


Figure 4-8. Tournées de la solution optimale du RCPSPR avec deux véhicules

4.2.3 Les outils de modélisation d'une solution du RCPSPR

Avec l'hypothèse du transport direct des ressources entre les activités, il est possible d'utiliser un modèle de flot (Artigues *et al.*, 2003) pour déterminer les opérations de transport définies entre deux activités. Une solution du RCPSP peut être caractérisée par un flot (Ahuja *et al.*, 1993) car elle implique la circulation de ressources, par exemple des biens, des personnes ou de l'énergie. Pour les mêmes raisons, une solution du RCPSP peut également être caractérisée par un flot. La Figure 4-9 donne une représentation du graphe flot $G_\varphi(V, E)$ associé à la solution optimale de la Figure 4-7. Ce graphe est défini par un ensemble de nœuds correspondant aux activités du RCPSP et un ensemble d'arcs modélisant les échanges de ressources entre deux activités, le coût associé à ces arcs représentant la quantité de ressources échangées.

On note φ_{ij} la quantité de flot transférée de l'activité A_i à l'activité A_j. Sur la Figure 4-9, une unité de ressource est transférée de l'activité A_0 à l'activité A_4, on a alors $\varphi_{04} = 1$. Dans cet exemple, on peut vérifier que la conservation du flot est respectée.

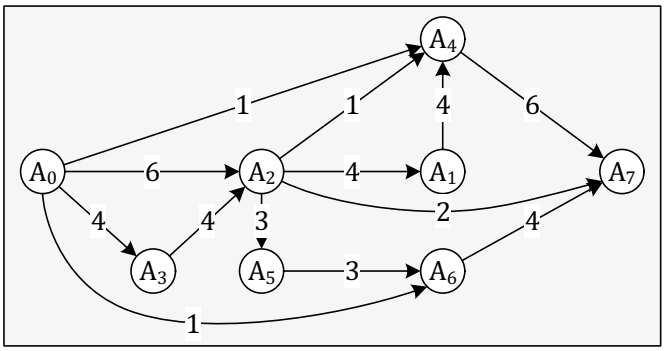


Figure 4-9. Flot associé à la solution optimale du RCPSPR avec deux véhicules

Une proposition de modélisation du RCPSP avec routage consiste en l'extension du graphe disjonctif proposé dans (Roy et Sussmann, 1964), (Knust, 1999), (Hurink et Knust, 2002) et (Hurink et Knust, 2005). Cette extension est proposée dans (Lacomme *et al.*, 2007) et (Lacomme *et al.*, 2010). Le graphe disjonctif modélisant le job shop « classique » présenté dans le chapitre 1 est étendu dans (Knust, 1999) par l'ajout d'opérations de transport, représentées par des carrés, entre les opérations machines d'un même job.

Dans le RCPSPR, la notion de job étant restreinte à une machine/activité, la localisation des opérations de transport dans le graphe disjonctif n'est pas définie par l'énoncé du problème. La définition d'un flot est une solution pour définir des disjonctions entre les activités et ainsi constituer des jobs avec des opérations de transport entre deux activités d'un même job. Contrairement à une opération de transport, une opération de transfert ne prend pas en compte la capacité des véhicules. Ces opérations peuvent être définies directement à partir d'un flot. On utilise un carré vide pour représenter une opération de transfert entre deux activités (Figure 4-10). Ce carré modélise une opération de transport lorsqu'il contient un numéro, qui correspond au véhicule qui effectue l'opération de transport.

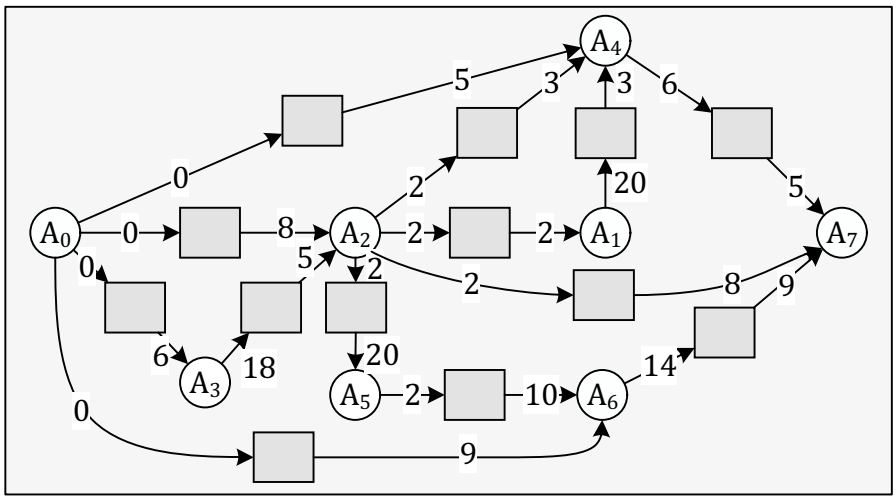


Figure 4-10. Définition des opérations de transfert à partir d'un flot conduisant à la solution optimale du RCPSPR

Pour chaque opération de transfert entre deux activités, une ou plusieurs opérations de transport peuvent être définies. Un nouveau graphe $G_\varphi(V_A \cup V_T, E_A \cup E_T)$ peut alors être introduit pour représenter une solution du RCPSPR avec :

- V_A l'ensemble des nœuds représentant les activités fictives et non fictives ;
- V_T l'ensemble des nœuds représentant les opérations de transport ;
- E_A l'ensemble des arcs représentant les disjonctions entre les activités ;
- E_T l'ensemble des arcs représentant les disjonctions entre les opérations de transport.

Á partir de la solution présentée en Figure 4-7 une modélisation sous forme de graphe disjonctif orienté et évalué est donné en Figure 4-11. Sur cette figure apparaissent trois types d'arcs disjonctifs appartenant à deux catégories et un seul type d'arcs conjonctifs correspondant aux contraintes de précédences du problème. Ces arcs sont explicités dans le Tableau 4-3, avec les coûts associés.

Afin de faciliter la compréhension de la Figure 4-11 et du lien entre opération de transfert et opération de transport, des sous-graphes avec les activités A_0, A_2 et A_3 sont représentés sur la Figure 4-12. On peut remarquer que l'opération de transfert entre les activités A_0 et A_2 donne lieu à deux opérations de transport affectées à deux véhicules différents. Les véhicules ayant des capacités ($C_1 = 4$ et $C_2 = 2$) inférieures à la quantité totale de ressources transférée $\varphi_{0,2} = 6$, au minimum deux opérations de transport sont effectuées.

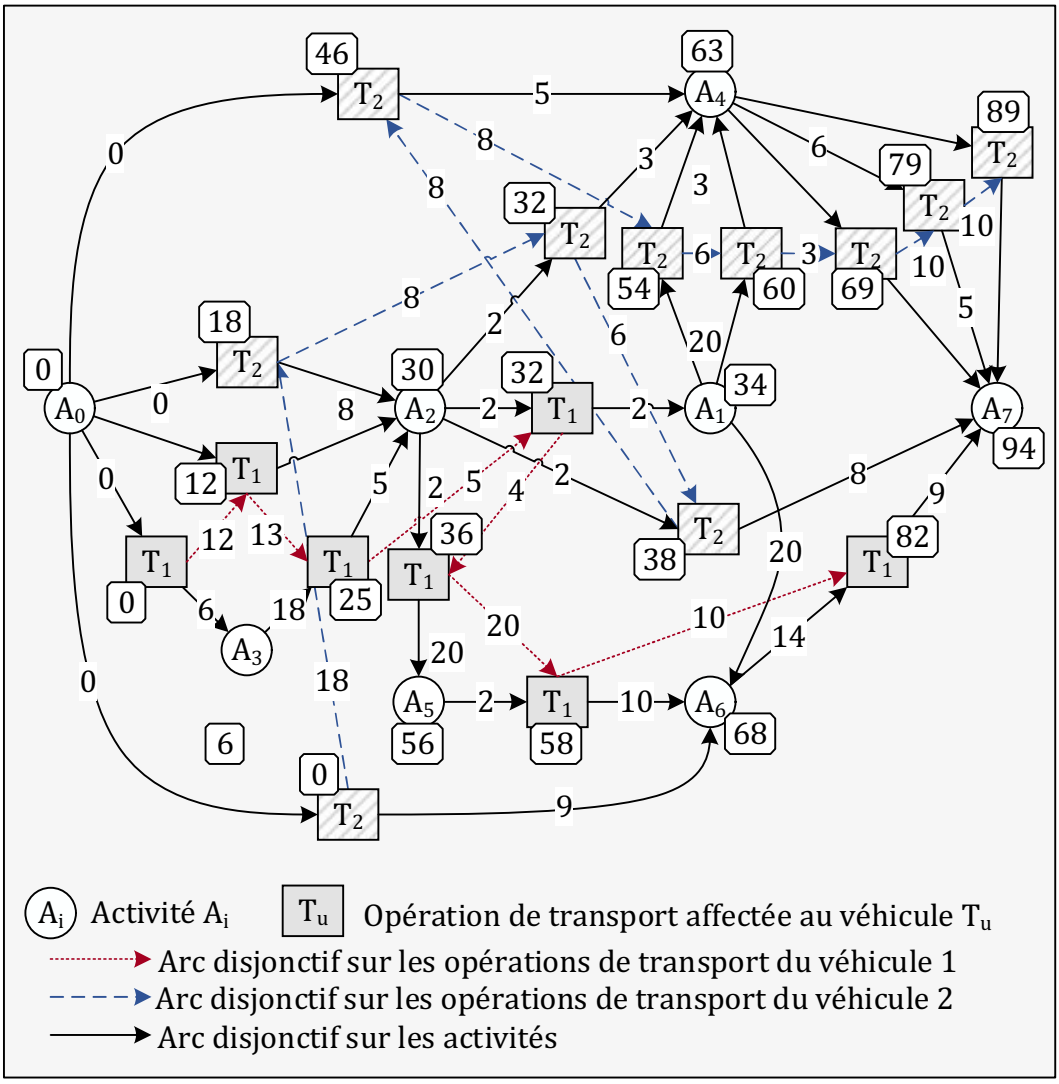


Figure 4-11. Représentation de la solution optimale du RCPSPR avec deux véhicules sous la forme d'un graphe disjonctif orienté et évalué

On peut également noter que la totalité des ressources demandées par l'activité A_2 sont transportées grâce à trois opérations de transport dont deux effectuées par le véhicule T_1. Sur l'exemple de la Figure 4-12, le début de la tournée du véhicule T_1 est représenté. Cette tournée débute par l'opération de transport entre l'activité A_0 et l'activité A_3 à la date 0. Ensuite, une opération de transport doit avoir lieu entre l'activité A_0 et l'activité A_2, le véhicule doit donc revenir à vide sur l'activité A_0 pour collecter les ressources à la date $t = t_{03} + t_{30} = 12$. Le véhicule peut donc livrer les ressources à l'activité A_2 à la date $t = 12 + t_{02} = 20$. La troisième opération de transport à effectuer doit avoir lieu entre l'activité A_3 et l'activité A_2, le véhicule revient donc sur l'activité A_3 à la date $t = 20 + t_{23} = 25$. Les ressources peuvent être collectées puisque l'activité A_3 est achevée ($C_3 = S_3 + p_3 = 6 + 18 = 24$), elles sont alors transportées jusqu'à l'activité A_2 et livrées à la date $t = 25 + t_{32} = 30$.

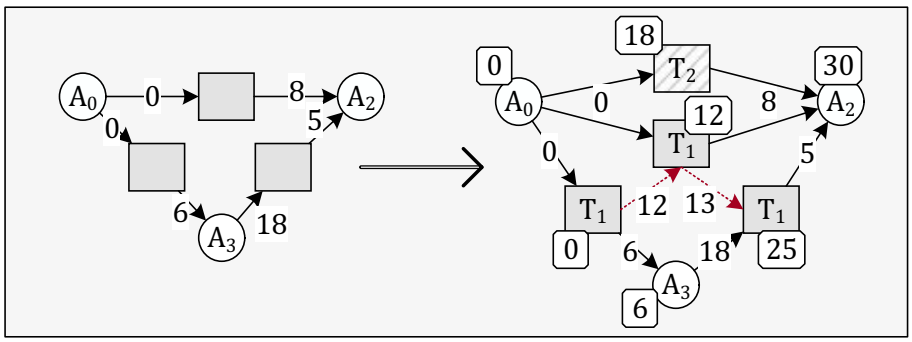


Figure 4-12. Lien entre le graphe flot avec les opérations de transfert et le graphe disjonctif orienté et évalué avec les opérations de transport

Tableau 4-3
Les catégories et types d'arcs dans le graphe disjonctif.

Catégorie	Type	Coût
Arc conjonctif sur les activités	$A_i \rightarrow A_j$	p_i
Arc disjonctif sur les activités	$A_i \rightarrow T_u \rightarrow A_j$	p_i
	$A_i \rightarrow T_u \rightarrow A_j$	t_{ij}
Arc disjonctif sur les opérations de transport	$A_i \rightarrow T_u \rightarrow A_j$ $A_i \rightarrow T_u \rightarrow A_j$	$t_{ij} + t_{jk}$

4.3 Formalisation linéaire : proposition

La formalisation linéaire présentée ici repose sur deux étapes permettant de définir complètement le problème du RCPSPR.

La première étape réside dans la définition des opérations de transport grâce à la résolution du RCPSP en s'appuyant sur la notion de flot. De nombreuses formulations ont été introduites pour résoudre le RCPSP. Inspirés par la formulation de (Balas *et al.*, 1970) sur le problème du job shop et par la formulation d'Alvarez-Valdés et de Tamarit (1993) avec la notion d'ensemble interdits sur le RCPSP, (Artigues *et al.*, 2003) ont proposé un modèle basé sur le flot pour le RCPSP. La résolution RCPSP basée sur la notion de flot a été étudiée par Artigues et Roubellat (2000) et Artigues *et al.* (2003).

Les données du problème du RCPSP dans la formulation reprennent les notations présentées précédemment. On introduit plusieurs variables pour formuler le programme linéaire lié au RCPSP avec la notion de flot :

- φ_{ijk} quantité de ressource R_k transférée entre l'activité A_i et l'activité A_j
- z_{ij} vaut 1 si l'activité A_i est ordonnancée avant l'activité A_j

Le programme linéaire $P1$, donné sur la Figure 4-13 fait apparaître les contraintes suivantes :

(1) objectif ;
(2) cette contrainte exprime la relation de précédence entre deux activités ;
(3) dans le cas où $z_{ij} = 1$, cette contrainte se réécrit $S_j \geq S_i + p_i$, ce qui signifie que l'activité A_j doit débuter après la fin de l'activité A_i. Dans le cas où $z_{ij} = 0$, cette contrainte se réécrit $S_j \geq S_i + p_i + M$, la valeur de M est suffisamment grande pour que cette inégalité soit toujours vérifiée, et ce quel que soit les valeurs des variables S_i et S_j ;
(4) dans le cas où $z_{ij} = 1$, cette contrainte se réécrit $\varphi_{ij} \leq min(b_i, b_j)$, ce qui signifie que le flot entre l'activité A_i et l'activité A_j ne peut pas dépasser les demandes des activités. Dans le cas où $z_{ij} = 0$, cette contrainte se réécrit $\varphi_{ij} \leq 0$, ce qui signifie que le flot entre l'activité A_i et l'activité A_j est nul ;
(5) cette contrainte assure que chaque activité A_j reçoit un flot de ressources égal à sa demande en ressources ;
(6) cette contrainte assure que chaque activité A_i fournit un flot de ressources égal à sa demande en ressources ;

Définition des variables :
(7) la variable z_{ij} est binaire ;
(8) les variables S_i sont réelles ;
(9) les variables φ_{ij} sont entières.

$$
P1: \begin{cases}
 & Minimiser\ S_{n+1} & (1)\\
\forall\ (A_i\ , A_j) \in E & z_{ij} = 1 & (2)\\
\forall\ (A_i\ , A_j) \in V^2 & S_j \geq S_i + p_i + M(z_{ij} - 1) & (3)\\
\forall\ (A_i\ , A_j) \in V^2 & \varphi_{ij} \leq min(b_i, b_j) z_{ij} & (4)\\
\forall\ A_j \in V & \sum_{i \in V} \varphi_{ij} = b_j & (5)\\
\forall\ A_i \in V & \sum_{j \in V} \varphi_{ij} = b_i & (6)\\
\forall\ (A_i\ , A_j) \in V^2 & z_{ij} \in \{0,1\}, z_{ii} = 0 & (7)\\
\forall\ A_i \in V & S_i \in \mathbb{R} & (8)\\
\forall\ (A_i\ , A_j) \in V^2 & \varphi_{ij} \in \mathbb{N}, \varphi_{ii} = 0 & (9)
\end{cases}
$$

Figure 4-13. Programme linéaire pour le RCPSP

Dans la suite de ce chapitre, on suppose que $b_0 = b_{n+1} = B$. Cette hypothèse permet de traiter les activités fictives A_0 et A_{n+1} comme les autres activités du point de vue du flot et des contraintes de conservation du flot.

La deuxième étape s'appuie sur la formulation précédente en ajoutant toutes les contraintes permettant la gestion de la flotte de véhicules. Elle permet de définir à partir d'un flot une ou plusieurs opérations de transport, d'affecter les véhicules aux opérations de transport et de déterminer les tournées des véhicules.

Les données du problème dans la formulation reprennent les notations présentées précédemment. On introduit plusieurs variables binaires :

- y_{ijut} quantité de ressource transportée de l'activité A_i à l'activité A_j par le véhicule T_u pour la $t^{ième}$ opération de transport entre les deux activités ;
- A_{ijut} date d'arrivée sur l'activité A_j du véhicule T_u pour la $t^{ième}$ opération de transport entre les deux activités ;
- B_{ijut} date de départ de l'activité A_i du véhicule T_u pour la $t^{ième}$ opération de transport entre les deux activités ;
- x_{ijut} qui vaut 1 si des ressources sont transportées par le véhicule T_u pour la $t^{ième}$ opération de transport entre l'activité A_i et l'activité A_j ;
- a^u_{ijtpqv} qui vaut 1 si la $t^{ième}$ opération de transport entre l'activité A_i et l'activité A_j est ordonnancée avant la $v^{ième}$ opération de transport entre l'activité A_p et l'activité A_q. Ces deux opérations de transport doivent toute deux être effectuées par le même véhicule T_u.

Pour plus de lisibilité, les indices, i, j, p, q sont utilisés pour les activités, u est utilisé pour les véhicules et t, v sont utilisés pour les numéros des opérations de transport. Le transport de toutes les ressources d'une activité A_i vers une activité A_j peut-être effectué dans le pire des cas (avec un véhicule de capacité 1), en $t = min(b_i, b_j)$ opérations de transport. Selon cette hypothèse, pour chaque couple (A_i , A_j) d'activités, un ensemble X_{il} de valeurs dans l'intervalle$[\![1, min(b_i, b_j)]\!]$ peut être créé pour numéroter les opérations de transport. Enfin, on considère une constante suffisamment grande que l'on note H.

Le programme linéaire $P2$, donné sur la Figure 4-17 fait apparaître les contraintes suivantes :

(10) ces contraintes assurent que toutes les ressources échangées entre les activités (A_i , A_j) dues au flot φ_{ij}, sont transportées par un ou plusieurs véhicules en une ou plusieurs opérations de transport ;

(11) dans le cas où $x_{ijut} = 1$, cette contrainte se réécrit $y_{ijut} \leq min(b_i, b_j, C_u)$, ce qui signifie que les opérations de transport entre l'activité A_i et l'activité A_j effectuées par le véhicule T_u ne peuvent ni dépasser les demandes des activités, ni la capacité du véhicule. Dans le cas où $x_{ijut} = 0$, cette contrainte se réécrit $x_{ijut} \leq 0$, ce qui signifie que la quantité de ressource transportée entre l'activité A_i et l'activité A_j par le véhicule T_u pour la $t^{ième}$ opération de transport est nulle ;

(12) dans le cas où $y_{ijut} = 0$ alors $x_{ijut} = 0$, ce qui signifie que si la quantité de ressource transporté entre l'activité A_i et l'activité A_j par le véhicule T_u pour la $t^{ième}$ opération de transport est nulle alors l'opération de transport associée n'existe pas ;

(13) dans le cas où $x_{ijut} = 1$, cette contrainte se réécrit $A_{ijut} \geq B_{ijut} + t_{ij}$, ce qui signifie que la durée entre le départ de l'activité A_i et l'arrivée sur l'activité A_j doit respecter la durée du transport effectué par le véhicule T_u. Dans le cas où $x_{ijut} = 0$, la valeur de H implique que l'inégalité est trivialement vérifiée ;

(14) dans le cas où $x_{ijut} = 1$, cette contrainte se réécrit $S_j \geq A_{ijut}$, ce qui signifie que l'activité A_j ne peut débuter que lorsque toutes les opérations de transport à destination de l'activité A_j sont arrivées. Dans le cas où $x_{ijut} = 0$, la valeur de H implique que l'inégalité est trivialement vérifiée. Un exemple est illustré sur la Figure 4-13 avec deux opérations de transport à destination de l'activité A_j pour livrer des ressources. La date de début de l'activité S_j, correspond à la livraison de la deuxième opération de transport à la date $A_{ijut} > A_{kjwv}$;

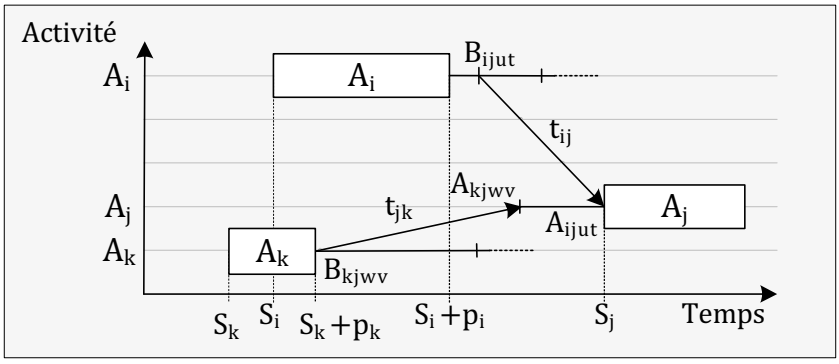
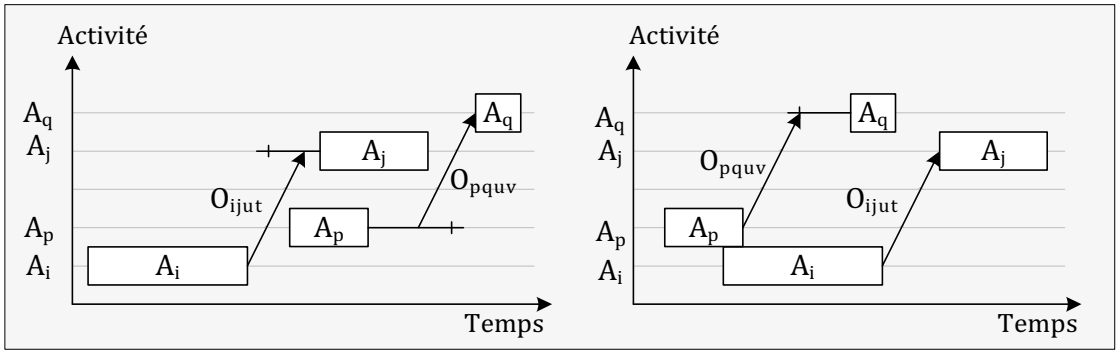


Figure 4-14. Illustration des contraintes (14) et (15)

(15) dans le cas où $x_{ijut} = 1$, cette contrainte se réécrit $B_{ijut} \geq S_i + p_i$, ce qui signifie que les opérations de transport au départ de l'activité A_i ne peuvent débuter que lorsque l'activité A_i est achevée. Dans le cas où $x_{ijut} = 0$, la valeur de H implique que l'inégalité est trivialement vérifiée. Un exemple est illustré sur la Figure 4-13 avec deux opérations de transport. Ces deux opérations de transport ne peuvent effectuer les opérations de collecte sur les activités A_i et A_k que lorsque celles-ci sont achevées, *i.e.*, $B_{ijut} \geq S_i + p_i$ et $B_{kjwv} \geq S_k + p_k$;

(16), (17) et (18) ces contraintes expriment les disjonctions entre les opérations de transport effectuées par le même véhicule. Dans le cas où $x_{ijut} = 0$, ou $x_{pquv} = 0$, les contraintes (16) et (17) impliquent $a^u_{ijtpqv} = a^u_{pqvijt} = 0$, aucune disjonction existe entre ces deux opération de transport. Dans le cas où $x_{ijut} = x_{pquv} = 1$, les contraintes (16), (17) et (18) impliquent $a^u_{ijtpqv} + a^u_{pqvijt} = 1$, deux cas sont alors possibles. Dans le premier cas, $a^u_{ijtpqv} = 1$ et l'opération de transport $O_{i,j,u,t}$ précède l'opération de transport $O_{p,q,u,v}$, ou dans le second cas, $a^u_{pqvijt} = 1$ et l'opération de transport $O_{p,q,u,v}$ précède l'opération de transport $O_{i,j,u,t}$. Ces deux cas sont illustrés sur la Figure 4-15 ;



a) cas où $a^{u}_{ijtpqv} = 1$ *b) cas où $a^{u}_{pqvijt} = 1$*

Figure 4-15. Illustration des contraintes (16), (17) et (18)

(19) dans le cas où $a^{u}_{ijtpqv} = 1$, cette contrainte se réécrit $B_{pquv} \geq A_{ijut} + t_{jp}$, ce qui signifie que l'opération de transport $O_{p,q,u,v}$ au départ de l'activité A_p ne peut débuter que lorsque l'opération de transport $O_{i,j,u,t}$ est arrivée sur l'activité A_j et que le véhicule s'est ensuite déplacé jusqu'à l'activité A_p, ce cas de figure est représenté sur la Figure 4-16. Dans le cas où $a^{u}_{ijtpqv} = 0$, la valeur de H implique que l'inégalité est trivialement vérifiée ;

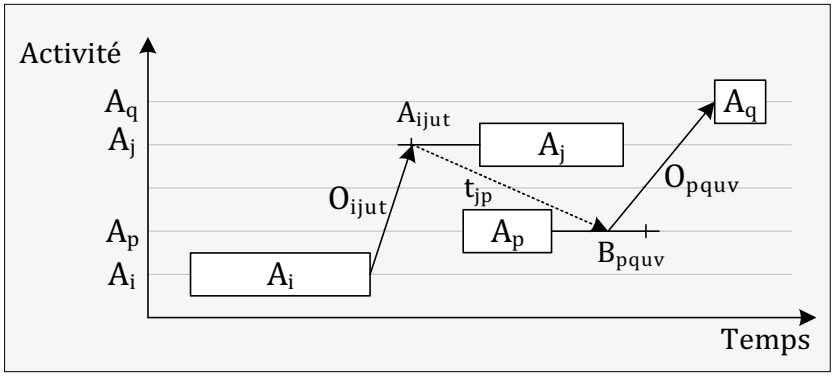


Figure 4-16. Illustration de la contrainte (19)

(20) ces contraintes permettent de supprimer les symétries entre les opérations de transport effectuées entre les mêmes activités par le même véhicule. Les numéros des opérations de transport sont triés par ordre décroissant sur les quantités transportées.

Définition des variables :

(21) les variables y_{ijut} sont entières ;

(23) et (24) les variables A_{ijut}, B_{ijut} sont réelles ;

(22) et (25) les variables x_{ijut}, a^{u}_{ijtpqv} sont binaires.

La formulation linéaire comprenant les contraintes des formulations $P1$ et $P2$ définit complètement le problème du RCPSPR.

$$P2:\begin{cases}
\forall (A_i, A_j) \in V^2 & \sum_{u\in T}\sum_{t\in X_{ij}} y_{ijut} = \varphi_{ij} & (10)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & y_{ijut} \le min(b_i, b_j, C_u)x_{ijut} & (11)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & x_{ijut} \le y_{ijut} & (12)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & A_{ijut} \ge B_{ijut} + t_{ij} + (x_{ijut} - 1).H & (13)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & S_j \ge A_{ijut} + (x_{ijut} - 1).H & (14)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & B_{ijut} \ge S_i + p_i + (x_{ijut} - 1).H & (15)\\
\forall (A_i, A_j, A_p, A_q) \in V^4,\ \forall T_u \in T,\ \forall (t,v) \in X_{ij}^2 & a^u_{ijtpqv} + a^u_{pqvijt} \le x_{ijut} & (16)\\
\forall (A_i, A_j, A_p, A_q) \in V^4,\ \forall T_u \in T,\ \forall (t,v) \in X_{ij}^2 & a^u_{ijtpqv} + a^u_{pqvijt} \le x_{pquv} & (17)\\
\forall (A_i, A_j, A_p, A_q) \in V^4,\ \forall T_u \in T,\ \forall (t,v) \in X_{ij}^2 & a^u_{ijtpqv} + a^u_{pqvijt} \ge 1 - (2 - x_{ijut} - x_{pquv}).H & (18)\\
\forall (A_i, A_j, A_p, A_q) \in V^4,\ \forall T_u \in T,\ \forall (t,v) \in X_{ij}^2 & B_{pquv} \ge A_{ijut} + t_{jp} + (a^u_{ijtpqv} - 1).H & (19)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in [\![1, min(b_i, b_j) - 1]\!]\} & y_{ijut+1} \le y_{ijut} & (20)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & y_{ijut} \in \mathbb{N} & (21)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & x_{ijut} \in \{0,1\} & (22)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & A_{ijut} \in \mathbb{R} & (23)\\
\forall (A_i, A_j) \in V^2,\ \forall T_u \in T,\ \forall t \in X_{ij} & B_{ijut} \in \mathbb{R} & (24)\\
\forall (A_i, A_j, A_p, A_q) \in V^4,\ \forall T_u \in T,\ \forall (t,v) \in X_{ij}^2 & a^u_{ijtpqv} \in \{0,1\} & (25)
\end{cases}$$

Figure 4-17. Deuxième partie du programme linéaire pour le RCPSPR

4.4 Proposition pour la résolution du RCPSPR

4.4.1 Approche de résolution proposée et contributions

L'approche de résolution proposée dans ce chapitre se base : 1) sur une métaheuristique qui va parcourir l'ensemble des vecteurs d'activités, 2) sur une fonction « d'évaluation » qui permet à partir d'un vecteur d'activités d'obtenir une solution du RCPSPR. Cette approche de résolution est basée sur une représentation indirecte de la solution avec un vecteur d'activités (Figure 4-18). Un vecteur d'activités est une permutation de l'ensemble des activités à ordonnancer dans le problème du RCPSPR. Ce type d'approche permet d'associer à un vecteur d'activités une solution du RCPSPR, et la fonction définit une application mais pas une bijection puisqu'il est tout à fait possible qu'une solution du RCPSPR puisse être obtenue à partir de plusieurs vecteurs d'activités différents au sens lexicographique.

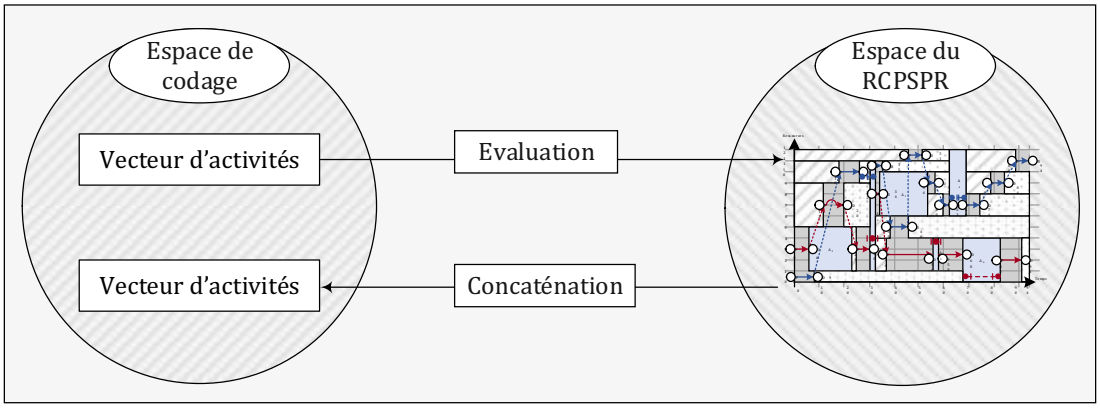


Figure 4-18. Les espaces de codages pour la construction d'une solution pour le RCPSPR

La première étape de l'évaluation d'un vecteur d'activité porte sur l'ordonnancement et permet la création d'un flot. Ce flot permet la définition des opérations de transfert. La deuxième étape génère, à partir de ces opérations de transfert, des opérations de transport prenant en compte les contraintes de capacités des véhicules. Une troisième étape permet de construire un tour géant à l'aide des opérations de transport précédemment définies. Un algorithme de type SPLIT est ensuite exécuté sur ce tour géant pour définir l'affectation des véhicules aux opérations finales de transport. Á l'issue du SPLIT, un graphe disjonctif orienté est obtenu, une solution du RCPSPR est obtenue en évaluant ce graphe à l'aide d'un plus long chemin. Une fois l'évaluation terminée, une procédure de concaténation permet de transformer une solution du RCPSPR en un vecteur d'activités. Cette sixième étape est également représentée sur la Figure 4-19.

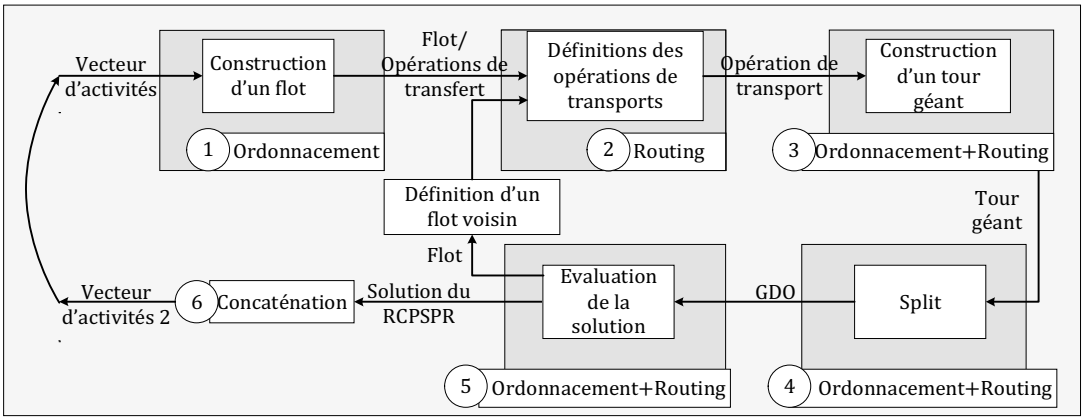


Figure 4-19. Construction d'une solution pour le RCPSPR

La résolution du RCPSPR repose sur six étapes :

- une méthode permettant la construction d'un flot afin de connaître les opérations de transfert (étape 1, Figure 4-19) ;
- une méthode permettant la définition des opérations de transport à partir des opérations de transfert en prenant en compte les contraintes de capacité sur les véhicules (étape 2, Figure 4-19) ;
- une méthode permettant la construction d'un tour géant à partir des opérations de transport (étape 3, Figure 4-19) ;
- une méthode SPLIT dédiée qui permet de définir les opérations finales de transport en regroupant des opérations de transport et d'affecter les véhicules à celles-ci (étape 4, Figure 4-19) ;
- une méthode d'évaluation permettant l'obtention d'une solution avec un ordonnancement au plus tôt (étape 5, Figure 4-19) ;
- une méthode de concaténation pour transformer une solution du RCPSPR en vecteur d'activités (étape 6, Figure 4-19).

Á ces six étapes et afin d'améliorer la solution finale du RCPSPR peut s'ajouter une étape permettant de modifier le flot d'une solution. Cette étape, nommée « Définition d'un flot voisin », permet de générer plusieurs flots à partir d'un même vecteur d'activités avec une notion de recherche locale sur le flot.

Ces différentes étapes sont développées dans les sections suivantes ainsi que la méthode permettant la génération d'un vecteur d'activités. Ces différentes étapes de résolution répondent à plusieurs questions soulevées dans le problème du RCPSPR :

1. Quel est la quantité de ressource échangée entre deux activités ?
2. Combien d'opérations de transport sont réalisées entre deux activités et avec quelle quantité de ressources ?
3. Quel véhicule effectue chacune des opérations de transport ?
4. Dans quel ordre les opérations de transport d'un même véhicule sont effectuées ?
5. Á quelles dates les activités et les opérations de transport peuvent-elles débuter ?

La première étape permet de répondre à la première question grâce à la définition d'un flot. La quantité de ressource échangée φ_{ij} entre deux activités $A_i$et A_j est caractérisée par $\varphi_{ij} \in [\![0, min(b_i, b_j)]\!]$. La réponse à la deuxième question est donnée par les étapes 2 et 4. L'étape 2 permet le découpage des opérations de transfert en opérations de

transport pour permettre à n'importe quel véhicule d'effectuer chaque opération de transport, cela en vue d'un regroupement possible des opérations lors du SPLIT pendant la quatrième étape. La quatrième étape permet aussi de répondre à la troisième question en affectant les véhicules aux opérations de transport. La réponse à la quatrième question est donnée par l'étape 3. Le tour géant détermine l'ordre des opérations de transport. Enfin, la dernière étape de l'évaluation (étape 5), évalue la solution obtenue grâce aux précédentes étapes et détermine les dates de début au plus tôt des opérations de transport et des activités.

Ces différentes étapes sont développées dans les sections suivantes ainsi que la méthode permettant la génération d'un vecteur d'activités.

4.4.2 Génération du vecteur d'activités

Un vecteur d'activités w se définit comme un vecteur de permutations sur les activités, dont l'ordre est en accord avec les contraintes de précédence du RCSPSR. Un vecteur d'activités peut être obtenu en classant les activités par niveau. Les niveaux sont calculés grâce à un plus long chemin dans le graphe de précédence, de telle sorte que $niveau(A_i) < niveau(A_j)$ si $(A_i, A_j) \in E$. Les activités ayant le même niveau peuvent être permutées sans risque de violer les contraintes de précédence.

Le graphe de précédence représenté sur la Figure 4-1 permet la classification des activités en quatre niveaux :

- niveau 0 : activité A_0
- niveau 1 : activités A_1, A_2 et A_3
- niveau 2 : activités A_4, A_5 et A_6
- niveau 3 : activité A_7

Le vecteur d'activités $w = [A_0, A_3, A_2, A_1, A_4, A_5, A_6, A_7]$ est un vecteur possible, il respecte le classement imposé par les niveaux et les contraintes de précédence. On peut noter que le vecteur d'activités $w = [A_0, A_3, A_1, A_4, A_5, A_6, A_2, A_7]$ respecte les contraintes de précédence mais ne respecte pas le classement par niveau des activités, l'activité A_2étant de niveau 1 est classée après les activités de niveau 2. Cette méthode de construction de vecteur d'activités ne permet pas l'accès à tous les vecteurs respectant les contraintes de précédence.

Pour obtenir un vecteur voisin, un opérateur de permutation est introduit. Cet opérateur, $swap(u, v)$, permute les deux activités aux positions u et v dans le vecteur d'activités. Cet opérateur n'effectue l'échange des activités que lorsqu'aucune contrainte de précédence n'est violée. Grace à cet opérateur tous les vecteurs d'activité peuvent être obtenus.

4.4.3 Construction d'un flot (première étape de l'évaluation)

Étant donné que le problème intégré du RCPSPR ne peut pas être résolu efficacement de manière séquentielle (en prenant en compte d'abord le problème du RCPSP puis le problème de tournée), il semble difficile de fournir une solution de problème de flot de bonne qualité pour le problème du RCPSPR. Par conséquent, on envisage une approche efficace en temps de calcul permettant la construction rapide d'une solution du flot. Dans les sections suivantes, les expérimentations numériques prouvent que cette assertion est pertinente puisque la solution optimale du problème de flot liée au RCPSP ne permet pas l'obtention de la solution optimale du RCPSPR.

Les solutions étant codées comme un vecteur d'activités w, la construction d'un graphe flot $G_\varphi(w)$ à partir du vecteur d'activités w doit satisfaire les demandes en ressources des activités en respectant la contrainte de conservation du flot et les contraintes de précédence entre les activités imposées par le vecteur d'activités w. Pour obtenir un algorithme de construction fortement efficace en temps de calcul (et en évitant une procédure coûteuse de réparation), il est possible de créer un graphe flot $G_\varphi(w)$ acyclique en introduisant une contrainte sur les flots liée à l'ordre des activités dans le vecteur w tel que $\forall\, j < i,\ \varphi_{w(i)w(j)} = 0$.

En s'aidant des propriétés du graphe flot et du vecteur d'activités, il est possible de proposer une heuristique de construction basée sur la création prioritaire d'un flot entre les activités proches dans le vecteur d'activités. Cette heuristique de construction (Algorithme 1) fournit une solution du problème de flot par propagation du flot de ressources d'une activité à l'autre en respectant l'ordre des activés dans le vecteur w.

Algorithme 1 . Construction d'un flot

```
1.  procedure Flow construction
2.  input parameters
3.    w : Activities feasible schedule
4.  output parameters
5.    φ : the flow
6.  local variables
7.    in : array 1..n+2 of integer
8.    out : array 1..n+2 of integer
9.  begin
10.   | ∀i ∈ [[1,n+2]], in(i) = b_w(i)  ;  ∀i ∈ [[1,n+2]], out(i) = b_w(i)
11.   | j = 1
12.   | while (j ≤ n+1) loop
13.   | | k:= j+1
14.   | | while (out(j) ≠ 0) and (k ≤ n+2) loop
15.   | | | φ_w(j),w(k) = min[out(j); in(k)]
16.   | | | out(j) := out(j) − φ_j,k
17.   | | | in(k) := in(k) − φ_j,k
18.   | | | k:= k+1
19.   | | end while
20.   | | j:= j+1
21.   | end while
22.   end
```

L'Algorithme 1 se compose de deux boucles while : la première entre les lignes 12 et 21 qui parcourt le vecteur d'activités w et la seconde entre les lignes 14 et 19 permettant la distribution des ressources de l'activité $w(j)$ à l'activité $w(k)$. Dans cette seconde boucle, le flot $\varphi_{w(j),w(k)}$ créé ne peut dépasser ni la quantité $out(j)$ de ressource disponible sur l'activité $w(j)$, ni la quantité maximale $in(k)$ de ressources que l'activité $w(k)$ doit recevoir ; cette contrainte est exprimée à travers la ligne 15. Les lignes 16 et 17 permettent de mettre à jour l'état des deux activités concernées par la propagation du flot. La quantité de ressource restant à transférer à partir de l'activité $w(j)$ est décrémentée, tout comme la quantité de ressource restant à transférer vers l'activité $w(k)$. Toutes les ressources disponibles sur l'activité initiale A_0 sont propagées et restituées à l'activité finale A_{n+1} à la fin de l'algorithme.

Exemple de construction d'un flot :

Á partir du vecteur d'activités $w = [A_0, A_3, A_2, A_1, A_4, A_5, A_6, A_7]$ donné dans la partie précédente et du problème défini dans le Tableau 4-1 et la Figure 4-1, le flot représenté par la Figure 4-20 peut être construit. Le flot $\varphi_{i,j}$ est représenté par un arc de coût positif égal à la quantité de ressources échangée entre les deux activités dans le vecteur w. La première itération de l'algorithme sur cet exemple ($j = 1$ et $k = 2$), permet la création du flot $\varphi_{0,3} = 4$, l'activité A_3 reçoit toutes ses ressources de l'activité A_0 ($in(2) = 0$). Lors de la deuxième itération ($j = 1$ et $k = 3$), les ressources restantes sur l'activité A_0 ($out(1) = 8$) sont totalement transférées vers l'activité A_2 avec un flot $\varphi_{0,2} = 8$. Toutes les ressources disponibles en A_0 ont été transférées ($out(1) = 0$).

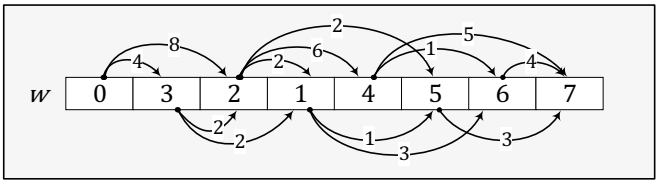


Figure 4-20. Exemple de solution du problème de flot à partir d'un vecteur d'activité w

Le flot défini sur la Figure 4-20 peut également être représenté sous forme matricielle avec $\varphi \in \mathcal{M}_{n+2,n+2}(\mathbb{N})$ telle que, $\varphi(i,j) = \varphi_{ij}$ et alors :

$$\varphi = \begin{pmatrix} 0 & 0 & 8 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 0 \\ 0 & 2 & 0 & 0 & 6 & 2 & 0 & 0 \\ 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \in \mathcal{M}_{n+2,n+2}(\mathbb{N})$$

La définition d'un flot permet la définition des opérations de transfert, toutes les quantités qui doivent être transférées entre les activités sont par conséquent définies. Le flot φ_{ij} entre l'activité A_i et l'activité A_j définit une opération de transfert.

4.4.4 Définition d'un flot voisin : recherche locale sur le flot

Le flot construit avec l'heuristique présentée précédemment possède la particularité de pouvoir être représenté par une matrice strictement triangulaire supérieure φ^w en respectant l'ordre des activités $\big(w(0), \dots, w(n+1)\big)$ données par le vecteur d'activités w, tel que $\varphi^w(i,j) = \varphi_{w(i),w(j)}$.

Pour définir un flot voisin à φ^w, une matrice de limitation du flot M_φ est introduite. Cette matrice peut être précalculée grâce aux demandes des activités et est valable pour un problème sans dépendre du flot ou du vecteur d'activités. Á l'état initial, les éléments de cette matrice sont donnés par les demandes des activités, on a alors $\forall (i,j) \in [\![0, n+2]\!], M_\varphi(i,j) = min(b_i, b_j)$. Cette matrice exprime les quantités maximales de ressource pouvant être échangées entre deux activités, en prenant en compte les contraintes de précédence.

L'algorithme de recherche locale parcourt la matrice φ^w de droite à gauche et de bas en haut afin de trouver un flot modifiable entre deux activités. Un flot $\varphi^w(i,j) > 0$ avec $i < j$ est modifiable si, en effectuant des modifications élémentaires sur les flots $\varphi^w(k,l)$ tels que $k \geq i$ et $l \geq j$, un nouveau flot respectant les contraintes de conservation peut être obtenu. Si le flot $\varphi^w(i,j) > 0$ est modifiable, la matrice de limitation est modifiée telle que $M_\varphi(i,j) = M_\varphi(i,j) - 1$. Un flot voisin peut alors être construit grâce à la méthode présentée dans la partie précédente en prenant en compte la matrice M_φ.

Á l'issue de l'évaluation de la solution (à la fin des cinq étapes) si le flot voisin a permis l'obtention d'une meilleure solution que le flot initial, la recherche locale du flot recommence le parcours de la matrice φ^w afin de trouver un flot modifiable sans modifier la matrice M_φ. Tant qu'un flot voisin n'est pas améliorant, l'algorithme remonte dans la matrice φ^w.

Exemple de recherche locale sur un flot:

En s'appuyant sur le problème énoncé en exemple et le flot construit précédemment, on peut définir les matrices $\varphi^w \in \mathcal{M}_{n+2,n+2}(\mathbb{N})$ et $M_\varphi \in \mathcal{M}_{n+2,n+2}(\mathbb{N})$ telles que :

$$\varphi^w = \begin{pmatrix} 0 & 4 & 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 6 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad M_\varphi = \begin{pmatrix} 0 & 4 & 10 & 4 & 6 & 3 & 4 & 12 \\ 0 & 0 & 4 & 4 & 4 & 3 & 4 & 4 \\ 0 & 4 & 0 & 4 & 6 & 3 & 4 & 10 \\ 0 & 4 & 4 & 0 & 4 & 3 & 4 & 4 \\ 0 & 0 & 6 & 4 & 0 & 3 & 4 & 6 \\ 0 & 3 & 3 & 0 & 3 & 0 & 3 & 3 \\ 0 & 0 & 4 & 4 & 4 & 3 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

L'algorithme de recherche locale parcourt la matrice φ^w de droite à gauche et de bas en haut afin de trouver un flot entre deux activités modifiables :

Etape 1. $\varphi^w(6,7) = 4$, ce flot n'est pas modifiable puisqu'un flot de la forme $\varphi^w(6,k)$ avec $k > 7$ ne peut être défini dans un problème avec sept activités.

Etape 2. $\varphi^w(5,7) = 3$, ce flot n'est pas modifiable pour la même raison que le précédent.

Etape 3. $\varphi^w(5,6) = 0$, ce flot n'est pas modifiable puisqu'il est nul.

Etape 4. $\varphi^w(4,7) = 0$, ce flot n'est toujours pas modifiable.

Etape 5. $\varphi^w(4,6) = 1$, ce flot est modifiable.

En effet avec les opérations élémentaires suivantes dans la sous-matrice $\varphi^w_{\{6,7\}\{4,5\}}$, on peut définir un flot voisin valide :

$$\varphi^w_{\{6,7\}\{4,5\}} = \begin{pmatrix} 1 & 5 \\ 0 & 3 \end{pmatrix} \rightarrow + \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \rightarrow \varphi^w_{\{6,7\}\{4,5\}} = \begin{pmatrix} 0 & 6 \\ 1 & 2 \end{pmatrix}$$

La matrice de limitation du flot est alors modifiée telle que $M_\varphi(w(4), w(6)) = \varphi^w(4,6) - 1 = 0$. Le flot voisin généré est représenté sur la Figure 4-21.

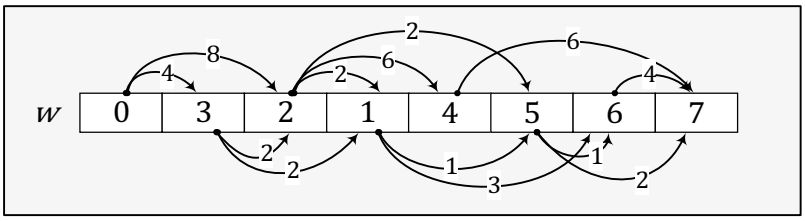


Figure 4-21. Premier flot voisin généré

Si ce flot voisin après évaluation mène à une solution du RCPSPR moins bonne que le flot initial, la dernière modification sur la matrice de limitation est annulée et l'algorithme de recherche locale continue le parcours de la matrice φ^w. Sinon, la matrice $M_\varphi$est sauvegardée et le parcours de la matrice φ^w est réinitialisé.

4.4.5 Définition des opérations de transport (deuxième étape de l'évaluation)

Le véhicule effectuant chaque opération de transport n'étant pas déterminé à cette étape de l'algorithme, la notation $*$ est introduite pour rester générique. De plus, pour faciliter la lecture de ce chapitre, une opération de transport peut être notée $O_{i,j,u,x}$ ou (i, j, u, x). Une opération de transfert entre l'activité A_i et l'activité A_j associée au flot φ_{ij} est notée (i, j, φ_{ij}).

Une opération de transfert entre une activité A_i et une activité A_j peut se ramener à une seule opération de transport si, $\forall u \in [\![1, N]\!], \varphi_{ij} \leq C_u$, *i.e.*, si tous les véhicules peuvent transporter la quantité φ_{ij} sans aller-retour. Cependant, si $\exists u \in [\![1, N]\!], \varphi_{ij} \geq C_u$, l'opération de transfert entre l'activité A_i et l'activité A_j peut demander plusieurs opérations de transport suivant l'affectation des véhicules. Pour une opération de transfert, il est donc possible de définir un lot d'opérations de transport $O_{ij} = \bigcup_k (i, j, *, x)_k \,/ \sum_k x_k = \varphi_{ij}$. En d'autres termes, un lot d'opérations de transport O_{ij} est un ensemble ordonné d'opérations de transport entre les activités A_i et A_j, avec une certaine quantité de ressource à transporter.

Par exemple, un lot d'opérations de transport unitaires $O_{ij} = \bigcup_k (i, j, *, 1)_k$ définit un ensemble d'opérations de transport et chacune d'elle peut être affectée à n'importe quel véhicule. Pour un lot d'opérations de transport non unitaires $O_{ij} = \bigcup_k (i, j, *, x)_k$, une opération de transport $(i, j, *, x)$ ne peut être affectée qu'à un véhicule T_u de capacité $C_u \geq x$.

Les lots d'opérations de transport sont utilisés pour définir le tour géant dans l'étape suivante et permettent la création des tournées avec le SPLIT dans l'étape 4. La quantité de ressource dans chaque opération de transport $(i, j, *, x) \in O_{ij}$ a un impact sur l'exécution du SPLIT puisque la procédure SPLIT permet l'agrégation de plusieurs opérations de transport (voir Prins (2004)). La mauvaise définition d'un lot d'opérations de transport entraine la définition d'un tour géant dans lequel certaines agrégations ne peuvent être explorées en raison de la capacité des véhicules. En effet, plusieurs opérations de transport entre deux même activités peuvent être agrégées (dans la procédure SPLIT) si elles sont consécutives dans le tour géant. Cette agrégation crée une

opération de transport à partir de plusieurs opérations de transport (avec une quantité de ressource égale à la somme des quantités de ressource dans les opérations de transport).

Par exemple, un lot d'opérations de transport unitaires $O_{ij} = \{(i,j,*,1)(i,j,*,1)(i,j,*,1)\}$ peut-être agrégé pour obtenir une opération de transport $(i,j,*,3)$ avec trois unités de ressource. L'agrégation n'est possible que s'il existe un véhicule qui peut transporter une telle quantité de ressource. Ce problème peut survenir dans certains cas, par exemple, si un lot d'opérations de transport $O_{ij} = \{(i,j,*,2)(i,j,*,2)\}$ est créé à partir d'une opération de transfert $(i,j,*,\varphi_{ij})$ avec $\varphi_{ij} = 4$. Dans cet exemple, la procédure SPLIT ne pourra pas agréger les opérations de transport afin d'obtenir les opérations de transport $(i,j,*,3)$ et $(i,j,*,1)$.

Création de lots d'opérations de transport unitaires

Un lot d'opérations de transport unitaires permet de créer le nombre maximal d'opérations de transport (égal à la valeur du flot) et de permettre l'affectation des opérations de transport à tous les véhicules grâce aux possibilités d'agrégation.

Pour toute opération de transfert (i,j,φ_{ij}), le problème consiste à créer un lot d'opérations de transport constitué de φ_{ij} opérations de transport unitaires $(i,j,*,1)$ permettant de modéliser toutes les combinaisons d'agrégation et d'affectation. Ce lot d'opérations de transport est défini par $O_{ij} = \{(i,j,*,1), \dots, (i,j,*,1)\}$ avec $Card(O_{ij}) = \varphi_{ij}$. La procédure SPLIT à l'étape 4 peut effectuer toutes les agrégations possibles. En dépit de cet avantage, une telle situation n'est pas profitable algorithmiquement. D'abord, la procédure SPLIT doit considérer toutes les agrégations d'opérations de transport (et affectation aux véhicules) et, deuxièmement, la longueur du tour géant est directement impactée.

Points-clés pour la création efficace de lots d'opérations de transport

Pour améliorer les performances de la procédure SPLIT, il est intéressant de fournir des lots d'opérations de transport de longueur minimale (nombre minimal d'opérations de transport), mais qui entraîne les mêmes agrégations d'opérations de transport que celles engendrées par des lots d'opérations de transport unitaires. On note $O_{ij}^{*} = \{(i,j,*,\varphi_1) \dots (i,j,*,\varphi_m)\}$ le lot d'opérations de transport entre l'activité A_i et l'activité A_j avec le nombre minimal d'opérations de transport permettant les mêmes agrégations d'opérations de transport que celles engendrées par des lots d'opérations de transport unitaires.

La Figure 4-22 illustre différents cas d'agrégations en fonction de lots d'opérations de transport différents sur un exemple avec une flotte hétérogène de trois véhicules, avec $C_1 = 5,\ C_2 = 3,\ C_3 = 2$ et une quantité totale de ressource de $\varphi_{ij} = 12$ à transporter.

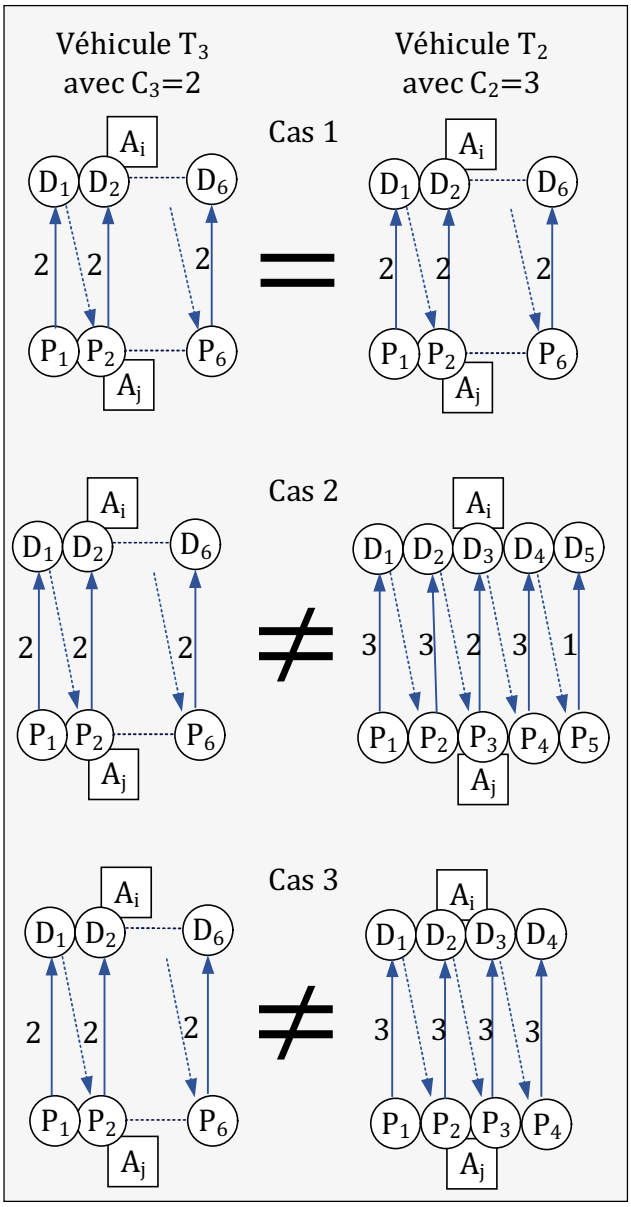


Figure 4-22. Agrégation des lots d'opérations de transport en fonction des véhicules

Cas 1:
Dans le premier cas, le lot d'opérations de transport n'est constitué que d'opérations de transport avec 2 unités de ressource, tel que : $O_{ij} = \{(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)\}$. Quel que soit le véhicule (T_3 ou T_2) choisi pour faire le transport de toutes les ressources, le nombre final d'opérations de transport est égal à six puisqu'aucune agrégation ne peut être effectuée (Figure 4-22- Cas 1).

Cas 2:
Si O_{ij} est composé d'opérations de transport avec 1 ou 2 unités de ressource, certaines opérations peuvent être agrégées. Avec $\phi_{ij} = \{(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)(i,j,*,1)(i,j,*,1)\}$, si toutes les ressources sont transportées par le véhicule T_3, celui-ci doit effectuer six opérations de transport, et si toutes les ressources sont transportées par le véhicule T_2, seules cinq opérations de transport peuvent être effectuées (Figure 4-22- Cas 2).

Cas 3:
Un lot d'opérations de transport engendrant de meilleures agrégations peut être défini, $O_{ij} = \{(i,j,*,2)(i,j,*,1),(i,j,*,1)(i,j,*,2)(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)\}$. Dans ce cas, le véhicule T_3, doit toujours effectuer six opérations de transport, mais le véhicule T_2, peut effectuer seulement quatre opérations de transport (Figure 4-22- Cas 3). Quelque soit le véhicule utilisé, le nombre d'opérations de transport effectuées est minimal (chaque opération de transport est réalisée avec un véhicule totalement chargé).

La Figure 4-22 prouve que pour la même opération de transfert, il est possible de définir différents lots d'opérations de transport (en fonction de l'ordre et des quantités des opérations de transport), conduisant à un nombre différent d'opérations de transport après

l'agrégation dû à l'affectation des véhicules. Par exemple, $O_{ij} = \{(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)\}$ (Figure 4-22 - Cas 3) est composé de huit opérations de transport. Celles-ci peuvent être agrégées pour le véhicule T_2 afin de définir seulement quatre opérations de transport.

Dans tous les cas présentés sur la Figure 4-22, le choix de O_{ij} n'a pas d'incidence sur l'agrégation pour le véhicule T_2. Un autre point peut être mis en évidence en prenant également en compte le véhicule T_1 sur le cas 3 précédent (Figure 4-23).

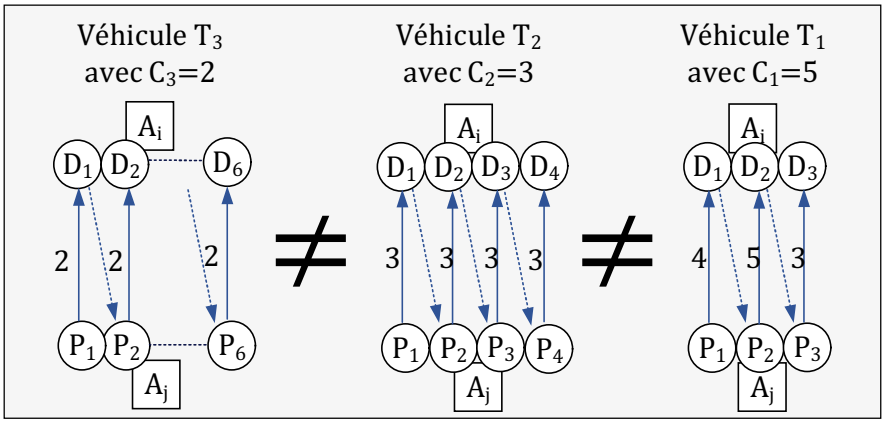


Figure 4-23. Agrégation d'un lot d'opérations de transport en fonction des véhicules

Le problème de définition des lots d'opérations de transport doit s'adapter aux véhicules pour anticiper l'agrégation / affectation effectuée dans la procédure SPLIT.

Proposition de méthode pour la création efficace de lots d'opérations de transport

Pour créer des lots d'opérations de transport favorables à la procédure de SPLIT, l'algorithme consiste à rechercher tous les points de coupure de tous les véhicules pris en compte séparément avec L_u, l'ensemble ordonné des points de coupure du véhicule T_u. L'ensemble des points de coupure d'un véhicule T_u définit une suite arithmétique de raison C_u. Notons $L = \bigcup_N L_u$ l'ensemble ordonné de tous les points de coupure de tous les véhicules (sans duplication).

Pour le véhicule T_1, le plus petit lot d'opérations de transport est égal à $O_{ij} = \{(i,j,*,5)(i,j,*,5)(i,j,*,2)\}$. Ce lot peut être obtenu directement grâce à l'ensemble des points de coupure du véhicule T_1. Le premier point de coupure vaut 0, le deuxième vaut $0 + 5 = 5$ et le troisième vaut $5 + 5 = 10$. Un quatrième point de coupure ne peut exister car $10 + 5 = 15 > \varphi_{ij} = 12$. Par conséquent, $L_1 = (0, 5, 10, 12)$. Une observation similaire s'applique au véhicule T_2, avec $O_{ij} = \{(i,j,*,3)(i,j,*,3)(i,j,*,3)(i,j,*,3)\}$ et l'ensemble des points de coupure $L_2 = (0, 3, 6, 9, 12)$. Pour le véhicule T_3, $O_{ij} = \{(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)\}$, l'ensemble des points de coupure est égal à $L_3 = (0, 2, 4, 6, 8, 10, 12.)$. Pour cet exemple, l'ensemble ordonné de tous les points de coupure de tous les véhicules est égal à $L = (0,2,3,4,5,6,8,9,10,12)$.

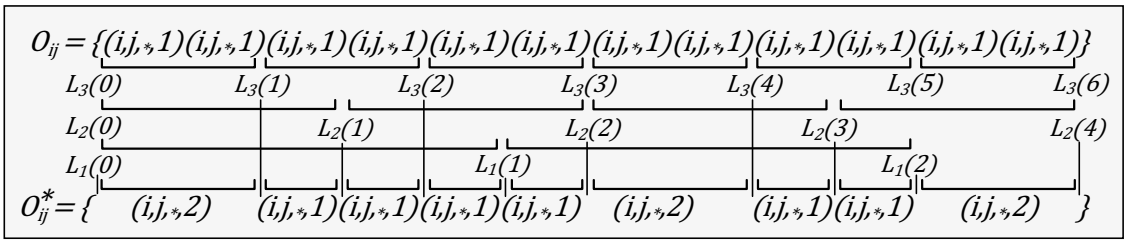


Figure 4-24. Représentation des points de coupure sur le lot d'opérations de transport unitaires avec trois véhicules, ($C_1 = 5,\ C_2 = 3,\ C_3 = 2$) et $\varphi_{ij} = 12$

Les points de coupure sont pris en compte sur le lot d'opérations de transport unitaires afin de créer de nouvelles opérations de transport. Ces opérations de transport sont définies entre deux points de coupure successifs dans L et définissent O^*_{ij}. Sur l'exemple énoncé dans cette partie, le meilleur lot d'opérations de transport est égal à $O^*_{ij} = \{(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,1)(i,j,*,1)(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)\}$.

Exemple :

Á partir du flot énoncé dans la partie précédente, des lots d'opérations de transport associés aux opérations de transfert sont calculés et présentés dans le Tableau 4-4. Dans ce problème, deux véhicules sont disponibles, tels que $C_1 = 4$ et $C_2 = 2$.

Tableau 4-4
Les lots d'opérations de transport associés aux opérations de transfert

Opération de transfert	Lot d'opérations de transport
(0,2,8)	$O^*_{02} = \{(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)\}$.
(0,3,4)	$O^*_{03} = \{(0,3,*,2)(0,3,*,2)\}$.
(1,5,1)	$O^*_{15} = \{(1,5,*,1)\}$.
(1,6,3)	$O^*_{16} = \{(1,6,*,2)(1,6,*,1)\}$.
(2,1,2)	$O^*_{21} = \{(2,1,*,2)\}$.
(2,4,6)	$O^*_{24} = \{(2,4,*,2)(2,4,*,2)(2,4,*,2)\}$.
(2,5,2)	$O^*_{25} = \{(2,5,*,2)\}$.
(3,1,2)	$O^*_{31} = \{(3,1,*,2)\}$.
(3,2,2)	$O^*_{32} = \{(3,2,*,2)\}$.
(4,2,1)	$O^*_{42} = \{(4,2,*,1)\}$.
(4,6,5)	$O^*_{46} = \{(4,6,*,2)(4,6,*,2)(4,6,*,1)\}$.
(5,7,3)	$O^*_{57} = \{(5,7,*,2)(5,7,*,1)\}$.
(6,7,4)	$O^*_{67} = \{(6,7,*,2)(6,7,*,2)\}$.

4.4.6 Construction d'un tour géant (troisième étape de l'évaluation)

La construction d'un tour géant consiste à parcourir l'ensemble des opérations de transfert $(w(i), w(j), \varphi_{w(i)w(j)})$ par ordre croissant sur les i et les j et à ajouter le lot d'opérations de transport correspondant dans le tour géant à la meilleure position possible. Au début de l'heuristique de construction, le tour géant est vide. Notons que cet algorithme n'ordonnance pas les opérations de transport une par une, mais par lot avec toutes les opérations de transport qui le compose. Cet algorithme de construction glouton est basé sur une procédure `best_insertion(λ, coût)` qui trouve la meilleure position d'insertion du dernier lot d'opérations de transport à insérer en respectant les contraintes de ressource et de précédence. Á chaque position d'insertion est associé un

coût égal au transport vide entre la dernière ou la première opération de transport du lot à insérer et les opérations déjà ordonnancées directement à droite ou à gauche. Le lot d'opérations de transport est alors inséré à la position de coût minimal.

Exemple :

Un tour géant peut être construit à partir de l'exemple détaillé au cours des précédentes étapes de l'évaluation, notamment avec les opérations de transfert et les lots d'opérations de transport dans le Tableau 4-4. Le Tableau 4-5 donne quelques étapes de la construction du tour géant.

Tableau 4-5
Étapes de construction du tour géant

Étape / Lot à insérer	Tour géant possible λ	Coût	Min
1 : $O_{03}^{*} = \{(0,3,*,2)(0,3,*,2)\}$	λ=$(\mathbf{0},\mathbf{3},*,\mathbf{2})(\mathbf{0},\mathbf{3},*,\mathbf{2})$	-	✓
2 : $O_{02}^{*} = \{(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)\}$	λ=(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,3,*,2)(0,3,*,2)	8	
	λ=$(\mathbf{0},\mathbf{3},*,\mathbf{2})(\mathbf{0},\mathbf{3},*,\mathbf{2})$(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)	6	✓
3 : $O_{32}^{*} = \{(3,2,*,2)\}$.	λ=(0,3,2)(0,3,2)(0,2,2)(0,2,2)(0,2,2)(0,2,2)$(\mathbf{3},\mathbf{2},\mathbf{2})$	5	✓
	λ=(0,3,2)(0,3,2)$(\mathbf{3},\mathbf{2},\mathbf{2})$(0,2,2)(0,2,2)(0,2,2)(0,2,2)	8	
4 : $O_{31}^{*} = \{(3,1,*,2)\}$.	λ=(0,3,*,2)(0,3,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(3,2,*,2)$(\mathbf{3},\mathbf{1},*,\mathbf{2})$	6	
	λ=(0,3,*,2)(0,3,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)$(\mathbf{3},\mathbf{1},*,\mathbf{2})$(3,2,*,2)	4	✓
	λ=(0,3,*,2)(0,3,*,2)$(\mathbf{3},\mathbf{1},*,\mathbf{2})$(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(3,2,*,2)	8	
...			
13: $O_{67}^{*} = \{(6,7,*,2)(6,7,*,2)\}$.	λ=(0,3,*,2)(0,3,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(3,1,*,2)(3,2,*,2)(2,1,*,2)(2,4,*,2)(2,4,*,2)(2,4,*,2)(1,5,*,1)(1,6,*,2)(1,6,*,1)(4,7,*,2)(4,7,*,2)(4,7,*,1)(4,6,*,1)(2,5,*,2)(5,7,*,2)(5,7,*,1)$(\mathbf{6},\mathbf{7},*,\mathbf{2})(\mathbf{6},\mathbf{7},*,\mathbf{2})$	9	
	λ=(0,3,*,2)(0,3,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(3,1,*,2)(3,2,*,2)(2,1,*,2)(2,4,*,2)(2,4,*,2)(2,4,*,2)(1,5,*,1)(1,6,*,2)(1,6,*,1)(4,7,*,2)(4,7,*,2)(4,7,*,1)(4,6,*,1)(2,5,*,2)$(\mathbf{6},\mathbf{7},*,\mathbf{2})(\mathbf{6},\mathbf{7},*,\mathbf{2})$(5,7,*,2)(5,7,*,1)	16	
	λ=(0,3,*,2)(0,3,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(3,1,*,2)(3,2,*,2)(2,1,*,2)(2,4,*,2)(2,4,*,2)(2,4,*,2)(1,5,*,1)(1,6,*,2)(1,6,*,1)(4,7,*,2)(4,7,*,2)(4,7,*,1)(4,6,*,1)$(\mathbf{6},\mathbf{7},*,\mathbf{2})(\mathbf{6},\mathbf{7},*,\mathbf{2})$(2,5,*,2)(5,7,*,2)(5,7,*,1)	8	✓

Lors de la première étape de construction du tour géant (Tableau 4-5), le lot d'opérations de transport $O_{03}^{*} = \{(0,3,*,2)(0,3,*,2)\}$ est inséré directement dans le tour géant puisque $\lambda = \emptyset$. Le tour géant initial est alors égal à λ=(0,3,*,2)(0,3,*,2). La deuxième étape consiste à insérer le lot d'opérations de transport O_{02}^{*}={(0,2,*,2)(0,2,*

$,2)(0,2,*,2)(0,2,*,2)\}$. Ce lot peut être inséré à la fin du tour géant ou au début, *i.e.* soit après soit avant le lot d'opérations déjà inséré dans le tour géant. La meilleure position d'insertion est déterminée grâce au coût d'insertion. Si l'insertion est effectuée en fin, les opérations $(0,2,*,2)(0,3,*,2)$ sont alors consécutives. Dans le cas à un véhicule, cette suite d'opérations implique un déplacement à vide du véhicule entre l'activité A_2 et l'activité A_0, le coût d'insertion est alors égal à $t_{2,0} = 8$. Si l'insertion est effectuée au début, les opérations $(0,3,*,2)(0,2,*,2)$ sont alors consécutives. Dans le cas à un véhicule, cette suite d'opérations implique un déplacement à vide du véhicule entre l'activité A_3 et l'activité A_0, le coût d'insertion est alors égal à $t_{3,0} = 6$. Cette insertion est meilleure que la précédente puisque le coût d'insertion est inférieur. Après insertion, le tour géant est désormais égal à λ=$(0,3,*,2)(0,3,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)$. Lors de la troisième étape (insertion d'un troisième lot d'opérations de transport), deux lots sont déjà insérés dans le tour géant, le nouveau lot peut donc possiblement être inséré à trois positions différentes. Cependant l'insertion du lot $O_{32}^{*} = \{(3,2,*,2)\}$ ne peut être effectuée au début du tour géant. En effet, l'opération de transport $(3,2,*,2)$ modélisant une opération de collecte sur l'activité A_3 ne peut avoir lieu avant l'opération de transport $(0,3,*,2)$ modélisant une opération de livraison sur l'activité A_3. Toutes les ressources doivent être livrées puis collectées sur chaque activité non fictive. Tous les lots sont insérés au mieux durant la procédure composée de 13 étapes. Á l'issue de la procédure, le tour géant est égal à $\lambda = (0,3,*,2)(0,3,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(3,1,*,2)(3,2,*,2)(2,1,*,2)(2,4,*,2)(2,4,*,2)(2,4,*,2)(1,5,*,1)(1,6,*,2)(1,6,*,1)(4,7,*,2)(4,7,*,2)(4,7,*,1)(4,6,*,1)(6,7,*,2)(6,7,*,2)(2,5,*,2)(5,7,*,2)(5,7,*,1)$. Ce tour géant est composé de 24 opérations de transport.

4.4.7 Méthode SPLIT (quatrième étape de l'évaluation)

La procédure SPLIT « classique », utilisée dans les problèmes de tournées de véhicules ne prend en compte que des contraintes directement liées au routage. On peut citer des contraintes concernant une limite sur le nombre de véhicules et/ou leurs capacités (voir Duhamel *et al.*, 2012). Comme souligné par (Duhamel *et al.*, 2012), un label sauvegardé sur un nœud doit permettre de définir, par un ensemble d'information, une solution partielle du problème grâce à un coût et à une description complète de l'état du système. Une approche de résolution basée sur une procédure SPLIT repose sur les caractéristiques suivantes :

- la définition d'un label pour modéliser l'état du système ;
- une règle de propagation pour créer de nouveaux labels ;
- une règle de dominance pour sauvegarder uniquement les labels non dominés sur un nœud.

L'originalité du SPLIT présenté dans cette partie, dédié au RCPSPR, repose sur le fait que l'état du système englobe à la fois l'état de la flotte de véhicules et l'état des activités :

- l'état de la flotte de véhicules se compose de trois informations pour chaque véhicule : la position du véhicule, la date de départ et la date d'arrivée. Ces données sont nécessaires même si la flotte est homogène.
- l'état des activités est donné par la date de début au plus tôt de toutes les activités.

La procédure SPLIT consiste à construire un graphe auxiliaire acyclique H avec $nt + 1$ nœuds numérotés de 0 à nt (nt étant le nombre d'opérations de transport dans le tour

géant λ) où un arc d'un nœud i à $j \neq i+1$ représente une agrégation d'opérations de transport en une seule. Le sous-graphe entre un nœud i et un noeud j est noté $\mu_{ij} = \big(\lambda(i+1), \dots, \lambda(j)\big)$, ce sous-graphe est constitué de $j-i$ nœuds, chacun modélisant une opération de transport. Pour faciliter la lecture de cette partie, $\lambda^+(i)$ désigne l'opération de collecte de l'opération de transport $\lambda(i)$, $\lambda^-(i)$ désigne l'opération de livraison de l'opération de transport $\lambda(i)$ et $\lambda^{\varphi}(i)$ désigne le flot associé à l'opération de transport $\lambda(i)$. Le découpage optimal du tour géant (agrégation et affectation des opérations de transport) revient à trouver un plus court chemin dans le graphe H du nœud 0 au nœud nt.

Définition du label

Soit L_i^j le $i^{ème}$ label sur le nœud j, ce label permet de modéliser l'évaluation d'une solution partielle du problème dans laquelle les opérations de transport entre $\lambda(1)$ et $\lambda(j)$ sont ordonnancées. Un label $L_i^j = \big(V_i^j, S_i^j, (f_u, f_v)\big)$ est composé d'un $3v$-uplet V_i^j , d'un $n+1$-uplet S_i^j et d'un couple (f_u, f_v) :

- Le $3v$-uplet V_i^j contient à la fois les dates de départ et d'arrivé des véhicules et leurs positions, $V_i^j = \big(B_{ij}^1, \dots, B_{ij}^N, A_{ij}^1, \dots, A_{ij}^N, P_{ij}^1, \dots, P_{ij}^N\big)$, avec $P_{ij}^1, \dots, P_{ij}^N$ les positions de chaque véhicule ($P_{ij}^u = p$ signifie que le véhicule T_u est localisé sur l'activité A_p, $B_{ij}^1, \dots, B_{ij}^N$ les dates de départ au plus tôt de tous les véhicules, et $A_{ij}^1, \dots, A_{ij}^N$ les dates de départ au plus tôt de tous les véhicules.
- Le $n+1$-uplet S_i^j contient les dates de début au plus tôt de chacune des activités $S_i^j = \big(S_{ij}^0, \dots, S_{ij}^{n+1}\big)$;
- Le couple (f_u, f_v) fait référence à la position du label père dans le graphe $i.e.$ le label L_i^j a été généré à parti du label numéro f_u sur le nœud f_v.

Règle de propagation du label

Une fonction de propagation définie par $f: L \otimes T \rightarrow R$ permet de générer un nouveau label L_q^j à partir d'un label L_p^i, d'un coût dépendant du sous-graphe μ_{ij} et du choix du véhicule. Pour un sous-graphe μ_{ij}, le label L_q^j sur le nœud j est défini grâce au label L_p^i en utilisant le véhicule T_u avec les mises à jour suivantes :

$$
\begin{aligned}
B_{qj}^u &= \max\Big(S_{pi}^{\lambda^+(j)} + p_{\lambda^+(j)};\ A_{pi}^u + e_{P_{pi}^u, \lambda^+(j)}\Big) \\
B_{qj}^v &= \max\big(B_{pi}^v;\ A_{pi}^v\big) \,/\, v \neq u \\
A_{qj}^u &= B_{qj}^u + t_{\lambda^+(j), \lambda^-(j)} \\
P_{qj}^u &= \lambda^-(j) \\
S_{qj}^{\lambda^-(j)} &= \max\Big(S_{pi}^{\lambda^-(j)};\ A_{qj}^u\Big)
\end{aligned}
$$

Pour le label L_p^i, la règle de propagation envisage l'affection de l'opération de transport à tous les véhicules, N labels sont ainsi générés. Ces labels sont stockés ou non sur le nœud j destination en fonction des labels non-dominés déjà présents sur le nœud j.

Proposition d'une règle de dominance entre les labels

Afin de garder seulement les labels non-dominés sur chaque nœud, une règle de dominance doit être définie. Soit les deux labels L_p^i et L_q^i, L_p^i domine L_q^i ($L_q^i \ll L_p^i$), si les conditions suivantes sont vérifiées :
Condition 1. Tous les véhicules sont localisés au même endroit : $\forall T_u \in T, P_{ip}^u = P_{iq}^u$
Condition 2. $\forall T_u \in T, A_{ip}^u \leq A_{iq}^u$
Condition 3. $\forall T_u \in T, B_{ip}^u \leq B_{iq}^u$
Condition 4. $\exists T_u \in T, A_{ip}^u < A_{iq}^u$ ou $\exists T_u \in T, B_{ip}^u < B_{iq}^u$

La date de début au plus tôt des activités n'est pas un critère pertinent pour la règle de dominance puisque si toutes les dates liées aux véhicules dans le label L_p^i sont inférieures ou égales à toutes les dates liées aux véhicules dans le label L_q^i, alors $\nexists k / S_{iq}^k < S_{ip}^k$.

Si L_p^i ne domine pas L_q^i ($L_q^i \not\ll L_p^i$), cela n'implique pas que L_q^i domine L_p^i. En effet, si $L_q^i \not\ll L_p^i$ et si $L_p^i \not\ll L_q^i$, alors les labels L_p^i et L_q^i sont incomparables. Par conséquent, si $\exists k$ sur le nœud i / $L_p^i \ll L_k^i$, alors le label L_p^i n'est pas inséré sur le nœud i.

De plus, chaque label L_k^i / $L_k^i \ll L_p^i$ peut être supprimé du nœud i. La règle de dominance permet de limiter le nombre de labels stockés sur chaque nœud à un plus petit sous-ensemble tout en maintenant l'optimalité. Afin de réduire le temps d'exécution de l'algorithme, le nombre maximal de labels stockés sur chaque nœud peut être limité (N_L). Cette limitation, ajoutée à la règle de dominance, peut réduire fortement le temps CPU, mais peut mener à des solutions sous-optimales. Néanmoins, d'après l'article de (Boudia *et al.,* 2007), la qualité des solutions trouvées à la fin de l'algorithme SPLIT n'est que très rarement impactée par la limitation du nombre de labels sur chaque nœud.

La procédure SPLIT est présentée dans l'Algorithme 2. Elle repose sur l'utilisation de plusieurs sous-procédures permettant la mise à jour du graphe auxiliaire et des labels :

- `InitGraph`$(S, \lambda, k, n, N, N_L)$ initialise le graphe auxiliaire avec le label initial ;
- `CompareTransportOperation`(λ, i, j), cette fonction retourne 1 si les opérations de transport $\lambda(i+1)$ et $\lambda(j)$ correspondent à la même opération de transfert, *i.e.*, si $\lambda^-(i+1) = \lambda^-(j)$ et $\lambda^+(i+1) = \lambda^+(j)$;
- `PropagateLabel`$(\lambda(j), flow, n\_vehicle, L_k^i)$ retourne le label généré à partir du label L_k^i, de l'opération de transport $\lambda(j)$, du flot et du véhicule $T_{n\_vehicle}$ affecté;
- `TestInsertLabel`(P, j, S)compare le label P aux labels sur le nœud j. Si le label P domine au moins un label, la procédure retourne 1 et le ou les labels sont supprimés, sinon la procédure retourne 0;
- `InsertLabel`(P, j, S, N_L) insert le label P sur le nœud j par ordre croissant sur la valeur maximale des dates (départ et arrivé) des véhicules. Le nombre de labels sur le nœud j ne peut excéder N_L;
- `Extract_trips()` permet l'extraction du meilleur label et retourne les tournées des véhicules.

Algortihme 2 : SPLIT

```
1.  procedure Split
2.  input parameters
3.    λ: giant tour
4.    k: number of transport operation in the giant tour
5.    S: auxiliary graph
6.  output parameters
7.    T: solution
8.  global parameter
9.    n: number of activities
10.   N: number of vehicles
11.   N_L: maximum number of labels on each node
12.  begin
13.  | call InitGraph(S, λ, k, n, N, N_L)
14.  | i:= 0
15.  | while (i < k) do
16.  | | j := i + 1; flow := 0
17.  | | while (CompareTransportOperation(λ, i + 1, j) = 1) do
18.  | | | flow := flow + λ^φ(j)
19.  | | | for k = 1 to NB_i do              //for each label on node i
20.  | | | | n_vehicle := 1
21.  | | | | while ((n_vehicle ≤ N) and (flow ≤ C_n_vehicle)) do
22.  | | | | | P := call PropagateLabel(λ(j), flow, n_vehicle, L_k^i)
23.  | | | | | if (NB_j = 0) then
24.  | | | | | | call InsertLabel(P, j, S, N_L)
25.  | | | | | else insert := call TestInsertLabel(P, j, S)
26.  | | | | | | if (insert = 1) then call InsertLabel(P, j, S, N_L) endif
27.  | | | | | endif
28.  | | | | | n_vehicle := n_vehicle + 1
29.  | | | | endwhile
30.  | | | endfor
31.  | | | j := j + 1
32.  | | endwhile
33.  | | i := i + 1
34.  | endwhile
35.  | T := call Extract_trips ()          //save the best solution
36.  end
```

L'algorithme SPLIT (Algorithme 2) débute par une procédure d'initialisation permettant d'initialiser le graphe auxiliaire (ligne 13). La boucle principale de l'algorithme (de la ligne 15 à 34) permet de parcourir les opérations de transport définies par le tour géant λ. La seconde boucle de la ligne 17 à 32 permet l'évaluation du sous-graphe $(\lambda(i+1), \ldots, \lambda(j))$ avec la propagation des labels sur le nœud i (boucle for des lignes 19 à 30). La troisième boucle (de la ligne 21 à 29) parcourt l'ensemble des véhicules pouvant être affectés à l'opération de transport correspondant au sous-graphe $(\lambda(i+1), \ldots, \lambda(j))$, pour générer les nouveaux labels (line 22). Les labels ainsi générés sont insérés ou non sur le nœud j, avec la procédure `InsertLabel`. Le meilleur label à la fin de l'algorithme est utilisé pour extraire les tournées permettant d'obtenir la meilleure solution, celle minimisant le makespan (line 35).

Exemple détaillé de la procédure SPLIT

Cet exemple s'appuie sur le tour géant introduit précédemment, tel que $\lambda = (0,3,*,2)(0,3,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(0,2,*,2)(3,1,*,2)(3,2,*,2)(2,1,*,2)$ $(2,4,*,2)(2,4,*,2)(2,4,*,2)(1,5,*,1)(1,6,*,2)(1,6,*,1)(4,7,*,2)(4,7,*,2)(4,7,*,1)$ $(4,6,*,1)(6,7,*,2)(6,7,*,2)(2,5,*,2)(5,7,*,2)(5,7,*,1)$. Dans cet exemple, chaque label contient six informations sur les véhicules (dates de départ, d'arrivée et position) et les huit informations sur les activités (dates de début au plus tôt). Le couple faisant référence au label père dans les labels n'est pas mentionné dans cet exemple. Les labels sont donc tous de la forme :

$$L_p^i = \left(\left(B_{pi}^1, B_{pi}^2, A_{pi}^1, A_{pi}^2, P_{pi}^1, P_{pi}^2\right)\left(S_{pi}^0, S_{pi}^1, S_{pi}^2, S_{pi}^3, S_{pi}^4, S_{pi}^5, S_{pi}^6, S_{pi}^7\right)\right)$$

La première étape du SPLIT consiste à propager le label initial $L_1^0 = \left((0,0,0,0,0,0)(0,0,0,0,0,0,0,0)\right)$ en 0 vers le nœud 1 en affectant la première opération de transport du tour géant $(0,3,*,2)$ soit au véhicule T_1, soit au véhicule T_2. Si l'opération de transport est affectée au véhicule T_1 alors le label L_1^1 est obtenu par mise jour de certaines informations à partir du label L_1^0, telle que :

$$B_{11}^1 = \max\left(S_{10}^0 + p_0;\ A_{10}^1 + t_{0,0}\right) = \max(0+0;\ 0+0) = 0$$
$$B_{11}^2 = \max(B_{10}^2;\ A_{10}^2) = \max(0;\ 0) = 0$$
$$A_{11}^1 = B_{11}^1 + t_{0.3} = 0 + 6 = 6$$
$$P_{11}^1 = 3$$
$$S_{11}^3 = \max(S_{10}^3;\ A_{11}^1) = \max(0;\ 6) = 6$$

On obtient alors le label $L_1^1 = \left((0,0,6,0,3,0)(0,0,0,6,0,0,0,0)\right)$. Le même raisonnement se transpose en affectant l'opération de transport $(0,3,*,2)$ au véhicule T_2. On obtient alors le label $L_2^1 = \left((0,0,0,6,0,3)(0,0,0,6,0,0,0,0)\right)$. Ces deux labels sont incomparables puisque la position des véhicules est différente d'un label à l'autre.

La deuxième étape du SPLIT consiste à propager le label initial $L_1^0 = \left((0,0,0,0,0,0)(0,0,0,0,0,0,0,0)\right)$ en 0 vers le nœud 2. Cette propagation correspond à l'affectation de l'opération de transport $(0,3,*,4)$ obtenue par agrégation des opérations $(0,3,*,2)$ et $(0,3,*,2)$. Cette opération de transport n'est réalisable que par le véhicule T_1 puisque $C_1 = 4$. Le véhicule T_2, étant de capacité $C_2 = 2$, ne peut effectuer cette opération de transport avec quatre ressources. Un seul label est alors généré par propagation. On obtient alors le label $L_1^2 = \left((0,0,6,0,3,0)(0,0,0,6,0,0,0,0)\right)$. Ce label est strictement identique au label L_1^1, puisque la quantité de ressource des opérations de transport n'est pas prise en compte dans la création d'un label. Ces deux premières étapes sont illustrées sur la Figure 4-25 avec une représentation des trois premiers nœuds du graphe auxiliaire ainsi que les trois premiers labels générés.

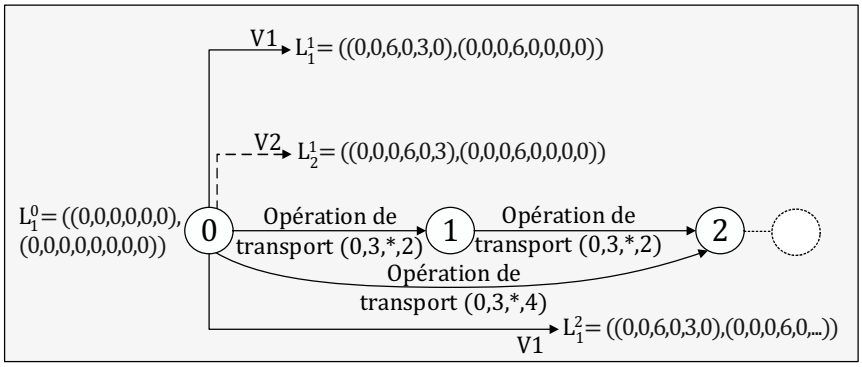


Figure 4-25. Illustration des deux premières étapes de la procédure SPLIT.

La propagation du label L_1^0 ne peut être effectuée jusqu'au nœud 3, les opérations de transport étant différentes $\big((0,3,*,2)(0,3,*,2)(0,2,*,2)\big)$ l'agrégation est impossible. La troisième étape du SPLIT consiste donc à propager les labels présents sur le nœud 1 vers le nœud 2. Cette étape permet de générer quatre labels à partir des deux labels stockés sur le nœud 1 (L_1^1 et L_2^1). Le premier label généré L_2^2 est obtenu par propagation du label L_1^1 en utilisant le véhicule T_1 pour réaliser l'opération de transport $(0,3,*,2)$. Le label L_2^2 est obtenu par mise à jour de certaines informations à partir du label L_1^1, telle que :

$$B_{22}^1 = \max\big(S_{11}^0 + p_0;\ A_{11}^1 + t_{3,0}\big) = \max(0+0;\ 6+6) = 12$$
$$B_{22}^2 = \max(B_{11}^2;\ A_{11}^2) = \max(0;\ 0) = 0$$
$$A_{22}^1 = B_{11}^1 + t_{0.3} = 12 + 6 = 18$$
$$P_{22}^1 = 3$$
$$S_{22}^3 = \max(S_{11}^3;\ A_{22}^1) = \max(6;\ 18) = 18$$

On obtient alors le label $L_2^2 = \big((12,0,18,0,3,0)(0,0,0,18,0,0,0,0)\big)$. En comparant ce label à celui déjà stocké sur le nœud 2 (L_1^2), on peut noter que la position des deux véhicules est identique $P_{22}^* = P_{21}^*$, et que $B_{22}^1 > B_{21}^1$ et $A_{22}^1 > A_{21}^1$. Le label L_2^2 est dominé par le label L_1^2 ($L_1^2 \gg L_2^2$). Le label L_2^2 n'est par conséquent pas stocké sur le nœud 2. Les solutions partielles associées à ces deux labels sont représentées sur la Figure 4-26. Les labels supprimés grâce à la règle de dominance sont soulignés sur les figures.

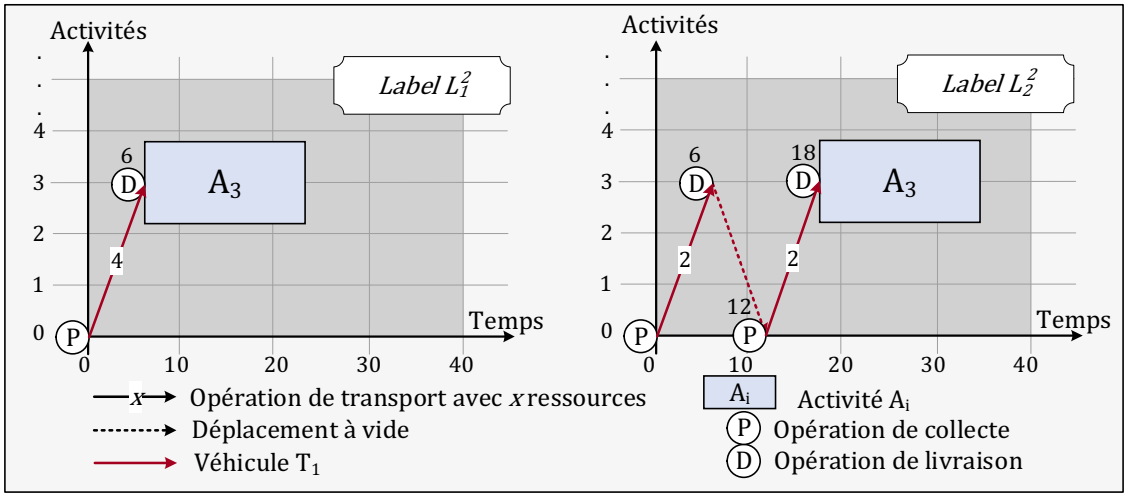


Figure 4-26. Solutions partielles associées aux labelsL_1^2 et L_2^2 avec $L_1^2 \gg L_2^2$.

Dans les deux solutions partielles représentées sur la Figure 4-26, seul le véhicule T_1 est utilisé. Sur la première solution le véhicule collecte quatre unités de ressource au niveau de l'activité A_0 pour les livrer à l'activité A_3. L'activité A_3 nécessitant quatre unités de ressource peut débuter à la date 6. Sur la deuxième solution les quatre ressources sont acheminées en deux allers-retours avec deux ressources à chaque fois. L'activité A_3 ne peut donc débuter qu'à la date $6 + 6 + 6 = 18$.

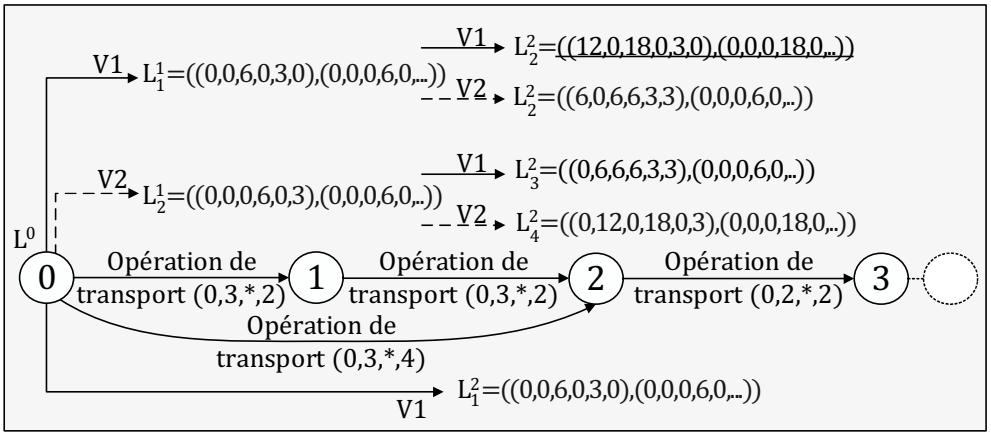


Figure 4-27. Illustration des trois premières étapes de la procédure SPLIT.

Une fois la troisième étape du SPLIT terminée, quatre labels sont stockés sur le nœud 2, ces labels sont donnés sur la Figure 4-27. La quatrième étape consiste à propager ces labels sur le nœud 3. Parmi les huit labels générés lors de cette cinquième étape, seul cinq sont stockés sur le nœud 3 grâce à la règle de dominance. La Figure 4-28 donne le détail des cinq premières étapes de la procédure SPLIT, avec tous les labels générés dont les labels stockés sur les nœuds et les labels supprimés par dominance.

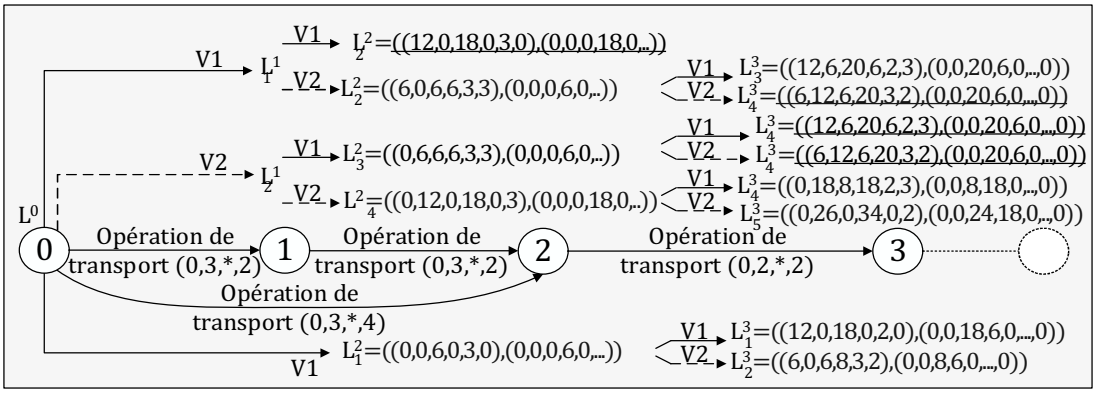


Figure 4-28. Illustration des quatre premières étapes de la procédure SPLIT.

Á la fin de toutes les étapes du SPLIT, le meilleur label est égal à :

$$L_{best} = \big((130,140,146,149,7,7)(0,34,30,6,54,128,108,149)\big)$$

La tournée du véhicule T_1 est donnée par les neuf opérations de transport suivantes :

$$(0,3,1,4)(0,2,1,4)(3,2,1,2)(2,1,1,2)(1,6,1,3)(4,7,1,4)(4,7,1,1)(2,5,1,2)(5,7,1,3)$$

La tournée du véhicule T_2 est donnée par les 10 opérations de transport suivantes :

$$(0,2,2,2)(0,2,2,2)(3,1,2,2)(2,4,2,2)(2,4,2,2)(2,4,2,2)(1,5,2,1)(4,6,2,1)(6,7,2,2)(6,7,2,2)$$

Les 24 opérations de transport initialement données par le tour géant permettent la construction de deux tournées avec au total 19 opérations de transport, on peut en déduire que 10 opérations de transport ont été agrégées pour former cinq nouvelles opérations de transport. Le makespan associé au meilleur label L_{best} est égal à $C_{max} = 149$.

4.4.8 Evaluation d'une solution (cinquième étape de l'évaluation)

Les tournées permettent de définir, les opérations de transport entre les différentes activités et, les disjonctions entre les opérations de transport. Ces informations permettent de construire le graphe disjonctif orienté modélisant une solution du RCPSPR. Une solution est entièrement définie lorsque toutes les dates (activités et opérations de transport) sont calculées. Le calcul de ces dates peut être effectué par évaluation du graphe disjonctif orienté grâce à un algorithme de plus long chemin.

Exemple détaillé de l'évaluation des tournées obtenues grâce au SPLIT

Deux tournées ont été constituées à la fin du SPLIT, la première étant composée de neuf opérations de transport et la deuxième étant composée de 10 opérations de transport. Ces opérations de transport sont modélisées par des rectangles sur la Figure 4-29. Dans la Figure 4-29, afin de faciliter sa lecture, seules les disjonctions correspondant à la tournée du véhicule T_2 apparaissent.

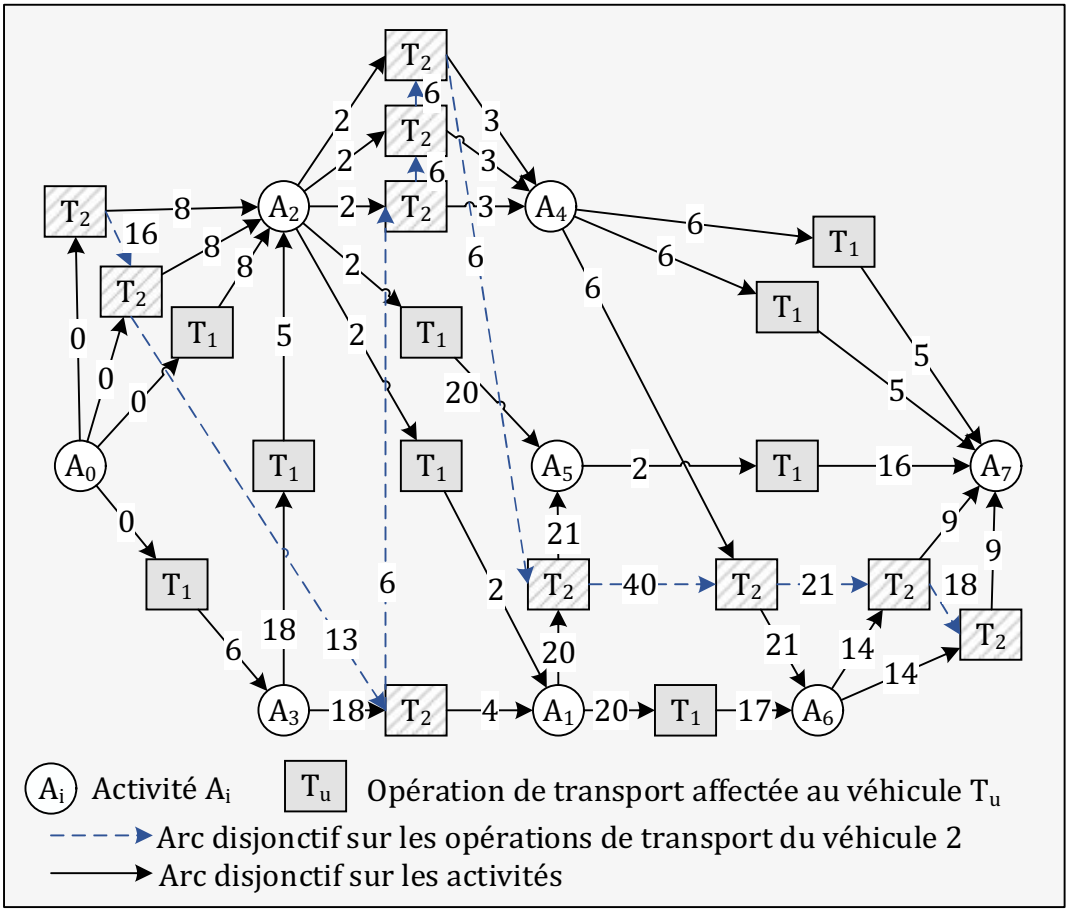


Figure 4-29. Graphe disjonctif avec les disjonctions pour le véhicule T_2.

Une fois les tournées modélisées dans le graphe disjonctif orienté, celui-ci peut être évalué (Figure 4-30). On peut remarquer que cette évaluation de la solution est en accord avec le label obtenu à la fin du SPLIT correspondant à cette même solution, $L_{best} = \big((130,140,146,149,7,7)(0,34,30,6,54,128,108,149)\big)$. On retrouve après évaluation toutes les dates de début au plus tôt des activités. Les deux véhicules terminent leurs tournées sur l'activité A_7 avec pour le véhicule T_1 une dernière opération de collecte à la date $B_{5,7,1,3} = 130$ avec l'opération de livraison à la date $A_{5,7,1,3} = B_{5,7,1,3} + t_{5,7} = 130 + 16 = 146$ et pour le véhicule T_2 une dernière opération de collecte à la date $B_{6,7,2,2} = 140$ avec l'opération de livraison à la date $A_{6,7,2,2} = B_{6,7,2,2} + t_{6,7} = 140 + 9 = 149$. Le projet est donc terminé à la date $C_{max} = 149$.

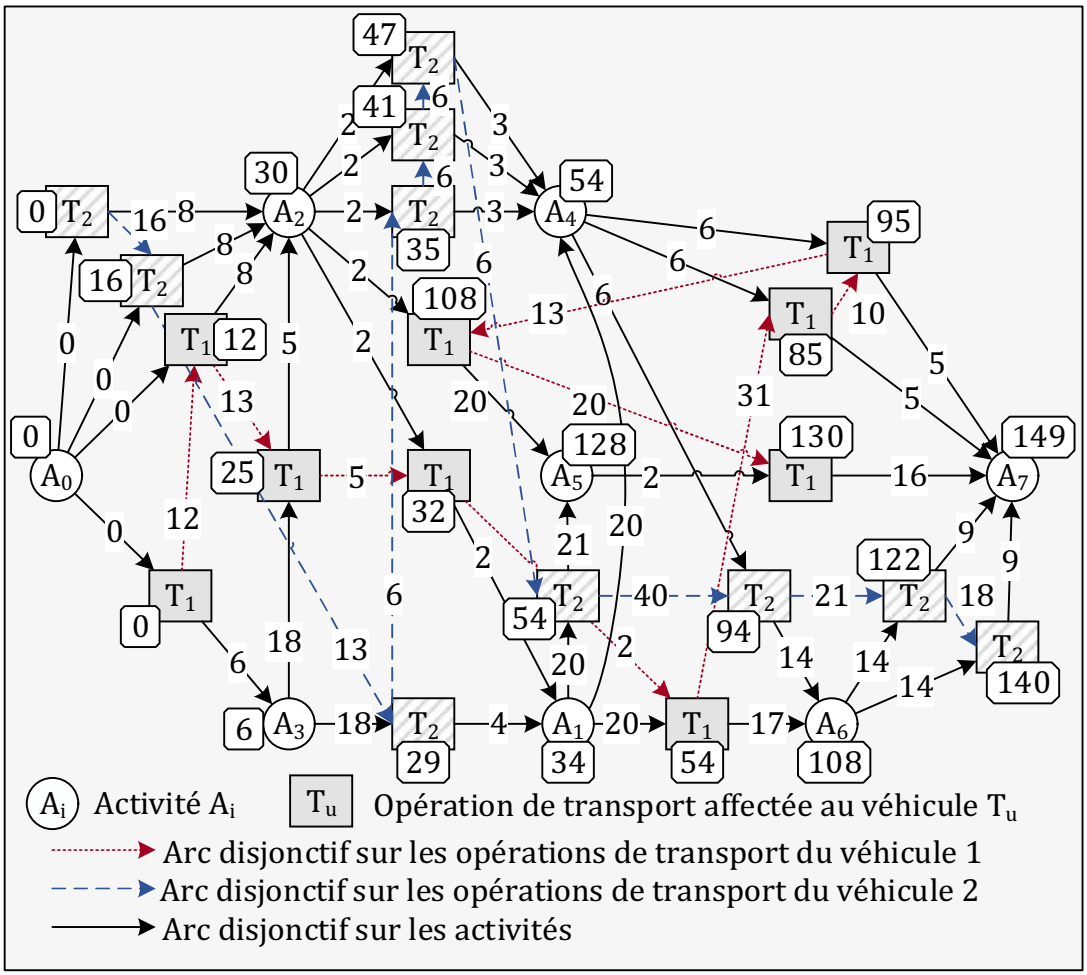


Figure 4-30. Graphe disjonctif évalué.

4.4.9 Recherche locale sur le chemin critique

Une solution initiale du RCPSPR est obtenue par évaluation d'un vecteur d'activités w. Des voisins peuvent être générés grâce à l'opérateur $swap\,(u,v)$, permettant d'échanger deux activités dans le vecteur w tout en conservant un ordre respectant les contraintes de précédence.

Pour améliorer une solution RCPSPR, un algorithme de recherche local efficace peut être défini en fonction du chemin critique (la Figure 4-31 met en évidence le plus court chemin dans le graphe disjonctif évalué). Dans ce problème, la notion de blocs déjà introduite dans les problèmes d'ordonnancement par (Van Laarhoven *et al.*, 1992; Matsuo *et al.*, 1988; Dell'amico et Trubian, 1993; Nowicki et Smutnicki, 1996) peut être étendue avec une classification mettant en évidence quatre types de blocs présents sur le chemin critique:

- Bloc de type 1: une opération de transport et une activité;
- Bloc de type 2: deux opérations de transport;
- Bloc de type 3: une activité et une opération de transport;
- Bloc de type 4: deux activités.

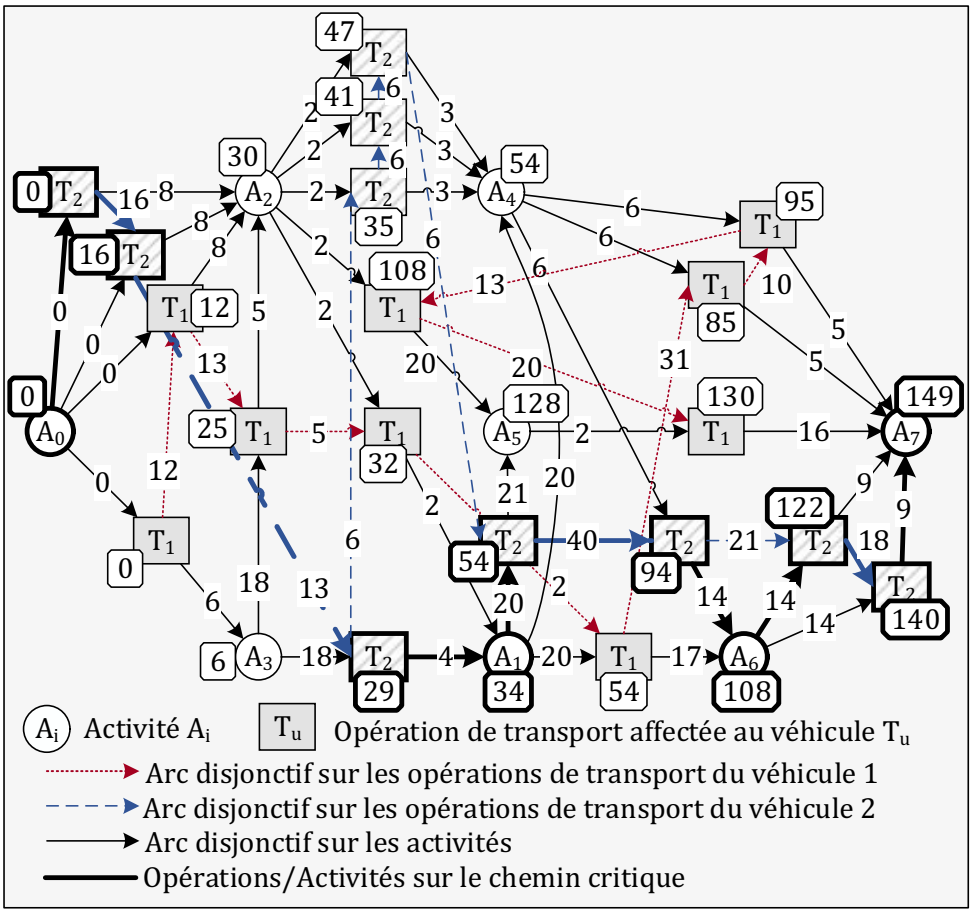


Figure 4-31. Graphe disjonctif évalué avec le chemin critique en gras.

La recherche locale remonte le chemin critique de la fin vers le début du graphe et, en fonction du type de bloc rencontré dans le chemin critique, des modifications sur le vecteur d'activités w sont envisagées. Si une modification du vecteur permet l'obtention d'une solution après évaluation, l'exploration du chemin critique repart du nœud final.

Les blocs de type 1 modélisent une situation où la date de début d'une activité A_j dépend d'une opération de transport $O_{i,j,*,*}$. Pour espérer obtenir une solution dans

laquelle cette situation ne se produit pas, l'opération de transport peut être supprimée. Cela peut être obtenu grâce à un décalage de l'activité A_j vers la gauche dans le vecteur d'activités w et ce jusqu'à ce que l'activité A_j soit à gauche de l'activité A_i. Aucun flot ne peut alors être généré de l'activité A_i à l'activité A_j.

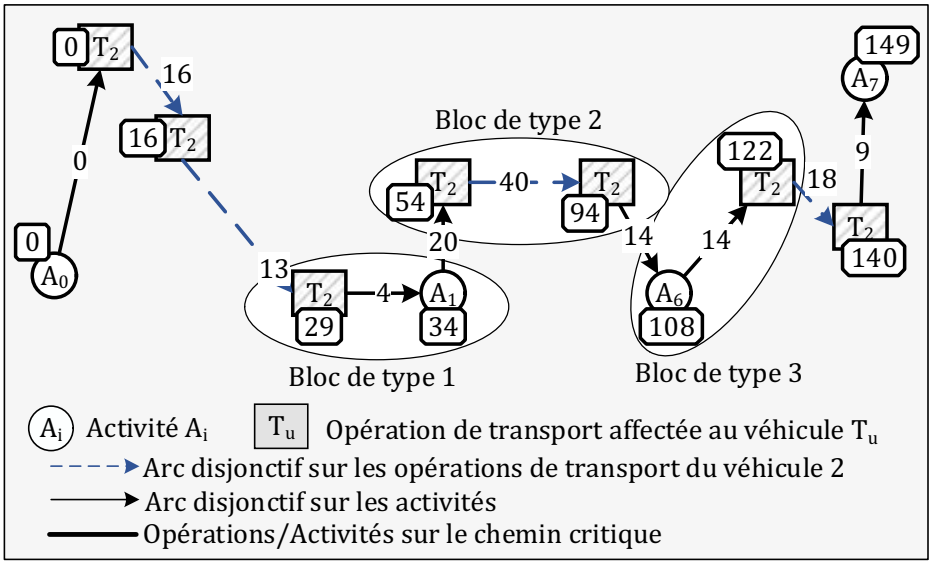


Figure 4-32. Illustration de trois types de bloc sur le chemin critique en gras.

Les blocs de type 2 correspondent à deux opérations de transport consécutives $O_{i,j,*,*}$. et $O_{k,p,*,*}$. avec un déplacement à vide. Dans cette situation, la deuxième opération de transport doit attendre la fin de la première avec le déplacement à vide du véhicule. En déplaçant l'activité A_k à gauche de l'activité A_i dans le vecteur w toutes les opérations de transport d'origine A_k ont lieu avant toutes les opérations de transport d'origine A_i. Cette situation ne peut plus se produire entre ces activités.

Les blocs de type 3 modélisent une situation où la date de début d'une opération de transport $O_{i,j,*,*}$ dépend directement de la date de fin de l'activité A_i. Pour espérer obtenir une solution dans laquelle cette situation ne se produit pas, l'opération de transport peut être supprimée. Cela peut être obtenu grâce à un décalage de l'activité A_j vers la gauche jusqu'à ce qu'elle soit à gauche de l'activité A_i dans le vecteur d'activités w. Aucun flot ne peut alors être généré de l'activité A_i à l'activité A_j.

Les blocs de type 4 sont présents entre deux activités et modélisent le fait qu'une activité doit attendre la fin d'une autre activité. Cette situation se présente lorsque les deux activités sont liées par une contrainte de précédence. Pour tenter de supprimer cet arc du chemin critique, la première activité doit être planifiée plus tôt. Cela peut être réalisé en décalant la première activité vers la gauche dans le vecteur d'activités w.

4.5 Schéma général de résolution du RCPSPR : GRASP×ELS

Le schéma général de résolution est basé sur une métaheuristique de type GRASP×ELS permettant l'intégration des procédures présentées précédemment pour résoudre le problème du RCPSPR.

Le GRASP×ELS permet l'intégration des points-clés mentionnés précédemment :

- une méthode de génération d'un vecteur d'activités ainsi qu'un opérateur pour effectuer des mutations sur celui-ci ;
- une fonction d'évaluation d'un vecteur d'activités en solution du RCPSPR composée de cinq étapes et d'une recherche locale sur le flot ;
- une recherche locale basée sur le chemin critique dans le graphe disjonctif orienté et évalué associé à une solution du RCPSPR.
- une procédure de concaténation pour obtenir un vecteur d'activités à partir d'une solution du RCPSPR.

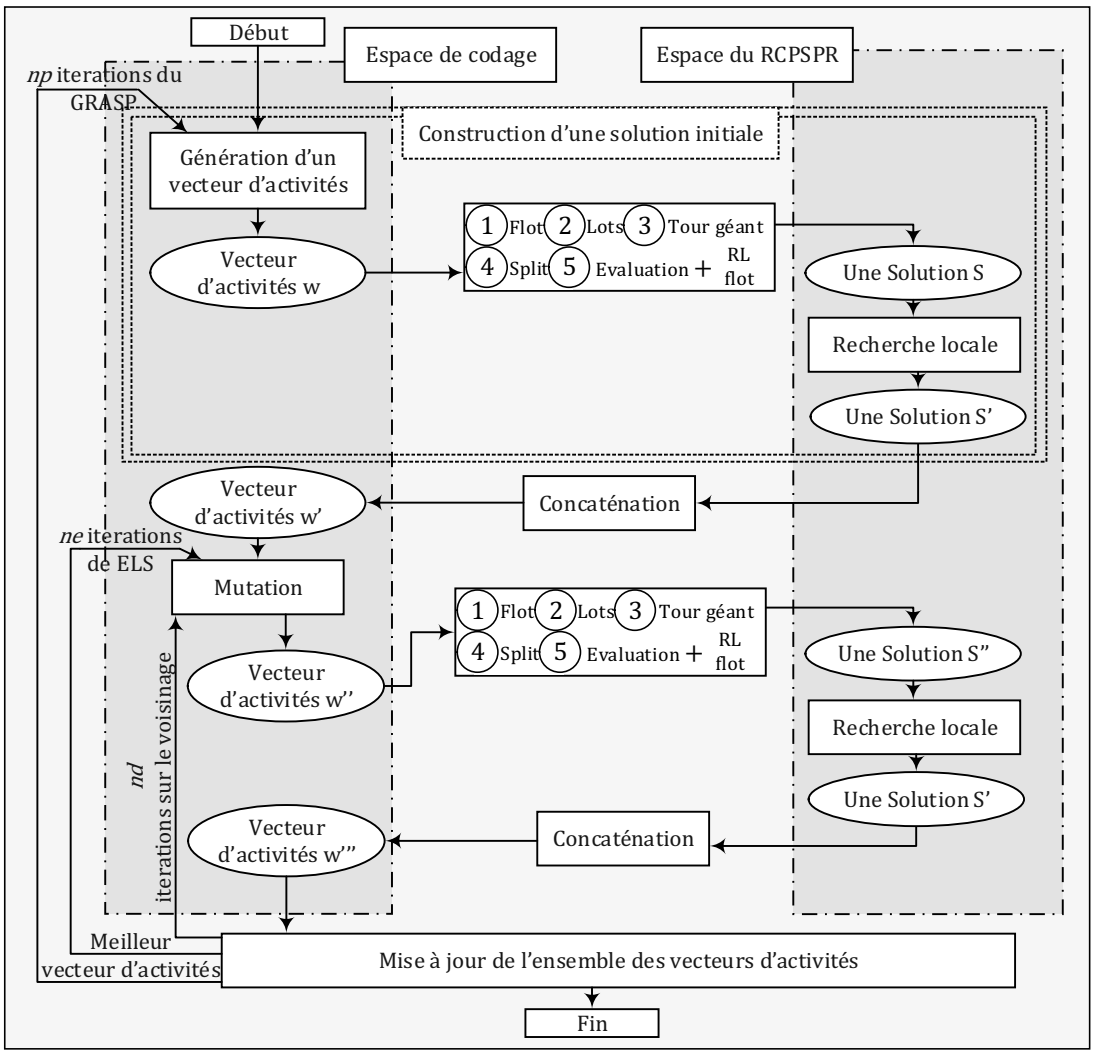


Figure 4-33. GRASP×ELS pour le RCPSPR.

Comme représenté sur la Figure 4-33, le GRASP×ELS est défini sur deux espaces de codage : l'ensemble des vecteurs d'activités et les solutions du RCPSPR. L'efficacité de la méthode repose sur différents opérateurs (fonction d'évaluation, recherche locale, concaténation et mutation) définis sur des espaces solutions spécifiques pour favoriser l'exploration de la métaheuristique.

4.6 Résultats numériques

Á notre connaissance, aucune instance traitant d'un problème similaire au RCPSPR n'est disponible. Par conséquent, les expérimentations numériques pour le problème du RCPSPR sont réalisées sur deux jeux d'instances présentés ci-dessous :

- Le premier jeu d'instances regroupe 18 instances de petite taille, *i.e.*, avec six activités non fictives, pour lesquelles une résolution optimale utilisant CPLEX est possible ;
- Le second jeu d'instances regroupe neuf instances de taille moyenne, *i.e.*, avec 30 activités non fictives, ces instances ne sont pas solvables par CPLEX.

Ces jeux d'instances sont disponibles sur la page web suivante :
http://fc.isima.fr/~vinot/Research/RCPSP&Routing.html

Les expérimentations numériques détaillées dans cette partie permettent de :

- prouver (en utilisant les instances de petite taille) que la résolution séquentielle des deux sous-problèmes (le problème de flot, puis le routing / ordonnancement) conduit à des solutions sous-optimales pour le problème du RCPSPR ;
- évaluer la performance de l'approche de résolution avec les solutions fournies par GRASP×ELS et les solutions optimales fournies par CPLEX (en utilisant les instances de petite taille) ;
- évaluer la convergence GRASP×ELS sur les instances de taille moyenne pour des travaux futurs.

Pour chaque instance, cinq réplications du GRASP×ELS ont été effectuées. Toutes les expérimentations sont réalisées avec le même ensemble de paramètres introduit dans le Tableau 4-6. Les notations suivantes sont utilisées dans les tableaux résultats :

- Avg valeur moyenne
- LB borne inférieure
- n_{tot} nombre minimum d'opérations à ordonnancer
- BFS meilleure solution trouvée
- TT temps CPU total en secondes lié à la réplication dans laquelle la BFS est obtenue
- T* temps CPU en secondes pour obtenir la BFS
- Nb. Opt Nombre d'instances où la borne inférieure est atteinte (au moins une fois parmi les cinq réplications). Dans ce cas, la solution obtenue est une solution optimale du RCPSPR.

Tableau 4-6
Paramétrage du GRASP×ELS pour les instances

Paramètre	Définition	Valeur
np	Nombre d'itérations pour le GRAPS	100
ne	Nombre d'itérations pour le ELS	25
nd	Nombre de voisins générés	10
lr	Nombre d'itérations pour la recherche locale sur le chemin critique	10
lf	Nombre d'itérations pour la recherche locale sur le flot	50
NBmax	Nombre maximal de labels par nœud	20

Toutes les expérimentations numériques sont réalisées sous un programme C en utilisant Visual Studio et Windows 7 comme système d'exploitation sur un Dell Optiflex9020 avec un processeur Intel Core i7-4770 CPU 3.4 GHz et 16 Gb de RAM. Le nombre de Mflop/s de cet ordinateur peut être estimé à 2671 Mflops (Dongarra, 2014).

4.6.1 Instances pour le RCPSPR

Les 18 instances de petite taille sont composées de six activités non fictives plus deux activités fictives modélisant l'activité de début du projet et l'activité de fin du projet. Ces deux activités fictives sont localisées au même endroit afin de modéliser un dépôt. Toutes les ressources sont initialement disponibles au dépôt et à l'issue du projet elles doivent être restituées au dépôt. Ces 18 instances peuvent être divisées en deux groupes de neuf instances en fonction de la localisation des activités réparties soit de façon uniforme, soit en deux clusters.

Tableau 4-7
Caractéristiques des instances de petite taille

Instance	n_{tot}	Localisation	Demandes	Quantité de ressource	Capacités des véhicules	Ratio
LMQV_U1	13	uniform	{4;10;3;3;8;4}	12	{12;12}	1.6
LMQV_U2	13	uniform	{4;10;3;3;8;4}	12	{12;12}	1.1
LMQV_U3	13	uniform	{4;10;3;3;8;4}	12	{12;12}	1.3
LMQV_U4	13	uniform	{2;8;7;8;8;5}	14	{6;5}	0.5
LMQV_U5	13	uniform	{2;8;7;8;8;5}	14	{6;5}	0.4
LMQV_U6	13	uniform	{2;8;7;8;8;5}	14	{6;5}	0.4
LMQV_U7	13	uniform	{5;3;8;2;3;1}	10	{4;2}	0.3
LMQV_U8	13	uniform	{5;3;8;2;3;1}	10	{4;2}	0.3
LMQV_U9	13	uniform	{5;3;8;2;3;1}	10	{4;2}	0.3
LMQV_C1	13	clusters	{7;7;5;7;5;4}	7	{7;7}	3.0
LMQV_C2	13	clusters	{7;7;5;7;5;4}	7	{7;7}	2.1
LMQV_C3	13	clusters	{7;7;5;7;5;4}	7	{7;7}	3.0
LMQV_C4	13	clusters	{7;1;2;6;2;6}	11	{6;5}	1.1
LMQV_C5	13	clusters	{7;1;2;6;2;6}	11	{6;5}	1.0
LMQV_C6	13	clusters	{7;1;2;6;2;6}	11	{6;5}	1.0
LMQV_C7	13	clusters	{4;10;4;6;3;4}	12	{4;2}	0.8
LMQV_C8	13	clusters	{4;10;4;6;3;4}	12	{4;2}	0.8
LMQV_C9	13	clusters	{4;10;4;6;3;4}	12	{4;2}	0.8

Les demandes en ressource des activités ainsi que la quantité de ressource disponible, les capacités des véhicules et le ratio sont détaillés dans le Tableau 4-7. Le nombre minimal d'opérations à ordonnancer n_{tot} est donné dans la deuxième colonne du tableau.

Ce nombre comprend les activités non fictives au nombre de $n = 6$ ainsi que le nombre minimum d'opérations de transport qui est égal à $n + 1$ (ordonnancement séquentiel des activités avec un véhicule de capacité suffisante). Par conséquent, le nombre minimal d'opérations à ordonnancer est égal à $n_{tot} = 2n + 1 = 13$.

Le ratio indiqué dans la dernière colonne des tableaux d'instances permet d'évaluer le ratio entre le temps moyen induit par la durée des activités et le temps moyen pour effectuer l'opération de transport. Un ratio plus grand que 1 pour une instance met en évidence une instance ou les temps de transport sont faibles par rapport à la durée des activités. Un ratio plus petit que 1 pour une instance met en évidence l'inverse, *i.e.*, les temps de transport sont grands par rapport à la durée des activités.

Les neuf instances de taille moyenne sont composées de 30 activités non fictives plus deux activités fictives. Ces neuf instances peuvent être divisées en trois groupes de trois instances en fonction de la localisation des activités réparties soit de façon uniforme, soit en deux clusters avec un dépôt extérieur aux clusters, soit en deux clusters avec un dépôt à l'intérieur d'un cluster.

Tableau 4-8
Caractéristiques des instances de taille moyenne

Instance	n_{tot}	Localisation	Demandes	Quantité de ressources disponible	Capacités des véhicules	Ratio
LMQV_J30_U1	61	uniform	[1;10]	13	{13;13}	0.7
LMQV_J30_U2	61	uniform	[1;10]	14	{8;6}	0.3
LMQV_J30_U3	61	uniform	[1;10]	13	{13;9}	1.0
LMQV_J30_C1	61	clusters	[1;10]	15	{15;15}	0.8
LMQV_J30_C2	61	clusters	[1;10]	11	{7;5}	0.3
LMQV_J30_C3	61	clusters	[1;10]	12	{12;9}	0.9
LMQV_J30_CC1	61	clusters	[1;10]	13	{13;13}	0.6
LMQV_J30_CC2	61	clusters	[1;10]	14	{8;6}	0.3
LMQV_J30_CC3	61	clusters	[1;10]	13	{13;8}	1.1

4.6.2 Résolution séquentielle du RCPSPR vs. résolution intégrée

Une première résolution séquentielle du RCPSPR peut être envisagée, en résolvant dans un premier temps le RCPSP puis, à partir d'un flot associé à cette solution, le RCPSPR peut être résolu. La résolution optimale du RCPSP peut être réalisée par CPLEX grâce à la formulation linéaire du flot présentée en début de chapitre. La résolution du RCPSPR à partir d'un flot optimal pour le RCPSP peut être réalisée par CPLEX grâce à la formulation linéaire du RCPSP présentée en début de chapitre en fixant les variables liées au flot. Ces deux étapes sont optimales.

Une deuxième résolution séquentielle du RCPSPR peut être envisagée, en résolvant dans un premier temps le RCPSP avec temps de transfert (RCPSPTT) puis, à partir d'un flot associé à cette solution, le RCPSPR peut être résolu. La résolution optimale du RCPSPTT peut être réalisée par CPLEX grâce à la formulation linéaire du flot présentée en début de chapitre à laquelle plusieurs contraintes sont ajoutées. On introduit une variable binaire :

- x_{ij} qui vaut 1 si l'activité A_i transfert des ressources l'activité A_j, *i.e.*, si $\varphi_{ij} > 0$.

Les contraintes C à ajouter au programme linéaire $P1$, (donné sur la Figure 4-13) sont explicitées dans la Figure 4-34.

$$C:\begin{cases} \forall (A_i, A_j) \in V^2 & \varphi_{ij} \geq x_{ij} & (1') \\ \forall (A_i, A_j) \in V^2 & \varphi_{ij} \leq min(b_i, b_j) x_{ij} & (2') \\ \forall (A_i, A_j) \in V^2 & S_i + p_i + t_{ij} \leq S_j + (1 - x_{ij})M & (3') \\ \forall (A_i, A_j) \in V^2 & x_{ij} \in \{0,1\} & (4') \end{cases}$$

Figure 4-34. Contraintes additionnelles pour le RCPSPTT

(1') et (2') expriment la relation entre le flot et la variable binaire x_{ij}, si $\varphi_{ij} > 0$ alors $x_{ij} = 1$ et si $\varphi_{ij} = 0$ alors $x_{ij} = 0$.

(3') dans le cas où $x_{ij} = 1$, cette contrainte se réécrit $S_i + p_i + t_{ij} \leq S_j$, ce qui signifie que l'activité A_j ne peut débuter que lorsque l'activité A_i est terminée et que la durée de transfert est respectée. Dans le cas où $x_{ij} = 0$, cette contrainte se réécrit $S_i + p_i + t_{ij} \leq S_j + M$, la valeur de M est suffisamment grande pour que cette inégalité soit toujours vérifié, et ce quelles que soient les valeurs des variables S_i et S_j ;

(4') la variable x_{ij} est binaire.

La résolution du RCPSPR à partir d'un flot optimal pour le RCPSPTT peut être réalisée par CPLEX grâce à la formulation linéaire du RCPSP présentée en début de chapitre en fixant les variables liées au flot. Ces deux étapes sont optimales.

Ces deux approches de résolution séquentielle sont comparées entre elles dans le Tableau 4-9. La dernière colonne nommée `Gap`, permet de calculer l'écart relatif entre les deux solutions du RCPSPR grâce à la formule : $Gap = \frac{RCSPR\ BFS(1) - RCSPR\ BFS(2)}{min(RCSPR\ BFS(1) - RCSPR\ BFS(2))} \times 100$. Un écart positif traduit l'obtention d'une meilleure solution par la résolution séquentielle avec le RCSPTT et un écart négatif traduit l'obtention d'une meilleure solution pour la résolution séquentielle avec le RCSP. Dans ce tableau, plusieurs remarques peuvent être faites :

- la résolution du RCPSP est moins coûteuse en temps de calcul que la résolution du RCPSPTT (0.02 seconde en moyenne contre 1 seconde en moyenne) ;
- la résolution séquentielle du RCPSPR à partir du RCPSP est moins coûteuse en temps de calcul que la résolution séquentielle du RCPSPR à partir du RCPSPTT (18.1 secondes en moyenne contre 29.2 secondes en moyenne) ;
- la résolution séquentielle du RCPSPR à partir du RCPSPTT permet l'obtention de meilleures solutions en moyenne avec un écart de 5.3%.

Les solutions données dans le Tableau 4-9 fournissent des bornes supérieures pour le problème du RCPSPR. Ces deux méthodes séquentielles sont comparées à la méthode intégrée développée dans ce chapitre avec une métaheuristique de type GRASP×ELS.

Tableau 4-9
Comparaison de deux méthodes de résolution séquentielle du RCPSPR

		CPLEX (résolution séquentielle à partir d'une solution du RCPSP)					CPLEX (résolution séquentielle à partir d'une solution du RCPSP avec temps de transfert (RCPSPTT))					
Instance	n_{tot}	RCPSP BFS	T (sec.)	RCPSPR BFS (1)	T' (sec.)	TT=T+T' (sec.)	RCPSPTT BFS	T (sec.)	RCPSPR BFS (2)	T' (sec.)	TT=T+T' (sec.)	Gap (%)
LMQV_U1	13	19	0.03	95	21.0	21.0	57	1.68	85	6.9	8.58	11.8
LMQV_U2	13	27	0.02	172	10.1	10.1	112	0.73	193	21.5	22.23	-12.2
LMQV_U3	13	38	0.03	227	15.0	15.1	148	0.90	265	31.8	32.7	-16.7
LMQV_U4	13	21	0.02	115	39.0	39.0	79	1.90	125	57.0	58.9	-8.7
LMQV_U5	13	38	0.02	229	43.8	43.8	157	1.76	254	43.2	44.96	-10.9
LMQV_U6	13	42	0.02	319	69.4	69.4	178	1.19	332	99.9	101.09	-4.1
LMQV_U7	13	8	0.00	111	11.7	11.7	48	0.67	117	13.9	14.57	-5.4
LMQV_U8	13	15	0.02	225	8.7	8.7	96	0.72	237	17.8	18.52	-5.3
LMQV_U9	13	16	0.00	258	13.3	13.3	114	0.34	277	30.5	30.84	-7.4
LMQV_C1	13	45	0.05	89	9.0	9.0	70	1.48	70	5.4	6.88	27.1
LMQV_C2	13	65	0.03	156	7.4	7.4	118	0.53	118	5.2	5.73	32.2
LMQV_C3	13	90	0.03	181	9.0	9.0	143	0.97	143	4.5	5.47	26.6
LMQV_C4	13	12	0.02	44	15.5	15.5	25	0.76	45	15.5	16.26	-2.3
LMQV_C5	13	32	0.02	93	20.8	20.8	50	0.64	97	17.9	18.54	-4.3
LMQV_C6	13	24	0.02	82	11.3	11.4	51	0.51	91	18.4	18.91	-11.0
LMQV_C7	13	18	0.03	60	6.3	6.3	33	1.28	46	27.9	29.18	30.4
LMQV_C8	13	35	0.05	135	7.3	7.3	66	1.37	99	25.5	26.87	36.4
LMQV_C9	13	36	0.02	138	7.7	7.7	68	0.61	115	64.3	64.91	20.0
Avg.			0.02		18.1	18.1		1.00		28.2	29.2	5.3

Le Tableau 4-10 compare la première méthode de résolution séquentielle avec CPLEX à la méthode intégrée avec le GRASP×ELS. Dans ce tableau, on peut noter que :

- toutes les solutions obtenues avec le GRASP×ELS sont meilleures que celles obtenues avec la résolution séquentielle. Par exemple, le GRASP×ELS fournit une solution avec un coût égal à 74 pour l'instance LMQV_U1, alors que la méthode séquentielle fournit une solution avec un coût égal à 85. Une amélioration de 22.1% peut être notée entre les coûts de ces deux solutions. En moyenne, le GRASP×ELS fournit des bornes supérieures meilleures de 18.6% ;
- l'approche séquentielle est plus efficace du point de vue du temps, avec un temps de résolution moyen égal à 18.1 secondes. Cette méthode est six fois plus rapide que la méthode intégrée avec 117 secondes.

Tableau 4-10
La première méthode de résolution séquentielle vs. la méthode intégrée

		CPLEX (résolution séquentielle à partir d'une solution du RCPSP)					GRASP×ELS (résolution intégrée)		
Instance	n_{tot}	RCPSP BFS	T (sec.)	RCPSPR BFS	T' (sec.)	TT=T+T' (sec.)	BFS	TT (sec.)	Gap (%)
LMQV_U1	13	19	0.03	95	21.0	21.0	74	117.0	-22.1
LMQV_U2	13	27	0.02	172	10.1	10.1	149	124.0	-13.4
LMQV_U3	13	38	0.03	227	15.0	15.1	188	106.5	-17.2
LMQV_U4	13	21	0.02	115	39.0	39.0	100	208.7	-13.0
LMQV_U5	13	38	0.02	229	43.8	43.8	204	186.4	-10.9
LMQV_U6	13	42	0.02	319	69.4	69.4	231	237.9	-27.6
LMQV_U7	13	8	0.00	111	11.7	11.7	101	92.0	-9.0
LMQV_U8	13	15	0.02	225	8.7	8.7	206	97.6	-8.4
LMQV_U9	13	16	0.00	258	13.3	13.3	233	111.5	-9.7
LMQV_C1	13	45	0.05	89	9.0	9.0	70	0.5	-21.3
LMQV_C2	13	65	0.03	156	7.4	7.4	118	1.0	-24.4
LMQV_C3	13	90	0.03	181	9.0	9.0	143	0.7	-21.0
LMQV_C4	13	12	0.02	44	15.5	15.5	35	106.2	-20.5
LMQV_C5	13	32	0.02	93	20.8	20.8	71	119.0	-23.7
LMQV_C6	13	24	0.02	82	11.3	11.4	72	106.8	-12.2
LMQV_C7	13	18	0.03	60	6.3	6.3	46	203.4	-23.3
LMQV_C8	13	35	0.05	135	7.3	7.3	96	140.1	-28.9
LMQV_C9	13	36	0.02	138	7.7	7.7	100	145.9	-27.5
Avg.			0.02		18.1	18.1		117.0	-18.6

Le Tableau 4-11 compare la deuxième méthode de résolution séquentielle avec CPLEX à la méthode intégrée avec le GRASP×ELS. Dans ce tableau, on peut noter que :

- toutes les solutions obtenues avec le GRASP×ELS sont meilleures que celles obtenues avec la résolution séquentielle. Par exemple, le GRASP×ELS fournit une solution avec un coût égal à 74 pour l'instance LMQV_U1, alors que la méthode séquentielle fournit une solution avec un coût égal à 95. Une amélioration de 12.9% peut être notée entre les coûts de ces deux solutions. En moyenne, le GRASP×ELS fournit des bornes supérieures meilleures de 14.6% ;
- l'approche séquentielle est plus efficace du point de vue du temps, avec un temps de résolution moyen égal à 29.2 secondes. Cette méthode est quatre fois plus rapide que la méthode intégrée avec 117 secondes.

Tableau 4-11
La deuxième méthode de résolution séquentielle vs. la méthode intégrée

		CPLEX (résolution séquentielle à partir d'une solution du RCPSPTT)					GRASP×ELS (résolution intégrée		
Instance	n_{tot}	RCPSPTT BFS	T (sec.)	RCPSPR BFS	T′ (sec.)	TT=T+T′ (sec.)	BFS	TT (sec.)	Gap (%)
LMQV_U1	13	57	1.68	85	6.9	8.58	74	117.0	-12.9
LMQV_U2	13	112	0.73	193	21.5	22.23	149	124.0	-22.8
LMQV_U3	13	148	0.90	265	31.8	32.7	188	106.5	-29.1
LMQV_U4	13	79	1.90	125	57.0	58.9	100	208.7	-20.0
LMQV_U5	13	157	1.76	254	43.2	44.96	204	186.4	-19.7
LMQV_U6	13	178	1.19	332	99.9	101.09	231	237.9	-30.4
LMQV_U7	13	48	0.67	117	13.9	14.57	101	92.0	-13.7
LMQV_U8	13	96	0.72	237	17.8	18.52	206	97.6	-13.1
LMQV_U9	13	114	0.34	277	30.5	30.84	233	111.5	-15.9
LMQV_C1	13	70	1.48	70	5.4	6.88	70	0.5	0.0
LMQV_C2	13	118	0.53	118	5.2	5.73	118	1.0	0.0
LMQV_C3	13	143	0.97	143	4.5	5.47	143	0.7	0.0
LMQV_C4	13	25	0.76	45	15.5	16.26	35	106.2	-22.2
LMQV_C5	13	50	0.64	97	17.9	18.54	71	119.0	-26.8
LMQV_C6	13	51	0.51	91	18.4	18.91	72	106.8	-20.9
LMQV_C7	13	33	1.28	46	27.9	29.18	46	203.4	0.0
LMQV_C8	13	66	1.37	99	25.5	26.87	96	140.1	-3.0
LMQV_C9	13	68	0.61	115	64.3	64.91	100	145.9	-13.0
Avg.			1.00		28.2	29.2		117.0	-14.6

Pour conclure, ces résultats sont conformes aux hypothèses théoriques en prouvant que les approches séquentielles ne conduisent pas à des solutions de bonne qualité. Le flot associé à une solution de bonne qualité pour le RCPSP ou le RCPSPTT n'est pas nécessairement un flot de bonne qualité pour le RCPSPR.

4.6.3 Comparaison de méthode de résolution intégrée sur les petites instances

Le RCPSPR peut être résolu optimalement (de façon intégrée) grâce à la formulation linéaire du RCPSPR avec CPLEX. Pour les instances dites de petite taille, cette formulation contient un grand nombre de contraintes (nr) et un grand nombre de variables (nc), les valeurs numériques sont données dans le Tableau 4-12. Par exemple, pour l'instance LMQV_U1, la formulation linaire est composée de 54 186 variables et de 140 974 contraintes. Toutes les instances sont résolues optimalement, avec un temps moyen de résolution de l'ordre d'un jour. Pour l'instance LMQV_C9, la solution optimale est composée de 32 opérations (six activités et 26 opérations de transport). Cette solution est présentée dans la partie 4.2.2. De ce point de vue, les petites instances (en nombre d'activités) ne sont pas si petites compte tenu du nombre d'opérations de transport.

Tableau 4-12
Résolution optimale avec CPLEX

Instance	n_{tot}	CPLEX (résolution intégrée)			
		nc	*nr*	Solution Optimale	TT (sec.)
LMQV_U1	13	54186	140974	74	18 792
LMQV_U2	13	54186	140974	144	17 892
LMQV_U3	13	54186	140974	188	23 976
LMQV_U4	13	53999	140584	74	>172 800
LMQV_U5	13	53999	140584	192	71 172
LMQV_U6	13	53999	140584	228	>172 800
LMQV_U7	13	13421	34831	97	14 580
LMQV_U8	13	13421	34831	197	18 864
LMQV_U9	13	13421	34831	218	56 988
LMQV_C1	13	84523	220446	70	4 104
LMQV_C2	13	84523	220446	118	5 364
LMQV_C3	13	84523	220446	143	5 724
LMQV_C4	13	21072	54979	33	1 512
LMQV_C5	13	21072	54979	68	1 224
LMQV_C6	13	21072	54979	72	1 296
LMQV_C7	13	27848	72442	44	54 576
LMQV_C8	13	27848	72442	90	>172 800
LMQV_C9	13	27848	72442	94	>172 800
Avg.		42508	110709		>86 400

Le Tableau 4-13 compare les solutions obtenues avec deux approches permettant la résolution intégrée du problème de RCPSPR.

Tableau 4-13
Comparaison de la résolution optimale avec CPLEX et avec le GRASP×ELS

			CPLEX (résolution intégrée)		GRASP×ELS (résolution intégrée)			
Instance	n_{tot}	LB	S. Opt.	TT (sec.)	BFS	T* (sec.)	TT (sec.)	Gap (%)
LMQV_U1	13	57	74	18 792	74	58.6	117.0	0.0
LMQV_U2	13	112	144	17 892	149	65.9	124.0	3.5
LMQV_U3	13	148	188	23 976	188	64.8	106.5	0.0
LMQV_U4	13	79	95	>172 800	100	131.2	208.7	5.3
LMQV_U5	13	157	192	71 172	204	140.4	186.4	6.3
LMQV_U6	13	178	228	>172 800	231	0.0	237.9	1.3
LMQV_U7	13	48	97	14 580	101	15.1	92.0	4.1
LMQV_U8	13	96	197	18 864	206	18.1	97.6	4.6
LMQV_U9	13	114	218	56 988	233	0.1	111.5	6.9
LMQV_C1	13	70	70	4 104	70	0.0	0.5	0.0
LMQV_C2	13	118	118	5 364	118	0.3	1.0	0.0
LMQV_C3	13	143	143	5 724	143	0.0	0.7	0.0
LMQV_C4	13	25	33	1 512	35	2.7	106.2	6.1
LMQV_C5	13	50	68	1 224	71	2.7	119.0	4.4
LMQV_C6	13	51	72	1 296	72	62.8	106.8	0.0
LMQV_C7	13	33	44	54 576	46	0.2	203.4	4.5
LMQV_C8	13	66	90	>172 800	96	91.9	140.1	6.7
LMQV_C9	13	68	94	>172 800	100	0.2	145.9	6.4
Avg.				>86 400		36.4	117.0	3.3
Nb Opt.			18/18		6/18			

Le temps d'exécution du GRASP×ELS est en moyenne de 117 secondes sur les 18 instances de petite taille. Les solutions fournies par le GRASP×ELS ont un écart relatif de 3.3% par rapport aux solutions optimales trouvées avec CPLEX. Cependant, pour l'instance LMQV_U1, le GRASP×ELS trouve la solution optimale en 58.6 secondes (au total 117 secondes) contre 18 792 secondes (plus de 5 heures) pour CPLEX.

On peut remarquer que les instances LMQV_C1, LMQV_C2 et LMQV_C3 sont résolues optimalement par le GRASP×ELS en moins d'une seconde. En observant avec attention ces trois instances, on peut noter quelques caractéristiques :

- en observant les quantités demandées par chaque client, on peut remarquer que deux activités ne peuvent jamais avoir lieu en parallèle, puisque $\nexists(A_i, A_j) \in V, b_i + b_j \leq B$. Toutes les activités doivent être ordonnancées séquentiellement.
- en observant la capacité des véhicules, par rapport à la quantité de ressource disponible, on note que $\forall T_u \in T, B = C_u$. Par conséquent, aucun aller-retour n'est nécessaire pour le transport de ressources entre deux activités.

Ces caractéristiques permettent de qualifier ces instances de « faciles ». En effet, en définissant l'ordre des activités, les tournées des véhicules sont immédiates. De plus, la minimisation du makespan dans ces instances se ramène à une minimisation du temps de transport. En effet, dans ces instances, le makespan peut s'exprimer comme la somme de la durée des activés et du temps de transport entre deux activités consécutives, sachant que la somme des durées des activés est constante. Une solution optimale de l'instance LMQV_C1 est illustrée sur la Figure 4-35. Sur cet exemple, on observe immédiatement que les activités sont ordonnancées séquentiellement dans le temps. De plus, seuls trois opérations de transport sont effectuées par le véhicule T_1. Á partir de la date 37, le rôle des véhicules peut être échangé sans que cela ait un impact sur la solution. En effet à cette date les deux véhicules sont disponibles sur l'activité A_4 et leurs capacités sont identiques.

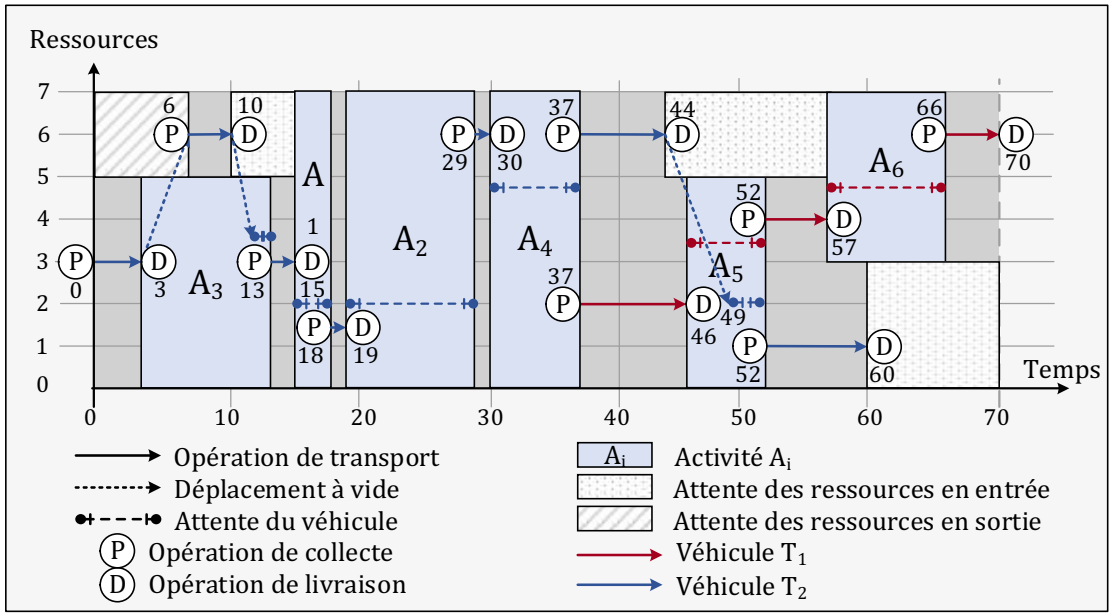


Figure 4-35. Solution optimale de l'instance LMQV_C1 du RCPSPR

4.6.4 Les impacts d'une variation du nombre de véhicules sur les petites instances

Dans cette section nous avons choisi d'étendre les petites instances proposées avec deux véhicules à plusieurs véhicules, pour évaluer l'impact du transport sur les solutions du RCPSPR. La borne inférieure utilisée dans cette section correspond à la solution optimale du RCPSPTT, *i.e.*, une solution du RCPSPR dans laquelle la flotte de véhicules est illimitée. Sur le Tableau 4-14, on peut remarquer qu'avec trois véhicules 6 solutions de même coût que la borne inférieure peuvent être trouvées (*i.e.*, 6 solutions optimales), avec quatre on monte à 15 solutions et avec cinq véhicules, on atteint 17 solutions optimales.

Tableau 4-14
Résolution du RCPSPR avec trois, quatre ou cinq véhicules avec le GRASP×ELS

			GRASP×ELS (3 véhicules)			GRASP×ELS (4 véhicules)			GRASP×ELS (5 véhicules)		
Instance	n_{tot}	LB	BFS	T* (s.)	TT (s.)	BFS	T* (s.)	TT (s.)	BFS	T* (s.)	TT (s.)
LMQV_U1	13	57	62	0.1	67.0	57	1.3	78.1	57	1.3	75.3
LMQV_U2	13	112	118	0.6	51.5	113	1.1	75.2	112	1.5	95.8
LMQV_U3	13	148	159	1.0	62.2	148	10.7	61.7	148	1.6	38.3
LMQV_U4	13	**79**	**79**	3.7	53.7	79	0.3	79.9	79	0.6	82.6
LMQV_U5	13	157	157	10.7	76.0	157	0.5	74.6	157	0.6	85.9
LMQV_U6	13	178	178	48.1	125.1	178	0.5	95.2	178	0.5	95.7
LMQV_U7	13	48	58	0.4	52.6	48	14.0	70.0	48	0.4	65.9
LMQV_U8	13	**96**	118	4.6	47.1	97	13.1	61.4	**97**	0.2	58.4
LMQV_U9	13	114	147	15.1	63.1	114	2.2	47.5	114	9.5	51.2
LMQV_C1	13	70	70	0.4	2.7	70	2.0	4.7	70	2.6	5.5
LMQV_C2	13	118	118	0.0	3.5	118	0.0	4.3	118	3.2	7.7
LMQV_C3	13	143	143	0.9	4.5	143	0.0	2.5	143	0.0	5.7
LMQV_C4	13	25	29	0.4	39.3	25	0.7	40.5	25	0.0	50.6
LMQV_C5	13	50	59	0.9	49.0	50	1.1	40.8	50	1.5	58.2
LMQV_C6	13	51	61	0.1	67.0	52	0.9	40.8	51	0.5	59.5
LMQV_C7	13	33	38	0.6	51.5	33	10.2	70.4	33	14.3	111.5
LMQV_C8	13	66	73	1.0	62.2	66	64.7	146.7	66	7.5	80.6
LMQV_C9	13	68	72	3.7	53.7	68	1.1	81.6	68	115.3	232.9
Avg.				6.2	49.8		7.9	55.8		11.1	69.2
Nb. Opt			6/18			15/18			17/18		

Après une étude approfondie de ces résultats, on peut affirmer qu'une flotte de cinq véhicules est suffisante pour atteindre toutes les bornes inférieures à l'exception de l'instance LMQV_U8, où la meilleure solution trouvée est égale à 97 alors que la borne inférieure vaut 96. Cependant, cette borne inférieure peut être atteinte en utilisant le flot donné par la solution optimale du RCPSPTT avec cinq véhicules. Toutes les bornes inférieures sont atteignables avec cinq véhicules.

Cette remarque est valable pour toutes les instances, *i.e.*, à partir d'un nombre suffisant de véhicules, le flot donné par la solution optimale du RCPSPTT permet d'obtenir une solution optimale pour le RCPSPR, puisque les deux problèmes deviennent équivalents. Néanmoins, pour un nombre aléatoire de véhicules, il a été montré précédemment qu'un bon flot pour le RCPSPTT n'est pas nécessairement un bon flot pour le RCPSPR.

4.6.5 Résolution des instances de taille moyenne

Pour les instances de taille moyenne (30 activités), la résolution intégrée avec CPLEX ne permet pas l'obtention de solution en plusieurs jours d'exécution, ce qui peut s'expliquer par le fait que les solutions du RCPSPR fournies par le GRASP×ELS sont constituées en moyenne de 50 opérations de transport. Les instances avec 30 activités sont donc composées de ~80 opérations à ordonnancer. Il n'est donc pas surprenant que la résolution optimale du problème du RCPSPR avec 30 activités ne soit pas possible dans des temps de calculs acceptables. Rappelons que l'obtention de solutions optimales, pour le RCPSP « classique », devient impossible (Valls *et al.,* 2005) pour des instances avec plus de 60 activités. Dans le Tableau 4-15, n_{tot} représente le nombre minimal d'opérations à ordonnancer.

Tableau 4-15
Résolution du RCPSPR avec le GRASP×ELS

Instance	n_{tot}	GRASP×ELS (résolution intégrée)		
		BFS	T* (sec.)	TT (sec.)
LMQV_J30_U1	61	175	37.5	213.2
LMQV_J30_U2	61	197	338.9	576.1
LMQV_J30_U3	61	107	27.7	244.6
LMQV_J30_C1	61	175	182.8	524.9
LMQV_J30_C2	61	177	51.2	172.4
LMQV_J30_C3	61	115	100.2	241.8
LMQV_J30_CC1	61	188	62.8	465.5
LMQV_J30_CC2	61	195	29.6	397.1
LMQV_J30_CC3	61	124	27.2	251.9
Avg.			95.3	343.0

La méthode proposée permet de résoudre les instances composées de 30 activités et de 2 véhicules dans des temps de calcul qui sont autour de 340 secondes.

4.6.6 Conclusion

Les résultats numériques montrent que cette nouvelle approche de résolution est efficace sur les instances de petite taille, puisqu'elle permet l'obtention de solutions avec un écart relatif de 3.3% par rapport aux solutions optimales, en des temps de calcul nettement inférieurs. Une analyse du problème du RCPSPR en faisant varier le nombre de véhicules permet de montrer qu'à partir d'un certain nombre de véhicules, le problème du RCPSPR peut se ramener au problème du RCPSPTT.

Pour les instances de taille moyenne, la métaheuristique fournit des premiers résultats dont la qualité n'est pour le moment pas mesurable puisque ces instances ne sont pas solvables par CPLEX dans des temps acceptables. Néanmoins, l'introduction de nouvelles méthodes de résolution (exactes ou heuristiques) pourrait être envisagée pour ces instances.

4.7 Conclusion du chapitre

Ce chapitre aborde la résolution d'un problème intégré avec un problème d'ordonnancement de type RCPSP (Resource Constrained Project Scheduling Problem) et un problème de tournées de véhicules de type VRPPD (Vehicle Routing Problem with Pickup and Delivery). Ce nouveau problème nommé RCPSPR (RCPSP with Routing) est une extension du RCPSP et du RCPSPTT (RCPSP with Transfer Time), dans lequel on intègre la gestion d'une flotte de véhicules nécessaires au transport des ressources entre les activités.

L'originalité et la difficulté de résolution du RCPSPR repose sur le fait que, contrairement aux problèmes d'ordonnancement de type flow shop ou job shop avec transport, dans lesquels les échanges de ressources sont connus (puisqu'on possède la notion de gamme définissant une suite ordonnée d'opérations), dans le RCPSPR il faut déterminer où (entre quelles activités) sont localisés les échanges de ressources. Cela signifie que, contrairement au job shop avec transport, les origines et destinations des déplacements à charge des véhicules ne sont pas définis par le problème, mais font partie de la solution à trouver.

La notion de flot, classiquement utilisée dans le RCPSP, offre une approche qui permet d'obtenir les échanges de ressources (*i.e.* un volume à transporter) entre deux activités. Selon les cas, une opération de transport assure le transport de tout ou d'une partie du flot. Connaître le flot n'est donc pas suffisant pour connaître les opérations de transport à planifier, mais cela permet de connaître les nœuds origine et destination des opérations de transport.

Nous proposons une modélisation mathématique du problème avec un programme linéaire en nombres entiers, permettant une résolution exacte des problèmes de petite taille, et une approche de résolution de type métaheuristique. L'efficacité de la métaheuristique, de type GRASP×ELS, repose sur plusieurs outils permettant la résolution efficace des sous-problèmes, avec par exemple, une procédure de type SPLIT pour définir les tournées des véhicules. L'originalité de la méthode SPLIT repose sur la définition de labels permettant à chaque instant de connaître l'état global du système, c'est-à-dire, l'état d'avancement des activités, ainsi que la position et la disponibilité de chaque véhicule.

La méthode proposée a prouvé son efficacité sur des petites instances grâce à une comparaison avec une résolution optimale et une résolution séquentielle. Des instances de grande taille, avec 30 activités, sont introduites et résolues par le GRASP×ELS. Il faut noter que la résolution d'instances composées de 30 activités peut paraître modeste (par rapport aux instances « classiques » du RCPSP qui peuvent contenir jusqu'à 120 activités), mais que cela représente près de 100 opérations à ordonnancer (activités et opérations de transport confondues).

La détermination d'une solution à partir d'un flot est une étape résolue de manière heuristique, mais qui fournit un très bon compromis, entre le temps d'exécution et la qualité de la solution associée au flot. Cette caractéristique est un des facteurs ayant

permis la conception d'un algorithme de type GRASP×ELS, lui-même efficace en temps d'exécution et donnant des solutions de bonne qualité.

Disposer d'une méthode « optimale » pour associer à un flot une solution (méthode qui est plus coûteuse en temps d'exécution), ne présente pas forcément d'intérêt pendant l'exécution d'une métaheuristique (la sous-optimalité de l'évaluation étant compensée par un plus grand nombre d'itérations), mais prend tout son sens dans une post-optimisation.

CHAPITRE 5

Résolution du RCPSPR avec une approche « flow-first, route and cluster-second »

Ce chapitre introduit une nouvelle méthode de résolution pour le problème d'ordonnancement de projet sous contraintes de ressources avec routage (Resource-Constrained Project Scheduling Problem with Routing - RCPSPR). Ce problème est décrit dans le chapitre 4. Ce problème intégré étend le problème d'ordonnancement de projet sous contraintes de ressources (RCPSP) en introduisant la gestion d'une flotte hétérogène de véhicules, permettant le transport des ressources. Dans ce nouveau problème, les véhicules permettent le transport des ressources d'une activité à une autre avec un seul type de ressource.

La méthode de résolution du RCPSPR proposée repose sur une approche « flow-first, route and cluster-second », inspirée de l'approche « route-first, cluster-second » proposée par Beasley en 1983 pour le VRP. La première phase consiste à générer un flot respectant les contraintes de ressource et les contraintes de précédence du problème. La deuxième phase consiste à définir à la fois les opérations de transport, un ordre sur les opérations de transport et l'affectation des opérations de transport à un véhicule. Cette deuxième phase repose sur un algorithme de plus court chemin à contraintes de ressources, appliqué sur le graphe obtenu à l'issue de la première phase. L'efficacité de cet algorithme repose sur trois points-clés : 1) la définition d'un label, 2) une règle de propagation et 3) une règle de dominance. L'algorithme de plus court chemin à contraintes de ressources proposé lors de la deuxième phase est optimal, par rapport au flot défini dans la première phase.

Les expérimentations numériques montrent que cette méthode de résolution est efficace. Les solutions obtenues sont comparées aux solutions optimales obtenues par programmation linéaire et aux solutions obtenues avec l'approche présentée dans le chapitre 4.

5.1 Présentation de l'approche de résolution proposée

L'approche de résolution du problème du RCPSPR proposée dans ce chapitre repose sur une méthode permettant de fournir la meilleure solution du RCPSPR à partir d'un flot. Le principe général de cette approche, nommée « flow-first, route and cluster-second », est présenté dans cette première partie.

5.1.1 Proposition d'une nouvelle fonction d'évaluation

L'approche de résolution proposée dans ce chapitre pour le problème du RCPSPR se base sur la définition d'une nouvelle fonction « d'évaluation » qui permet d'obtenir une solution du RCPSPR à partir d'un flot. Cette approche de résolution est basée sur une représentation indirecte de la solution, elle diffère de celle présentée dans le chapitre

précédent puisqu'elle repose sur une étape regroupant les étapes 2, 3, et 4 présentées sur la Figure 4-19. Un nouveau schéma de résolution peut être défini grâce à cette nouvelle fonction d'évaluation, celui-ci est représenté sur la Figure 5-1. L'originalité de cette deuxième approche repose donc sur l'étape 2 (Figure 5-1) avec un algorithme de plus court chemin à contraintes de ressources.

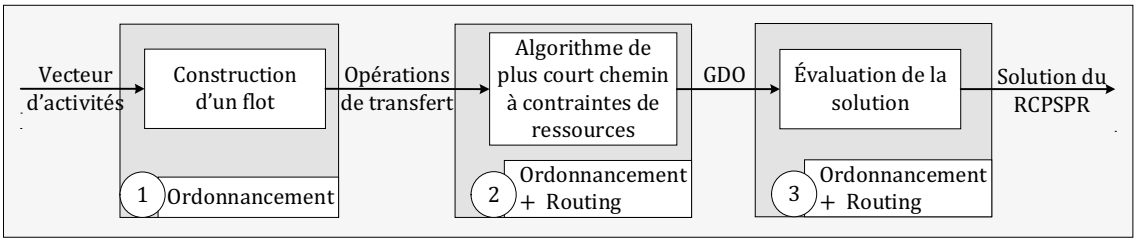


Figure 5-1. Construction d'une solution pour le RCPSPR

Cet algorithme de plus court chemin à contraintes de ressources est exact, dans le sens où il permet d'obtenir la meilleure solution du RCPSPR à partir d'un flot donné. Cette nouvelle fonction d'évaluation peut être comparée à celle présentée dans le chapitre précédent, et elle peut être intégrée au sein d'une métaheuristique comme la précédente. Cependant, afin de conserver le schéma de résolution du chapitre 4 reposant sur des fonctions efficaces en termes de temps, il est préférable d'utiliser cette nouvelle fonction comme une fonction de post-optimisation. Il peut également être envisagé de modifier l'algorithme de plus court chemin à contraintes de ressources pour le transformer en heuristique. Cette perte d'optimalité peut être compensée par une plus grande efficacité algorithmique et permettre une insertion plus efficace dans un schéma global d'optimisation, le plus grand nombre d'itérations effectuées compensant, pour tout ou partie, le caractère sous-optimal de l'algorithme.

5.1.2 Une approche de résolution « flow-first, route and cluster-second »

La fonction d'évaluation basée sur un algorithme de plus court chemin à contraintes de ressources s'inscrit dans une approche de résolution en deux phases. La première phase repose sur la définition d'un flot (solution du RCPSP) et la deuxième phase repose sur la définition des tournées des véhicules et l'ordonnancement des activités (solution du RCPRPR) (Figure 5-2).

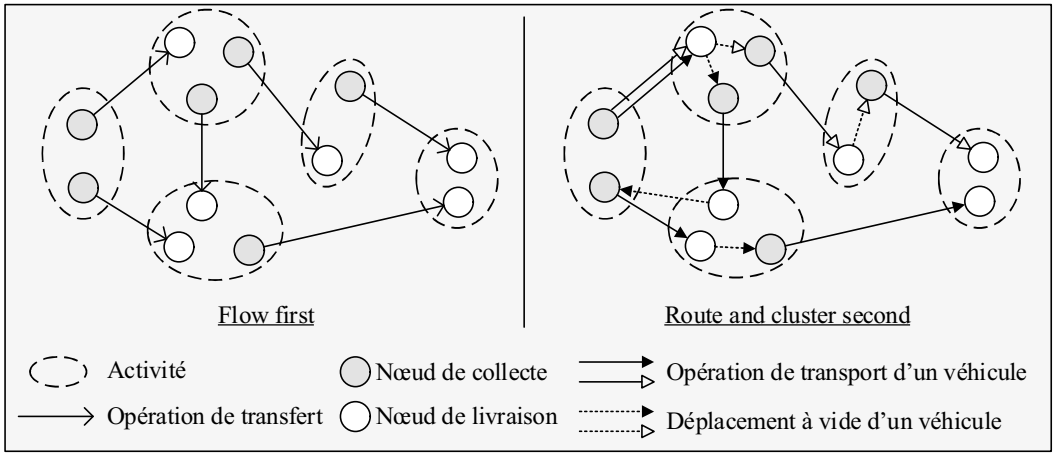


Figure 5-2. Illustration des deux phases de l'approche

Ces deux phases correspondent à deux espaces de solutions. La première phase permet de définir une solution du RCPSP grâce à un flot et la deuxième phase permet de définir une solution du RCPSPR. Un flot est une représentation indirecte de la solution du RCPSPR et la fonction d'évaluation proposée dans ce chapitre permet d'associer à un flot, une unique solution du RCPSPR (Figure 5-3). Cette fonction se différencie de la précédente, par le fait que la solution ainsi obtenue est la meilleure solution possible pour le flot : il s'agit de la solution « optimale par rapport au flot ».

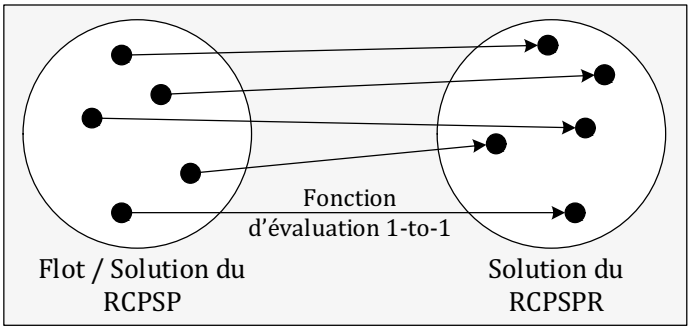


Figure 5-3. Illustration des deux espaces de recherche

Les deux phases de l'approche de résolution proposée dans ce chapitre sont illustrées avec l'exemple énoncé ci-après.

5.1.3 Introduction d'un exemple

Cet exemple de petite taille permet de détailler les deux phases de l'approche de résolution, notamment la deuxième phase avec l'introduction d'un graphe auxiliaire.

L'exemple introduit dans cette partie est composé de trois activités $n = 3$ et de deux activités fictives, modélisant à la fois l'activité de début/de fin du projet et le dépôt. Quatre unités de ressource sont disponibles $B = 4$, la durée des activités, leurs demandes en ressources ainsi que leurs successeurs sont indiqués dans le Tableau 5-1.

Tableau 5-1
Informations sur les activités

Activité A_i	Durée p_i	Demande b_i	Successeurs
A_0	0	/	1, 2, 3, 4
A_1	2	3	4
A_2	10	2	3, 4
A_3	5	2	4
A_4	0	/	/

Tableau 5-2
Matrice des distances

	A_0	A_1	A_2	A_3	A_4
A_0	0	2	2	3	0
A_1	2	0	2	3	2
A_2	2	2	0	5	2
A_3	3	3	5	0	3
A_4	0	2	2	3	0

Dans cet exemple, deux véhicules sont disponibles, $N = 2$, le véhicule T_1 est de capacité $C_1 = 3$ et le véhicule T_2 est de capacité $C_2 = 2$. Ces deux véhicules ont une vitesse constante égale à 1. Les temps de transport entre deux activités sont directement déduits de la matrice des distances entre les activités détaillée sur le Tableau 5-2.

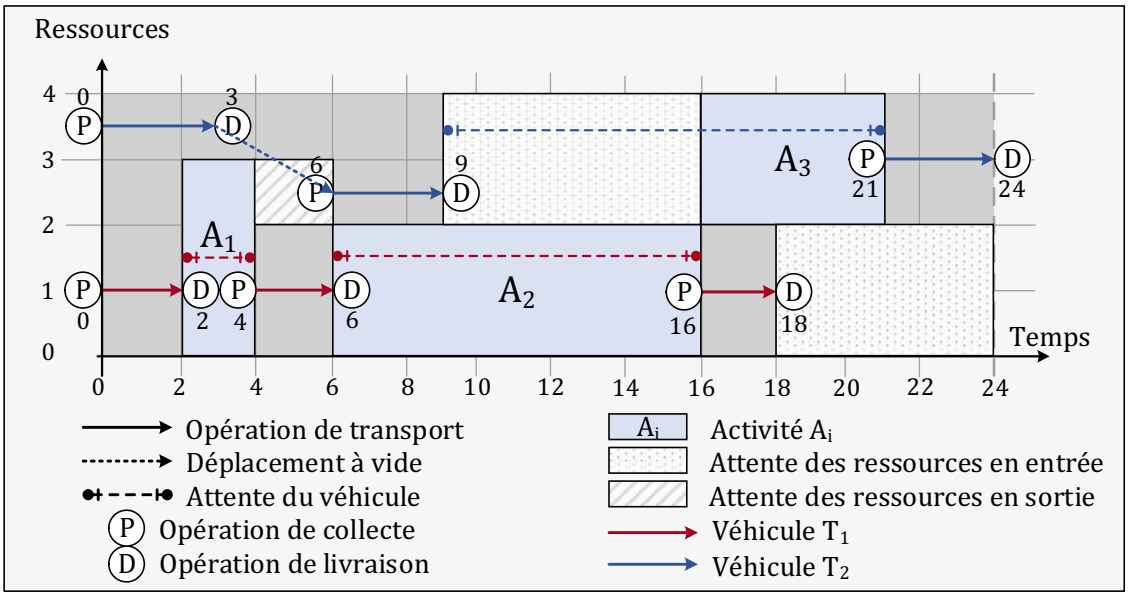


Figure 5-4. Une solution optimale de l'exemple avec trois activités et deux véhicules

Une solution optimale obtenue avec CPLEX sur la formulation linéaire présentée dans le chapitre précédent est illustrée sur la Figure 5-4. Cette solution a un makespan, $C_{max} = 24$. Chacun des véhicules effectue une tournée composée de trois opérations de transport. On rappelle qu'une opération de transport se note $O_{i,j,u,x}$ ou (i,j,u,x) traduisant un transport de x unités de ressources par le véhicule T_u entre les activités A_i et A_j. Sur la Figure 5-4, le véhicule T_1 effectue les opérations de transport $O_{0,1,1,3}$, $O_{1,2,1,2}$, $O_{2,4,1,2}$, dans cet ordre et le véhicule T_2 effectue les opérations de transport $O_{0,3,2,1}$, $O_{1,3,2,1}$, $O_{3,4,2,2}$ dans cet ordre.

5.1.4 La première phase : « flow-first »

La première phase de la méthode de résolution permet de définir un flot respectant les contraintes du RCPSPR. Dans le chapitre précédent, une méthode permettant de construire un flot à partir d'un vecteur d'activités est proposée. Cette méthode assure le respect des contraintes de ressource et de précédence du problème. D'autres méthodes peuvent être proposées pour résoudre le problème de flot associé au RCPSPR. Cependant, grâce aux résultats obtenus dans le chapitre précédent, on sait qu'une bonne solution pour le problème de flot n'est pas nécessairement une bonne solution pour le problème du RCPSPR. La méthode proposée lors de cette première phase doit donc avant tout permette la génération d'un flot de façon efficace, *i.e.* peu coûteuse en temps de calcul. Le flot ainsi construit peut se modéliser grâce à un graphe $G_\varphi(V,E)$ dans lequel les nœuds modélisent les activités et les arcs les échanges de ressource entre les activités, avec un coût égal à la quantité de ressource échangée.

En reprenant l'exemple présenté dans ce chapitre et grâce à la méthode de construction d'un flot énoncée dans le chapitre précédent, un vecteur d'activité $w = [0,1,2,3,4]$ est généré. À partir de ce vecteur, le flot représenté sur la Figure 5-5 sous forme de graphe est construit.

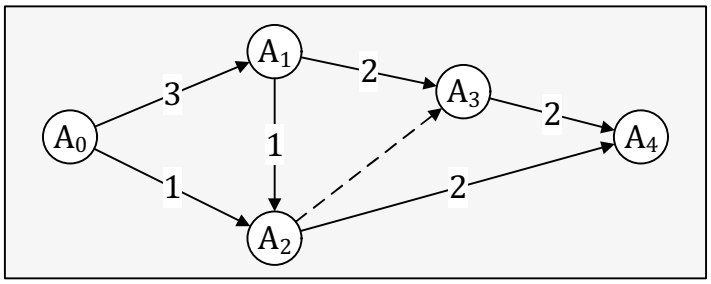


Figure 5-5. Graphe flot $G_\varphi(V,E)$ de l'exemple

Ce graphe est composé de cinq nœuds, $|V| = 5$, et de six arcs, $|E| = 6$. L'arc en pointillé permet de modéliser la contrainte de précédence entre l'activité A_2 et l'activité A_3. Les autres contraintes de précédence ne sont pas modélisées puisqu'elles sont nécessairement vérifiées grâce aux arcs portant du flot.

5.1.5 La deuxième phase : « route and cluster-second »

La première phase, en fournissant un flot pour le problème du RCPSP, permet de répondre à la question : Quelle est la quantité de ressource transférée d'une activité à une autre ? Avec cette information, les opérations de transfert sont fixées. On rappelle qu'une opération de transfert est définie par un triplet (i, j, φ_{ij}) qui modélise la nécessité d'échanger φ_{ij}ressources entre l'activité A_i et l'activité A_j.

Une solution du RCPSPR est constituée d'un ensemble de tournées et une tournée est constituée d'une ou plusieurs opérations de transport toutes affectées au même véhicule. L'enjeu de cette deuxième phase est de déterminer la meilleure solution du RCPSPR respectant le flot/les opérations de transfert définies lors de la première phase. Cette deuxième phase doit permettre de répondre aux questions suivantes :

- Quelle est la quantité de ressource transportée lors d'une opération de transport et par quel véhicule ?
- Dans quel ordre les opérations de transport sont-elles effectuées ?

Afin de répondre à toutes ces questions, en respectant les contraintes du problème et du flot, un algorithme de plus court chemin à contraintes de ressources effectué sur un graphe auxiliaire est introduit. Cette proposition s'inspire de l'approche de résolution « route-first, cluster-second » proposée par (Beasley, 1983) dans laquelle la deuxième phase peut être résolue optimalement avec l'utilisation d'un algorithme de plus court chemin sur un graphe auxiliaire. Néanmoins, le graphe auxiliaire et l'algorithme de plus court chemin diffèrent fortement entre ces deux approches.

5.2 Les algorithmes de plus court chemin à contraintes de ressources

Le problème de plus court chemin (SPP – Shortest Path Problem) est un problème classique très utilisé dans les problèmes d'optimisation (Ford, 1956), (Minty, 1957), (Bellman, 1958), (Dijkstra, 1959), aussi bien dans les problèmes d'ordonnancement que dans les problèmes de tournées de véhicules. Ce problème consiste à calculer le plus court

chemin d'un nœud source à un ensemble de nœuds dans un graphe. Il existe plusieurs classifications possibles pour ces méthodes, mais on peut « classiquement » considérer qu'on distingue les méthodes de type « label-setting » et les méthodes de type « label-correcting » comme le propose (Gilsinn and Witzgall 1973).

Une méthode est dite de type « label setting », si un label défini sur un nœud est un label permanent (Beasley et Christofides 1989), (Feillet *et al.*, 2004), c'est-à-dire un label qui n'est pas remis en question dans la suite de l'algorithme (l'algorithme de Dijkstra permettant de calculer un plus court chemin dans un graphe pondéré positivement est un algorithme de type « label setting »). La propagation des labels dans le graphe s'effectue nœud par nœud sans revenir sur les nœuds déjà visités. Un algorithme est dit de type « label-correcting » si un label sur un nœud peut être modifié dans les itérations suivantes de l'algorithme (Bardossy and Shier 2006) (les labels ne deviennent permanents qu'à la fin de l'algorithme). Dans la majorité des cas, on utilise des approches de type « label-correcting » plutôt que des approches de type « label-setting » (Glabowski *et al.*, 2014) dans le mesure où l'on peut trouver dans certains problèmes des arcs négatifs (pensons au time-lags maximaux qui sont utilisés couramment pour modéliser les fenêtres de temps) et on ne dispose alors pas d'un ordre topologique.

Une distinction importante peut être faite en fonction du nombre d'objectifs à minimiser. Dans le cas de plusieurs objectifs (MOSPP - Multi-Objective Shortest Path Problem), on recherche l'ensemble des chemins non dominés sur le nœud final (Gandibleux *et al.*, 2006).

Dans les problèmes de plus court chemin, aucune contrainte n'est imposée sur les arcs et tous les chemins sont envisagés. Dans certains cas, le passage par certains chemins doit respecter des contraintes additionnelles (en plus d'être le plus court chemin). On parle alors de plus court chemin avec contraintes (CSPP – Constrained Shortest Path Problem). Parmi les problèmes de plus court chemin avec contraintes (Beasley and Christofides, 1989), on peut citer le problème de plus court chemin à contraintes de ressources (RCSPP – Resource Constrained Shortest Path Problem) dans lequel les arcs/nœuds peuvent consommer des ressources avec une quantité limitée de ressources. Le problème du RCSPP est initialement introduit dans la thèse de (Desrochers, 1986). Les problèmes de type RCSPP peuvent être divisés en deux catégories :

- les problèmes avec une seule ressource (par exemple, un nombre maximal de véhicule à utiliser, un nombre maximal d'arcs à traverser) ;
- les problèmes avec plusieurs ressources qui vont faire intervenir par la suite des notions de dominance et créer des solutions incomparables entre-elles.

Dans ces deux cas, on peut distinguer les problèmes dans lesquels la quantité de ressource disponible est bornée supérieurement et uniquement supérieurement (comme par exemple, un nombre maximal d'arcs à traverser), et les problèmes dans lesquels la quantité de ressource disponible est bornée à la fois inférieurement et supérieurement. Ce deuxième cas apparaît, par exemple, dans les calculs de plus court chemin où l'on cherche un chemin qui utilise exactement k arcs dans le graphe. Les problèmes dans lesquels on trouve des contraintes sur un nombre de nœuds ou d'arcs à utiliser sont parfois classés dans la catégorie des « Vertex Constrained Shortest Path Problem » (Beasley and Christofides, 1989).

5.2.1 Les algorithmes de plus court chemin dans les approches « order-first, split-second »

On peut remarquer que dans les approches de type « order-first, split-second » (Prins et al, 2014), les algorithmes de type SPLIT permettent de calculer un plus court chemin dans un graphe pour lequel on connaît un ordre topologique. L'ordre topologique utilisé est l'ordre des sommets défini par un tour géant. Le premier algorithme de type SPLIT est dédié au VRP avec l'approche « route-first, cluster-second » introduite par (Beasley, 1983). Plus de 70 articles sont désormais disponibles sur des approches similaires, avec des extensions sur d'autres problèmes, comme par exemple, le HVRP (Heterogeneous fleet Vehicule Routing Problem) ou le CLRP (Capacitated Location Routing Problem). Le HVRP est une extension du VRP avec plusieurs véhicules de capacités différentes. Le CLRP est une extension du LRP avec des dépôts de capacité limités. Le problème du LRP consiste à déterminer la localisation d'un ensemble de dépôts et à déterminer les tournées des véhicules permettant de satisfaire les demandes d'un ensemble de clients, en minimisant un coût lié aux dépôts, aux véhicules et aux tournées.

Le SPLIT classique (VRP) repose sur un algorithme de plus court chemin utilisant une approche de type « label setting ». Le plus court chemin est calculé sur un graphe auxiliaire $G = (V, E)$, dont l'ensemble des nœuds V permet de modéliser les clients du tour géant avec un nœud fictif supplémentaire numéroté 0. L'ensemble des arcs E permet de modéliser les tournées réalisables (*i.e.*, les tournées respectant les contraintes de capacité et la taille maximale des tournées), ces arcs sont pondérés par un coût lié à la durée de la tournée associée. Les labels utilisés pour calculer le plus court chemin dans le graphe contiennent une information sur la solution, le coût du plus court chemin entre le nœud 0 et le nœud courant, et une information permettant de connaître le numéro du nœud correspondant au label (label père) ayant permis la création du label. À la fin de l'algorithme, un seul label est stocké sur chaque nœud, le makespan de la meilleure solution est donné par le label sur le nœud final. Pour retrouver le détail de la solution avec les tournées du véhicule, le graphe est parcouru de la fin au début en utilisant les informations sur les labels père.

Le SPLIT proposé pour le HVRP par (Duhamel *et al.*,2012) repose sur un algorithme de plus court chemin à contraintes de ressources. La structure du graphe auxiliaire utilisée est identique à celle présentée pour le VRP, mais les contraintes de ressource sont prises en compte dans les labels. Les labels contiennent plusieurs informations, le coût du plus court chemin entre le nœud 0 et le nœud courant, la quantité de ressources disponibles de chaque type, *i.e.*, le nombre de véhicules de chaque type (en fonction de la capacité) et, les informations sur le label père. Cette définition, proposée pour les labels, engendre des configurations dans lesquelles deux labels, sur un même nœud, peuvent être incomparables. Par conséquent, à la fin de l'algorithme, plusieurs labels peuvent être stockés sur chaque nœud, et on peut fournir à la fin de l'algorithme un ensemble de solutions. On peut s'intéresser, par exemple, à la solution stockée sur le nœud final, qui possède le makespan minimal. Comme pour le SPLIT classique, le détail de la solution avec les tournées des véhicules est obtenu en remontant le graphe en utilisant les informations sur le label père. Les informations sur le label père contiennent à la fois, le numéro du nœud et le numéro du label (label père) sur le nœud ayant permis la création du label courant.

Un autre exemple de SPLIT peut être présenté, celui proposé pour le CLRP (Duhamel *et al.*,2010) et qui repose aussi sur un algorithme de plus court chemin à contraintes de ressources. Dans ce problème, le coût des arcs dans le graphe auxiliaire dépend du dépôt sélectionné pour effectuer la tournée associée à cet arc. De plus, la demande totale de toutes les tournées affectées à un dépôt ne doit pas dépasser la capacité de celui-ci. Ces contraintes permettent de définir les labels qui sont utilisés pour trouver le plus court chemin dans le graphe. Un label contient alors le coût du plus court chemin entre le nœud 0 et le nœud courant, le nombre de véhicules (de chaque type) restant disponibles, la capacité résiduelle de chaque dépôt, et les informations sur le label père. Comme pour le SPLIT présenté pour le HVRP, deux labels peuvent être incomparables. Les meilleures solutions du problème sont stockées à la fin de l'algorithme sur le nœud final.

Tous ces algorithmes SPLIT sont définis à partir d'un graphe dans lequel l'ordre des sommets du tour géant définit un ordre topologique, permettant des implémentations en $O(n)$ dans les cas classiques (si on reprend la proposition de (Vidal, 2015)). Dans les autres cas, qui nécessitent la définition de labels permettant la gestion des ressources, bien que la complexité des algorithmes augmente, on conserve encore des approches de type « label setting ». Cette caractéristique signifie que, lorsque tous les labels sur un nœud i sont propagés vers tous les nœuds j (tels que j$> i$), alors plus aucune propagation de label ne peut être effectuée à destination ou à partir du nœud i.

5.2.2 Spécificité de l'algorithme de plus court chemin à contraintes de ressources proposé pour le RCPSPR

L'algorithme de plus court chemin proposé pour résoudre le RCPSPR à partir d'un flot, repose sur des principes similaires aux algorithmes SPLIT cités précédemment, avec la définition d'un graphe, d'un label et d'une règle de propagation. Néanmoins plusieurs points-clés diffèrent.

Le but de cet algorithme est de déterminer la tournée de chaque véhicule en définissant : 1) les quantités transportées par chaque opération de transport, 2) l'ordre des opérations de transport et 3) l'affectation des opérations de transport aux véhicules. Ces trois points ont un impact sur la définition du graphe auxiliaire sur lequel le plus court chemin à contraintes de ressources est effectué. Ce graphe étant construit à partir d'un flot (opérations de transfert) et l'ordre des opérations de transport devant être déterminé par l'algorithme, aucun ordre topologique ne peut être défini sur les arcs portant du flot (sauf pour le cas particulier où le graphe flot est constitué d'un chemin unique entre la source et le puits). Par conséquent, le graphe auxiliaire peut être composé de cycles, nécessaires à la définition de l'ordre des opérations de transport.

De plus, le problème du RCPSPR intègre des contraintes de précédence entre les activités et des contraintes de ressource à respecter. Pour pouvoir garantir la faisabilité des solutions partielles créées lors de l'algorithme, le label doit contenir des informations sur l'état global du système. Ces informations englobent l'état des véhicules, des activités mais aussi des ressources. Le label doit donc contenir des informations qui sont liées au problème d'ordonnancement et au problème de tournées de véhicules.

5.3 Proposition d'un algorithme de plus court chemin à contraintes de ressources

L'algorithme de plus court chemin à contraintes de ressources repose sur la définition d'un graphe auxiliaire spécifique ainsi que sur la définition d'un label intégrant des informations sur les activités, les véhicules et les ressources. Ces points sont explicités dans la suite de ce chapitre et illustrés avec l'exemple présenté précédemment.

5.3.1 Définition du graphe auxiliaire

Le graphe auxiliaire $G'(S \cup U \cup V, E \cup F)$ peut être construit à partir des opérations de transfert définies par le flot. Dans ce graphe, trois ensembles de nœuds et deux ensembles d'arcs sont modélisés :

- S : le couple de nœuds # et $*$ modélisant respectivement le nœud source et le nœud puits dans le graphe auxiliaire ;
- U : l'ensemble des nœuds modélisant les activités sur lesquelles une opération de collecte (pickup) est effectuée ;
- V : l'ensemble des nœuds modélisant les activités sur lesquelles une opération de livraison (delivery) est effectuée ;
- E : l'ensemble des arcs requis, *i.e.* modélisant des opérations de transfert permettant de définir des opérations de transport ;
- F : l'ensemble des arcs non requis modélisant les déplacements à vide des véhicules entre deux opérations de transport ou des temps d'attentes des véhicules.

Pour reprendre la terminologie des problèmes « d'arc routing », le terme d'« arc requis » (Ramdane-Chérif, 2002) est utilisé pour référencer un arc entre un nœud de collecte et un nœud de livraison, alors que le terme d'« arc non requis » est défini entre un nœud de livraison et un nœud de collecte. Les arcs requis sont pondérés grâce à un couple (φ_{ij}, t_{ij}) indiquant la quantité de ressource φ_{ij} à transférer entre les activités et le temps de transport t_{ij} associé. Les arcs non requis sont pondérés par $(0, t_{ij})$ indiquant : 1) la quantité de ressource à transférer entre les activités, quantité qui est obligatoirement nulle puisque la définition du flot interdit un échange de ressources entre ces activités ; 2) le temps de transport t_{ij} associé.

En reprenant l'exemple présenté dans ce chapitre et à partir du flot représenté sur la Figure 5-5, le graphe auxiliaire $G'(S \cup U \cup V, E \cup F)$ est construit, avec $S = \{\#,*\}$, $U = \{A_0, A_0, A_1, A_1, A_2, A_3\}$, $V = \{A_1, A_2, A_2, A_3, A_4, A_4\}$, $|E| = 6$, and $|F| = 20$. Deux représentations de ce graphe peuvent être faites. La première, sur la Figure 5-6, utilise une représentation linéaire des activités et des arcs requis, cette représentation se rapproche des représentations usuelles utilisées pour les graphes auxiliaires avec les algorithmes SPLIT.

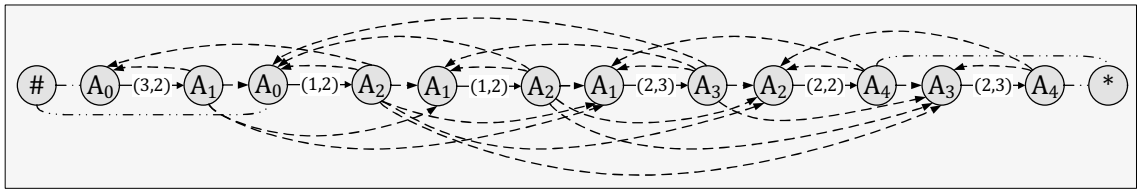


Figure 5-6. Graphe auxiliaire $G'(S \cup U \cup V, E \cup F)$ de l'exemple (représentation horizontale)

La deuxième, sur la Figure 5-7, utilise une représentation verticale des activités et des arcs requis, cette représentation se rapproche des représentations utilisées dans les problèmes de tournée sur arcs (Eiselt *et al.*, 1995) ou plus récemment par (Corbéran *et al.*, 2017).

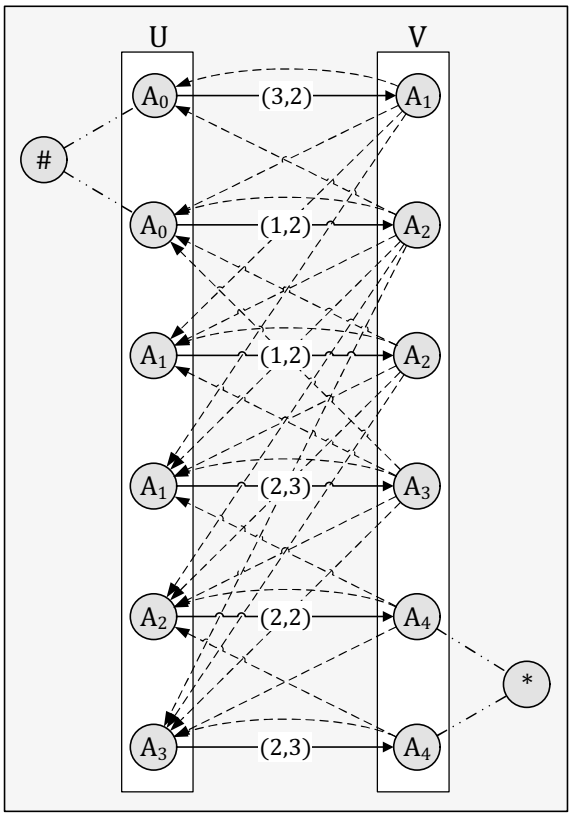


Figure 5-7. Graphe auxiliaire $G'(S \cup U \cup V, E \cup F)$ de l'exemple (représentation verticale)

Cette deuxième représentation (Figure 5-7) fait apparaître une caractéristique du graphe auxiliaire qui est dit biparti puisque chaque arc de $E \cup F$ a une extrémité dans U et l'autre dans V.

Sur la Figure 5-7, seuls les coûts sur les arcs requis sont représentés pour faciliter la lecture du graphe. Les arcs non requis sont représentés en pointillés sur les deux figures, ces arcs modélisent un déplacement à vide d'un véhicule et engendrent une relation de précédence entre les deux arcs requis reliés par cet arc non requis. Afin de respecter les contraintes de précédence du problème et celles imposées par le flot, certains arcs non requis ne peuvent être réalisables et ne sont donc pas modélisés dans le graphe auxiliaire.

Par exemple, on peut noter que l'arc non requis (A_1, A_3) entre l'arc requis (A_0, A_1) l'arc requis (A_3, A_4) n'est pas modélisé. En effet, ce déplacement à vide ne peut avoir lieu dans ce contexte puisqu'il engendre une violation des contraintes de ressource données par le flot. Plusieurs affirmations conduisant à une contradiction peuvent être énoncées si cette situation se produit :

- Pour que l'opération de transport entre les activités (A_3, A_4) puisse avoir lieu, l'activité A_3 doit avoir été livrée de toutes ses ressources.
- L'activité A_3 doit recevoir des ressources de l'activité A_1, d'après la définition du flot.

- L'opération de transport entre les activités (A_0, A_1) permet de livrer une partie ou la totalité des ressources nécessaires pour débuter l'activitéA_1.

Par conséquent, pour former une tournée réalisable l'opération de transport entre les activités (A_1, A_3) doit avoir lieu avant l'opération de transport entre les activités(A_3, A_4) et l'opération de transport entre les activités (A_0, A_1) doit avoir lieu avant l'opération de transport entre les activités(A_1, A_3). Les opérations (A_0, A_1) et (A_3, A_4) ne peuvent donc pas se succéder immédiatement.

5.3.2 Principe de l'algorithme de plus court chemin à contraintes de ressources

L'algorithme de plus court chemin à contraintes de ressources considéré dans ce chapitre permet de définir une solution optimale du RCPSPR par rapport à un flot donné en utilisant le graphe auxiliaire $G'(S \cup U \cup V, E \cup F)$. Une solution du problème peut se modéliser par un chemin dans le graphe auxiliaire $G'(S \cup U \cup V, E \cup F)$ du nœud # au nœud $*$ passant une ou plusieurs fois sur chaque arc avec un flot non nul. Les meilleures solutions, obtenues à l'issue de l'algorithme, minimisent la date d'arrivée de tous les véhicules sur le nœud $*$, en respectant toutes les contraintes. On choisit aléatoirement une solution parmi les meilleures, que l'on définit comme étant une meilleure solution.

L'algorithme repose sur cinq points-clés :

- l'utilisation de labels permettant la représentation partielle des solutions, ces labels ne contiennent que les informations sur l'état courant du système ;
- la définition d'une une règle de dominance limitant le nombre de labels stockés ;
- le précalcul de bornes inférieures pour chaque activité afin de limiter la propagation des labels « non prometteurs » ;
- la définition d'un sous-ensemble de labels « prometteurs » permettant de restreindre l'ensemble des labels « «prometteurs ». Cette restriction étant basée sur une heuristique, elle ne garantit pas l'optimalité du découpage ;
- une structure de données Λ de type pile pour gérer efficacement la propagation des labels.

Pour des raisons d'efficacité, les labels utilisés dans l'algorithme, ne stockent pas les dates de début de toutes les opérations de transport, mais uniquement l'état courant du système et des ressources. L'algorithme se termine donc en donnant l'un des meilleurs labels trouvés L^* sur le nœud final. Ce label permet de connaître les disjonctions entre les opérations de transport mais pas les dates de ces opérations. Pour obtenir le détail d'une solution, il faut reconstruire le graphe disjonctif et l'évaluer, dans le même principe que ce qui est présenté dans le chapitre 4 lors de la cinquième étape de l'évaluation.

Chaque solution partielle correspond à un chemin dans le graphe auxiliaire $G'(S \cup U \cup V, E \cup F)$ et est représenté par un label $L(f, S, R)$ qui fournit toutes les informations sur la solution, avec la fonction objectif, l'état du système et l'état des ressources permettant l'utilisation d'une règle de dominance. Une solution partielle est caractérisée par un label dans lequel tous les échanges de ressources imposés par le flot n'ont pas encore été ordonnancés et affectés à des véhicules, par conséquent, toutes les activités ne sont pas encore ordonnancées. À l'inverse, une solution finale est modélisée par un label où toutes les ressources ont été transportées, c'est-à-dire un label où toutes les activités sont ordonnancées.

L'ensemble des labels stockés sur chaque nœud i dans le graphe auxiliaire est divisé en deux sous-ensembles de labels dans l'algorithme :

- Y_i l'ensemble des labels non propagés sur tous les arcs (i, j) permettant l'obtention d'un label correspondant à une solution partielle réalisable.
- P_i l'ensemble des labels propagés et non dominés.

Sur chaque nœud, les labels appartenant à l'ensemble Y_i sont triés par ordre croissant sur un critère propre à chaque label. Ce critère donne une estimation de la durée nécessaire pour transporter les ressources restantes, avec un seul véhicule et en ne comptabilisant le transport que sur les arcs requis. L'ensemble Y_i peut-être restreint aux N_L premiers labels afin de limiter le nombre de labels sur chaque nœud. Cet ensemble est noté Z_i.

Le graphe auxiliaire n'étant pas caractérisé par un ordre topologique, l'ordre dans lequel les nœuds du graphe sont balayés a une influence significative sur le comportement de l'algorithme. Cette propriété offre un degré supplémentaire de liberté et plusieurs règles sont envisageables pour déterminer l'ordre de parcours des nœuds du graphe sur lesquels des labels doivent être propagés. Une structure de données Λ, de type pile, est utilisée pour stocker les numéros de nœuds contenant des labels non encore propagés. Cette structure offre un compromis intéressant en termes d'implémentation pour le parcours du graphe. Afin de limiter le nombre de labels non propagés sur chaque nœud, cette structure de données est triée par ordre décroissant sur le nombre de labels non propagés.

L'utilisation d'une règle de dominance est facultative dans le sens où l'algorithme de plus court chemin à contraintes de ressources énumère tous les chemins possibles à partir du noeud # vers le nœud $*$, mais celle-ci permet de réduire le nombre de labels à propager et ainsi de réduire le temps d'exécution de l'algorithme. De la même façon, l'introduction d'une borne supérieure au problème ainsi que l'estimation d'une borne inférieure sur tout nœud du graphe pour obtenir une solution du problème permet de limiter le nombre de labels stockés. La borne supérieure peut être déterminée en utilisant les étapes 2, 3 et 4 de la fonction d'évaluation dans le chapitre 4.

Le principe de l'algorithme de plus court chemin à contraintes de ressources pour résoudre le RCPSPR est présenté dans l'Algorithme 3. Le problème consiste à calculer un chemin de coût minimal respectant les contraintes du problème se terminant sur le noeud * à l'aide de labels.

L'algorithme commence par une étape d'initialisation (lignes 18-19) où tous les paramètres sont initialisés. L'initialisation consiste à définir le label initial et à le propager sur les nœuds correspondant à l'activité modélisant le début du projet. Tous les autres nœuds du graphe auxiliaire ne doivent contenir aucun label et la structure de type pile doit être initialisée en fonction des labels initiaux à propager. Les lignes 22 à 38 définissent la boucle for dans laquelle tous les labels non propagés sur un nœud sont propagés le long des arcs correspondant à un chemin réalisable en utilisant tous les véhicules disponibles. Les nouveaux labels générés sont insérés s'ils ne sont pas dominés par des labels déjà présents sur le nœud destination, et si la solution partielle associée à ce label permet l'obtention d'une meilleure solution finale que celles déjà générées. L'algorithme se termine lorsque tous les labels sont propagés, ce qui se traduit par une pile vide (lignes 20-39). L'algorithme mémorise les meilleurs labels S non dominés correspondant à une solution du problème du RCPSPR (ligne 32). Toutes ces étapes

reposent sur l'utilisation de plusieurs sous-procédures permettant la mise à jour du graphe auxiliaire et des labels.

Pour l'initialisation des paramètres, la fonction `InitLabel`$(N, flow, b_i)$ retourne le label initial L_0 associé au flot et la procédure `InitGraph`(L_0, P_i, Z_i, Λ) initialise le graphe auxiliaire avec le label initial. Cette procédure initialise les ensembles P_i, Z_i à l'ensemble vide pour tous les nœuds $i \in U \cup V$ sauf les nœuds correspondant à l'activité A_o pour lesquels le label initial est stocké dans l'ensemble des nœuds non propagés. Les nœuds pour lesquels $Z_i \neq \{\}$ sont ajoutés à la pile Λ.

Les contraintes de précédence sont vérifiées grâce à la boucle de la ligne 23 à la ligne 35 permettant de ne pas choisir un nœud destination $node\_j$ si celui-ci ne fait pas partie des successeurs au nœud $node\_i$. L'ensemble des successeurs d'un nœud k est donné par `Succ`(k), cet ensemble est déduit des relations de précédence. La fonction booléenne `Path`$(L_i, node\_j)$ (ligne 24) retourne vrai si le label L_i peut-être propagé sur l'arc $(node\_i\ , node\_j)$ sans violer les contraintes de ressource liées au problème et au flot.

La règle de propagation est implémentée grâce à la fonction `CreateLabel`$\left(L_i, node_i, node_j, v\right)$ (ligne 26) qui retourne le label L_w généré à partir du label L_i sur l'arc $(node\_i\ , node\_j)$ en utilisant le véhicule v. La procédure `InsertLabel`(L_w, Z_{node_j}, N_L) (ligne 28) permet l'insertion du label L_w dans l'ensemble Z_{node_j} en respectant l'ordre de tri et la limitation sur le nombre de labels stockés. Ensuite, la procédure `ApplyDominance`(L_w, Z_{node_j}) (ligne 30) supprime tous les labels dans Z_{node_j} qui sont dominés par le label L_w.

Deux fonctions booléennes peuvent interdire l'insertion d'un label (ligne 27). La fonction `Dominate`(L_w, Z_{node_j}) qui retourne vrai si le label L_w est dominé par aucun label dans l'ensemble Z_{node_j} et faux sinon. La fonction `CheckUbLb`(L_w, LB_{node_j}, UB) retourne vrai si le coût associé au label L_w, plus une estimation (LB_{node_j}) du coût restant pour atteindre le nœud final du graphe à partir du nœud $node\_j$, est inférieur au coût de la meilleure solution courante (UB) et faux sinon.

Enfin, la procédure `CheckSol`$\left(L_w, UB, L^*\right)$ sauvegarde le label L_w comme meilleur label s'il correspond à une solution finale et si le coût du label L_w et inférieur au coût du label L^*. Si c'est le cas, la borne supérieure est également mise à jour.

Algortihme 3 : Algorithme du plus court chemin à contraintes de ressources

```
1.procedure Shortest_Path
2.  input parameters
3.    G'(U,V,E,F): auxiliary graph
4.    UB: upper bound of the problem
5.    LB_i: lower bound to reach the final node from node i ∈ U ∪ V
6.    flow: flow (quantity of resources exchanged between two activities)
7.    b_i: quantity of resources required by activity i ∈ A
8.  output parameters
9.    Y_i : ordered set of unprocessed labels on node i ∈ U ∪ V
10.   P_i : set of processed and non-dominated labels on node i ∈ U ∪ V
11.   Z_i : restriction of Y_i to the N_L first labels on node i ∈ U ∪ V
12.   L* : best found label at node *
13.  global parameter
14.   Λ: stack
15.   N: number of vehicles
16.   N_L: maximum number of unprocessed labels into Z_{i∈U∪V}
17.  begin
18.  | L_o := InitLabel(N, flow, b_{i∈A})
19.  | InitGraph(L_o, P_i, Z_i, Λ)
20.  | while (Λ != ∅) do              //while the stack is not empty
21.  | | node_i := Pop(Λ)
22.  | | for L_i ∈ Z_node_i do                  //for each unprocessed label
23.  | | | for node_j ∈ Succ(node_i) do        //for each destination node
24.  | | | | if Path(L_i, node_j) then         //if the path is feasible
25.  | | | | | for v:= 1 to N do               //for each vehicle
26.  | | | | | | L_w := CreateLabel(L_i, node_i, node_j, v)
27.  | | | | | | if (Dominate(L_w, Z_node_j) and CheckUbLb(L_w, LB_node_j, UB)) then
28.  | | | | | | | InsertLabel(L_w, Z_node_j, N_L)
29.  | | | | | | | Push(Λ, node_j)
30.  | | | | | | | ApplyDominance(L_w, Z_node_j)
31.  | | | | | | endif
32.  | | | | | | if (λ(node_j) = *) then CheckSol(L_w, UB, L*); endif
33.  | | | | | endfor
34.  | | | | endif
35.  | | | endfor
36.  | | | Z_node_i = Z_node_i \ L_i
37.  | | | P_node_i = P_node_i ∪ L_i
38.  | | endfor
39.  | endwhile
40.  end
```

5.3.3 Règles de gestion des labels

L'algorithme de plus court chemin à contraintes de ressources doit à la fois gérer la disponibilité des véhicules, des activités et des ressources. Pour ce faire les points suivants qui ont été mentionnés précédemment sont détaillés dans la suite de ce chapitre :

- la définition d'un label caractérisant l'état du système à un instant donné ;
- une règle d'initialisation du label ;
- une règle de propagation pour créer un nouveau label ;
- une règle de vérification du label par rapport aux contraintes du problème ;

- une règle de dominance pour stocker uniquement les labels non dominés sur les nœuds ;
- une règle de sélection pour la propagation des labels afin de choisir les labels à propager à chaque itération.

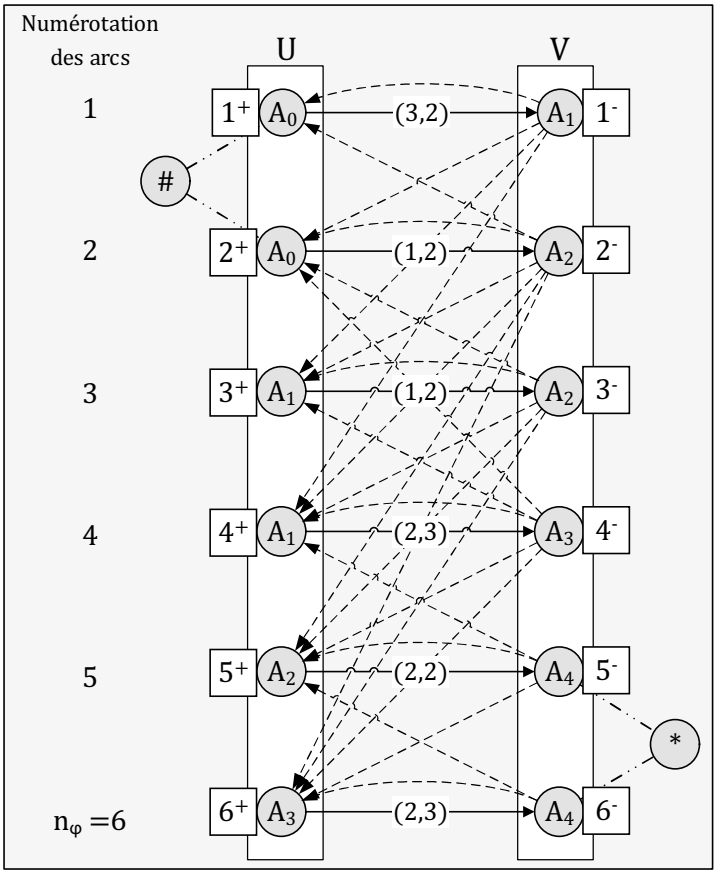


Figure 5-8. Graphe auxiliaire $G'(S \cup U \cup V, E \cup F)$ avec toutes les notations

Afin de faciliter la lecture du ce chapitre, une fonction β permettant d'attribuer une valeur dans $[\![1, n_\varphi]\!]$ à chaque arc $e \in E$ est introduite (n_φ étant le nombre d'arcs portant un flot non nul) :

$$\beta: e \in E \rightarrow [\![1, n_\varphi]\!],$$
$$\beta(e) = i, \qquad \forall e_1 \in E, \forall e_2 \in E,\ e_1 \neq e_2, \beta(e_1) \neq \beta(e_2).$$

Une autre notation est également introduite pour identifier le nœud de collecte appartenant à U et le noeud de livraison appartenant V de chaque arc numéroté $i \in [\![1, n_\varphi]\!]$. Le nœud de collecte associé à l'arc i est désigné par la notation i^+ et le nœud de livraison associé à l'arc i est désigné par la notation i^-. De plus, afin de connaître l'activité associée à chacun de ces nœuds, la fonction $\lambda: i^\pm \in [\![1, n_\varphi]\!]^\pm \rightarrow A$ est définie. L'activité associée au nœud de collecte i^+ (resp. livraison i^-) est donnée par $\lambda(i^+)$ (resp. $\lambda(i^+)$). Ces notations sont explicitées sur la Figure 5-8 reprenant l'exemple énoncé précédemment. L'ordre de numérotation des arcs n'est pas fixé par les données du problème. Dans l'exemple illustré ci-dessous, l'ordre respecte les relations de précédence engendrées par le flot.

Définition du label

Soit L_i^j le $i^{ème}$ label sur le nœud $j^{\pm}$, ce label permet de modéliser l'évaluation d'une solution partielle du problème du RCPSPR. Un label $L_i^j = (f, S, R)$ contient trois catégories d'information :

- Le coût de la solution f, correspondant à la date de disponibilité de tous les véhicules.
- L'état du système S, regroupant les informations sur les véhicules et les activités, avec :
  - Le $3v$-uplet V_i^j contenant à la fois la position, les dates d'arrivés et de départ des véhicules $V_i^j = \left(P_{ij}^1, \dots, P_{ij}^N, B_{ij}^1, \dots, B_{ij}^N, A_{ij}^1, \dots, A_{ij}^N\right)$, avec $P_{ij}^1, \dots, P_{ij}^N$ les positions de chaque véhicule ($P_{ij}^u = p$ signifie que le véhicule T_u est localisé sur l'activité A_p, $B_{ij}^1, \dots, B_{ij}^N$ les dates de départ au plus tôt de tous les véhicules, et $A_{ij}^1, \dots, A_{ij}^N$ les dates de départ au plus tôt de tous les véhicules.
  - Le $n+1$-uplet S_i^j contenant les dates de début au plus tôt de chacune des activités $S_i^j = \left(S_{ij}^0, \dots, S_{ij}^{n+1}\right)$;
- L'état des ressources R, permettant de donner l'avancement du système du point de vue des ressources avec :
  - Le n_φ-uplet $\varphi_i^{R,j}$ contenant la quantité de ressources restant à transporter sur chaque arc requis $\varphi_i^{R,j} = (\varphi_{ij}^{R,1}, \dots, \varphi_{ij}^{R,n_\varphi})$.
  - Le $n+1$-uplet $b_i^{R,j}$ contenant la quantité de ressources restant à transporter pour chaque activité $b_i^{R,j} = (b_{ij}^{R,0}, \dots, b_{ij}^{R,n+1})$.

Règle d'initialisation du label

Un label initial est stocké sur le nœud # et est propagé sur tous les nœuds $j = j^+$ tels que $\lambda(j^+) = A_0$. Le label initial est défini par les valeurs suivantes :

$\forall\, T_u \in T,\ P_{1j}^u = 0$
$\forall\, T_u \in T,\ A_{1j}^u = 0$
$\forall\, T_u \in T,\ B_{1j}^u = 0$
$\forall A_0 \in V/\{A_0\},\ S_{1j}^k = -\infty$ et $S_{1j}^0 = 0$
$\forall i \in [1..n_\varphi],\ \varphi_{1j}^{R,i} = \varphi_{\lambda(j^+),\lambda(j^-)}$
$\forall\, A_k \in V,\ b_{qj}^{R,k} = b_k$
$f_{1j} = 0$

Règle de propagation du label

La règle de propagation dépend de la nature de l'arc utilisé pour propager le label (arc requis ou non). Les informations à mettre à jour diffèrent puisque, lors de la livraison de ressource dans le problème aucun temps d'attente n'est nécessaire alors que la collecte de ressource peut nécessiter un temps d'attente du véhicule pour attendre la fin de l'activité sur laquelle la collecte est effectuée.

Règle de propagation du label sur un arc requis (opération de transport)

Une fonction de propagation définie par $f: L \otimes T \rightarrow R$ permet de générer un nouveau label $L_q^{i^-}$ à partir d'un label $L_p^{i^+}$, d'une opération de transport définie par un coût $(\varphi_{mk}, t_{mk})\ /\ m = \lambda(i^+), k = \lambda(i^-)$ et un véhicule $T_u \in T$. Le nouveau label L_q^j sur le nœud $j = i^-$ est défini grâce au label L_p^i sur le nœud $i = i^+$ en utilisant le véhicule T_u avec les mises à jour suivantes :

$$P_{qj}^u = \lambda(j)$$
$$A_{qj}^u = B_{pi}^u + t_{\lambda(i),\lambda(j)}$$
$$B_{qj}^u = A_{qj}^u$$
$$S_{qj}^{\lambda(j)} = max(A_{qj}^u;\ S_{pi}^{\lambda(j)})$$
$$\varphi_{qj}^{R,i} = \varphi_{pi}^{R,i} - min(\varphi_{pi}^{R,i};\ c_u)$$
$$b_{qj}^{R,\lambda(j)} = b_{pi}^{R,\lambda(j)} - min(\varphi_{pi}^{R,i};\ c_u)$$
$$f_{qj} = max(f_{pi};\ B_{qj}^u)$$

La règle de propagation met également à jour toutes les dates de début des activités qui succèdent à l'activité $\lambda(i)$, *i.e.*, $\forall A_s \in succ\big(\lambda(i)\big), S_{qj}^s = max(S_{pi}^s;\ S_{qj}^{\lambda(j)} + p_{\lambda(j)})$. Cette mise à jour garantit le respect des contraintes de précédence. Pour chaque label L_p^i, la règle de propagation itère sur tous les véhicules disponibles sur le nœud origine, permettant la création de un ou plusieurs labels à partir d'un seul. Ces labels sont stockés ou non sur le nœud destination en fonction des labels non dominés déjà stockés sur le nœud.

Règle de propagation du label sur un arc non requis (déplacement à vide)

Une fonction de propagation définie par $f: \mathrm{L} \otimes \mathrm{T} \rightarrow \mathrm{R}$ permet de générer un nouveau label $L_q^{j^+}$ à partir d'un label $L_p^{i^-}$, d'un déplacement à vide défini par un coût $(0, t_{mk})\ /\ m = \lambda(i^-), k = \lambda(j^+)$ et un véhicule $T_u \in T$. Le nouveau label L_q^j sur le nœud $j = j^+$ est défini grâce au label L_p^i sur le nœud $i = i^-$ en utilisant le véhicule T_u avec les mises à jour suivantes :

$$P_{qj}^u = \lambda(j)$$
$$A_{qj}^u = B_{pi}^u + e_{\lambda(i),\lambda(j)}$$
$$B_{qj}^u = max(A_{qj}^u;\ S_{pi}^{\lambda(j)} + p_{\lambda(j)})$$
$$f_{qj} = max(f_{pi};\ B_{qj}^u)$$

Règle de vérification du label

Un label ne peut être généré que s'il correspond à une solution partielle. Pour cela, les deux conditions suivantes doivent être respectées :

- La première condition provient du flot défini précédemment. Si le flot restant à transporter $\varphi_{qj}^{R,i} = 0$, c'est-à-dire que toutes les ressources sur l'arc $i = (\lambda(i^+), \lambda(i^-))$ sont transportées, alors un déplacement à vide vers l'arc i doit être interdit.

- La deuxième condition repose sur le fait qu'un déplacement à vide entre deux activités ne peut avoir lieu que si l'activité de destination $\lambda(j^+ = j)$ est déjà ordonnancée, c'est-à-dire, une activité pour laquelle toutes les ressources ont été livrées $b_{qi}^{R,\lambda(j)} = 0$. En effet, un déplacement à vide n'est utile que s'il est suivi d'une opération de transport. Or un transport à partir de l'activité $\lambda(j^+ = j)$ ne peut avoir lieu que si l'activité est terminée et donc que toutes les ressources ont été préalablement livrées.

Ces deux conditions sont liées à l'état des ressources et n'interviennent que lorsque la propagation est effectuée sur un arc non requis.

Règle de dominance du label

Chaque label généré par la règle de propagation est comparé aux labels non dominés déjà présents sur le nœud destination. Un label L_p^i domine un label L_q^i sur le nœud i ($L_q^i \ll L_p^i$), si les contraintes suivantes sont respectées :

Condition 1. Tous les véhicules sont localisés sur les mêmes activités :
$\forall u \in T, \quad P_{ip}^u = P_{iq}^u$
Condition 2. $\forall u \in T,\ B_{ip}^u \leq B_{iq}^u$
Condition 3. $\forall u \in T,\ A_{ip}^u \leq A_{iq}^u$
Condition 4. $\forall k \in [1..n_\varphi], \varphi_{ip}^{R,k} \leq \varphi_{iq}^{R,k}$
Condition 5. $S_{ip}^{n+1} \leq S_{ip}^{n+1}$
Condition 6. $\exists u \in T,\ B_{ip}^u < B_{iq}^u$
ou $\exists u \in T,\ A_{ip}^u < A_{iq}^u$
ou $\exists k \in [1..n_\varphi], \varphi_{ip}^{R,k} < \varphi_{iq}^{R,k}$
ou $S_{ip}^{n+1} < S_{ip}^{n+1}$

Si le label L_p^i ne domine pas le label L_q^i ($L_q^i \not\ll L_p^i$), cela n'implique pas que le label L_q^i domine L_p^i. De plus, si $L_q^i \not\ll L_p^i$ et si $L_p^i \not\ll L_q^i$, alors L_p^i et L_q^i sont dits incomparables. Par conséquent, si $\exists k\ L_p^i \ll L_k^i$, alors L_p^i n'est pas stocké sur le noeud i.

D'autre part, chaque label L_k^i tel que $L_k^i \ll L_p^i$ doit être supprimé du nœud i. La règle de dominance permet de limiter le nombre de labels stockés sur chaque nœud à un plus petit sous-ensemble tout en conservant l'optimalité de l'algorithme. Une autre approche, permettant de réduire le temps d'exécution de l'algorithme, peut être envisagée en limitant le nombre de labels stockés sur chaque nœud. Cette limitation, en plus de la règle de dominance permet de réduire très fortement le temps d'exécution de l'algorithme mais peut mener à l'obtention de solutions sous-optimales.

Règle de sélection des labels

Afin de déterminer l'ordre dans lequel la propagation des labels sur les nœuds est effectuée, une pile Λ est utilisée. Deux opérations peuvent être effectuées sur cette pile :

- Push (Λ, i): permet l'ajout du nœud numéro i dans la pile en garantissant l'unicité de ce nœud, *i.e.*, les répétitions ne sont pas autorisées ;

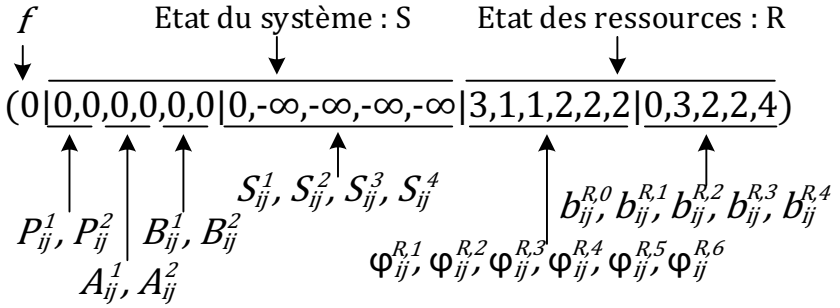
- Pop (Λ): permet de supprimer le nœud de la pile ayant le plus grand nombre de labels non propagés.

Cette approche de sélection des labels à propager favorise une propagation des labels permettant l'obtention de solutions finales plus rapidement.

5.3.4 Illustration de l'algorithme sur l'exemple

Les premières étapes de l'algorithme de plus court chemin à contraintes de ressources sont détaillées dans cette partie. L'exemple utilisé est celui présenté dans la section 5.1.3 et s'appuie sur le flot introduit sur la Figure 5-5 et sur le graphe auxiliaire présenté sur la Figure 5-8 .

La première phase de l'algorithme consiste à définir le label initial correspondant au flot. Le label initial est donc égal à :



Ce label noté $L^{\#}$ est stocké sur le nœud #. Ce label est propagé/stocké sur les nœuds 1^+ et 2^+car ils correspondent à l'activité A_0 ($\lambda(1^+) = A_0$ et $\lambda(2^+) = A_0$). Les deux labels sont notés $L_1^{1^+}$ et $L_1^{2^+}$. La pile est mise à jour et est égale à $\Lambda = [2^+, 1^+]$. Le graphe est alors initialisé.

La <u>première itération</u> de l'algorithme permet de dépiler le nœud numéro 1^+ de la pile. Tous les labels stockés sur le nœud sont alors propagés vers le nœud numéro 1^-. Cette première étape permet la création et l'insertion de deux labels correspondant à l'affectation du véhicule T_1 ou du véhicule T_2 :

- $L_1^{1^-} = (2|1,0,2,0,2,0|0,2,-\infty,-\infty,4|0,1,1,2,2,2|0,0,2,2,4)$
- $L_2^{1^-} = (2|0,1,0,2,0,2|0,2,-\infty,-\infty,4|1,1,1,2,2,2|0,1,2,2,4)$

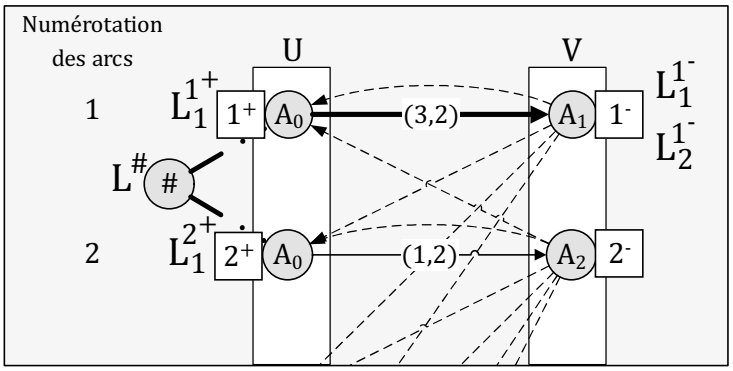


Figure 5-9. Représentation de l'état du graphe auxiliaire avec les labels à l'issue de la première itération

Le label $L_1^{1^-}$ est créé à partir du label $L_1^{1^+}$ et en choisissant d'utiliser le véhicule T_1 pour transporter les ressources entre l'activité A_0 et A_1 correspondant à l'arc numéro 1 de coût (3,2). L'opération de transfert associée à cet arc est égale à (0,1,3) avec $t_{0,1} = 2$. Le véhicule T_1 étant de capacité $C_1 = 3$, toutes les ressources peuvent être transportées. Le détail des calculs permettant la création du label $L_1^{1^-}$ à partir du label $L_1^{1^+}$, grâce à la règle de propagation sur les arcs requis, est explicité ci-dessous :

- $P_{1,1^-}^1 = \lambda(1^-) = 1$, la position du véhicule T_1 est mise à jour. Celui-ci est affecté à une opération de transport modélisée par l'arc numéro 1, le véhicule se déplace donc entre les activités A_0 et A_1. À la fin de l'opération de transport le véhicule est sur le nœud numéro 1^- correspondant à l'activité A_1.
- $A_{1,1^-}^1 = B_{1,1^+}^1 + t_{0,1} = 0 + 2 = 2$, la date d'arrivée du véhicule T_1 sur le nœud destination est mise à jour. Cette date est égale à la date $B_{1,1^+}^1 = 0$ de départ du véhicule T_1 sur le nœud origine, plus la durée $t_{0,1} = 2$ du transport entre le nœud origine et le nœud destination.
- $B_{1,1^-}^1 = A_{1,1^-}^1 = 2$, la date de départ du véhicule T_1 sur le nœud destination est mise à jour. Cette date est égale à la date $A_{1,1^1}^1 = 2$ d'arrivé du véhicule T_1 sur le nœud destination, puisqu'une fois la livraison des ressources effectuées, le véhicule peut immédiatement repartir.
- $S_{1,1^-}^1 = ma\,x\left(S_{1,1^+}^1;\ A_{1,1^-}^1\right) = ma\,x(-\infty; 2) = 2$, la date de début de l'activité A_1 modélisée par le nœud destination de l'arc numéro 1 est mise à jour. L'activité n'ayant par reçu de ressources, sa date est égale à $S_{1,1^+}^1 = -\infty$. Cependant des ressources viennent d'être transportées et celles-ci sont livrées à la date $A_{1,1^+}^1 = 2$, qui est la date d'arrivée du véhicule T_1 sur le nœud destination. L'activité peut débuter au plus tôt à la date $S_{1,1^-}^1 = 2$.
- $\varphi_{1,1^-}^{R,1} = \varphi_{1,1^+}^{R,1} - m\,in\left(\varphi_{1,1^+}^{R,1};\ C_1\right) = 3 - min(3; 3) = 0$, la quantité de ressources restant à transporter sur l'arc numéro 1 est mise à jour. Avant la propagation, il reste $\varphi_{1,1^+}^{R,1} = 3$ unités de ressources à transporter sur l'arc, or le véhicule T_1 est de capacité $C_1 = 3$. Toutes les ressources peuvent donc être transportées par le véhicule.
- $b_{1,1^-}^{R,1} = b_{1,2^+}^{R,1} - m\,in\left(\varphi_{1,1^+}^{R,1};\ C_1\right) = 3 - min(3; 3) = 0$, la quantité de ressources restant à transporter vers l'activité A_1 est mise à jour. Avant la propagation, il reste $b_{1,2^+}^{R,1} = 3$ unités de ressources à transporter vers l'activité, or le véhicule T_1 vient de transporter sur l'arc 3 unités de ressources. Toutes les ressources nécessaires à l'activité A_1 sont transportées.
- $f_{1,1^-} = ma\,x\left(f_{1,1^-};\ A_{1,1^-}^1\right) = ma\,x(0; 2) = 2$, le coût associé à la solution partielle est mis à jour. Avant la propagation, le coût de la solution est égal à $f_{1,1^-} = 0$ car tous les véhicules sont disponible à la date $t = 0$. Suite à la propagation, seul le véhicule T_1 s'est déplacé, sa date de disponibilité est alors égale à $A_{1,1^-}^1 = 2$. Le coût est alors égal à $f_{1,1^-} = A_{1,1^-}^1 = 2$ comme étant la date à laquelle tous les véhicules sont disponibles.

De plus, la date de début au plus tôt de toutes les activités succédant l'activité A_1 doit être mise à jour. Or d'après le Tableau 5-1, seul l'activité A_4 succède à l'activité A_1, par conséquent : $S_{1,1^-}^4 = S_{1,1^-}^1 + p_1 = 2 + 2 = 4$

La même méthode est utilisée pour créer le label $L_2^{1^-}$ à partir du label $L_1^{1^+}$ sauf que cette fois, le véhicule choisi pour effectuer l'opération de transport est le véhicule T_2 de capacité $C_2 = 2$. On a alors une unité de ressource qui ne peut être transporté. L'état des ressources est par conséquent différent entre les deux labels.

Les deux labels $L_1^{1^-}$ et $L_2^{1^-}$ créés lors de cette première itération sont incomparables, puisque les véhicules ne sont pas localisés sur les mêmes activités, ils sont donc non dominés et stockés sur le nœud numéro 1^-. Les solutions partielles associées à ces deux labels sont représentées sur la Figure 5-10. La pile est mise à jour avec le nœud 1^- et égale à $\Lambda = [2^+, 1^-]$.

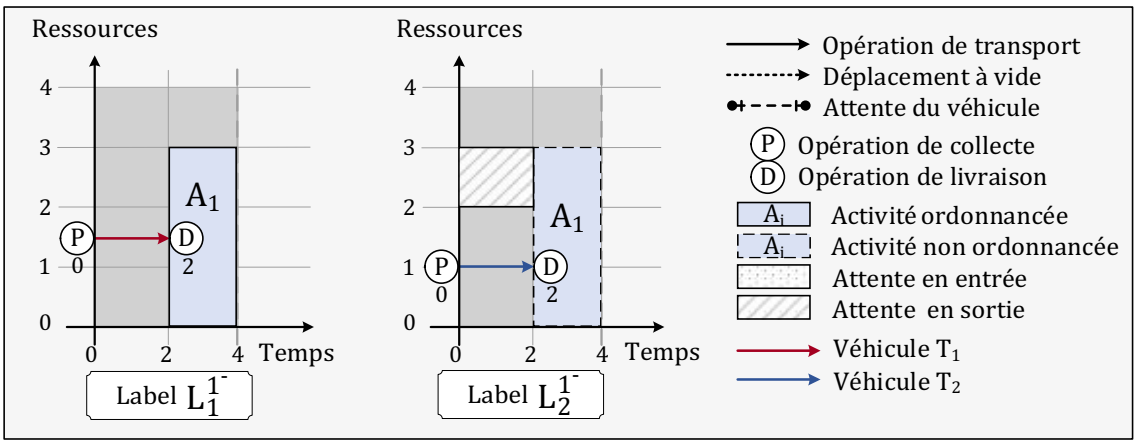


Figure 5-10. Solutions partielles associées aux labels $L_1^{1^-}$ et $L_2^{1^-}$.

La solution partielle modélisée par le label $L_1^{1^-}$ est représentée sur la Figure 5-10. Cette solution est caractérisée par une opération de transport réalisée par le véhicule T_1 entre les dates 0 et 2, permettant le transport de trois unités de ressources entre les activités A_0 et A_1. L'activité A_1.peut alors être ordonnancée à la date 2, $S_1 = 2$. La solution modélisée par le label $L_2^{1^-}$ est également représentée sur la Figure 5-10. L'opération de transport est effectuée par le véhicule T_2 entre les dates 0 et 2, permettant le transport de deux unités de ressources entre les activités A_0 et A_1. L'activité A_1.ne peut alors par être ordonnancée puisque celle-ci nécessite trois unités de ressources. La date de début de l'activité pour les solutions générées à partir de ce label sera supérieure ou égale à $t = 2$, $(S_1 \geq 2)$.

La deuxième itération de l'algorithme consiste à dépiler le nœud numéro 1^- de la pile. Tous les labels stockés sur le nœud sont alors propagés vers les nœuds numéro $1^+, 2^+, 3^+, 4^+$ en vérifiant la faisabilité de la propagation.

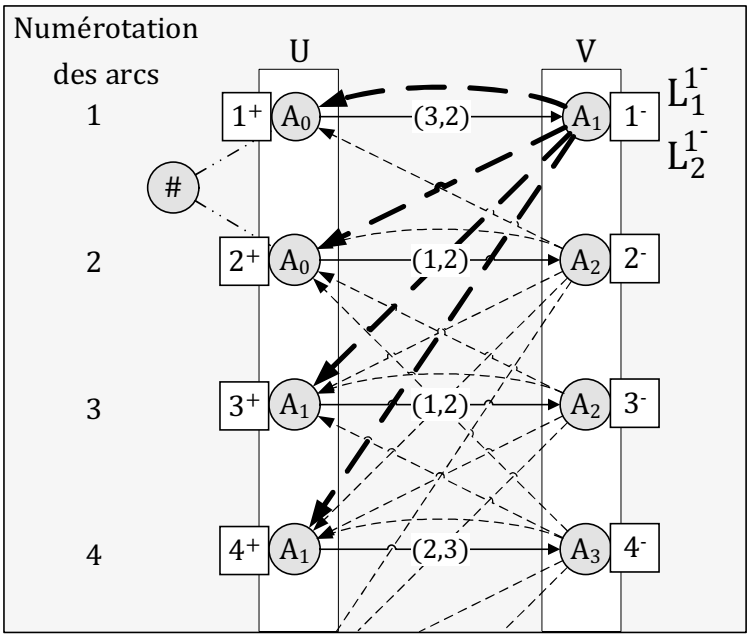


Figure 5-11. Représentation des deux labels à propager sur les arcs en gras lors de la deuxième itération

Cette étape consiste à propager les labels sur tous les arcs non requis, correspondant à des déplacements à vide. Comme énoncé précédemment, tous les déplacements à vide ne sont pas possibles, ce qui empêche la propagation de certains labels vers certains nœuds. Deux raisons font que la propagation le long d'un arc non requis $(j^- ; k^+)$ est impossssible :

- les ressources ont déjà été transportées sur l'arc requis $(k^+ ; k^-)$;
- l'activité liée au nœud k^+ n'a pas encore reçu toutes ces ressources.

Ces deux points sont illustrés lors de la deuxième itération où 10 labels sont générés alors que 16 labels peuvent être construits (propagation des deux labels vers les quatre destinations et les deux véhicules, $2 \times 4 \times 2 = 16$ labels). Six labels ne sont donc pas générés au cours de cette itération, certains labels ne peuvent pas être propagés vers certains nœuds :

- le label $L_1^{1^-}$ ne peut être propagé vers le nœud numéro 1^+ puisque toutes les ressources ont déjà été transportées sur l'arc requis, ce qui empêche la génération de deux labels ;
- le label $L_2^{1^-}$ ne peut être propagé vers les nœuds numéro 3^+ et 4^+ puisque l'activité A_1 n'a pas encore reçu toutes ses ressources, ce qui empêche la génération de quatre labels.

Les dix labels générés sont les suivants :

- Deux labels sur le nœud numéro 1^+:
  $L_1^{1^+} = (2|0,1,0,2,0,2|0,2,-\infty,-\infty,-\infty|1,1,1,2,2,2|0,1,2,2,4)$
  $L_2^{1^+} = (4|0,0,0,4,0,4|0,2,-\infty,-\infty,-\infty|1,1,1,2,2,2|0,1,2,2,4)$
- Quatre labels sur le nœud numéro 2^+:
  $L_1^{2^+} = (4|0,0,4,0,4,0|0,2,-\infty,-\infty,4|0,1,1,2,2,2|0,0,2,2,4)$
  $L_2^{2^+} = (2|1,0,2,0,2,0|0,2,-\infty,-\infty,4|0,1,1,2,2,2|0,0,2,2,4)$
  $L_3^{2^+} = (2|0,1,0,2,0,2|0,2,-\infty,-\infty,4|1,1,1,2,2,2|0,1,2,2,4)$

$L_4^{2^+} = (4|0,0,0,4,0,4|0,2,-\infty,-\infty,4|1,1,1,2,2,2|0,1,2,2,4)$

- Deux labels sur le nœud numéro 3^+:
  $L_1^{3^+} = (4|1,0,2,0,4,0|0,2,-\infty,-\infty,4|0,1,1,2,2,2|0,0,2,2,4)$
  $L_2^{3^+} = (4|1,1,2,2,2,4|0,2,-\infty,-\infty,4|0,1,1,2,2,2|0,0,2,2,4)$
- Deux labels sur le nœud numéro 4^+:
  $L_1^{4^+} = L_1^{3^+}$ et $L_2^{4^+} = L_2^{3^+}$.

Tous ces labels sont incomparables puisque les véhicules ne sont pas localisés sur les mêmes activités, ils sont donc non dominés et stockés sur les différents nœuds. La pile est mise à jour et égale à $\Lambda = [1^+, 3^+, 4^+, 2^+]$.

La troisième itération de l'algorithme permet de dépiler le nœud numéro 2^+ de la pile. Tous les labels stockés sur le nœud (au nombre de cinq) sont alors propagés vers le nœud numéro 2^-. Cette étape permet la création et l'insertion de six labels correspondant à l'affectation du véhicule T_1 ou du véhicule T_2 :

- $L_1^{2^-} = (2|2,0,2,0,2,0|0,-\infty,2,-\infty,12|3,0,1,2,2,2|3,1,1,2,4)$
- $L_2^{2^-} = (2|0,2,0,2,0,2|0,-\infty,2,-\infty,12|3,0,1,2,2,2|3,1,1,2,4)$
- $L_3^{2^-} = (6|2,0,6,0,6,0|0,2,6,-\infty,16|0,0,1,2,2,2|0,1,1,2,4)$
- $L_4^{2^-} = (2|1,2,2,2,2,2|0,2,2,-\infty,12|0,0,1,2,2,2|0,1,1,2,4)$
- $L_5^{2^-} = (2|2,1,2,2,2,2|0,2,2,-\infty,12|1,0,1,2,2,2|0,1,1,2,4)$
- $L_6^{2^-} = (4|0,2,0,6,0,6|0,2,6,-\infty,16|1,0,1,2,2,2|0,1,1,2,4)$

Tous ces labels sont incomparables entre eux puisque, même si, pour certains les véhicules sont localisés sur les mêmes activités, l'état du système et des ressources ne permet pas de conclure sur la dominance d'un label par rapport à l'autre. Tous ces labels sont donc non dominés et stockés sur le nœud numéro 2^-. La pile est mise à jour et égale à $\Lambda = [1^+, 3^+, 4^+, 2^-]$. La numérotation des labels correspond à l'ordre dans lequel ils sont générés au cours de l'algorithme, cependant, les labels sont triés sur chaque nœud en fonction de la durée nécessaire au transport de toutes les ressources restantes. On peut remarquer que l'état des ressources entre les six labels ne diffère que pour $\varphi_{.,2^-}^{R,1}$. L'état des ressources le plus avancé et donc nécessitant un temps de transport restant inférieur correspond aux labels $L_3^{2^-}$ et $L_4^{2^-}$, les autres labels sont jugés équivalent puisque le véhicule T_1 peut transporter une ou trois unités de ressource en un seul transport. L'ordre réel des labels sur le nœud à l'issue de cette itération est : $L_3^{2^-}$, $L_4^{2^-}$, $L_1^{2^-}$, $L_2^{2^-}$, $L_5^{2^-}$, $L_6^{2^-}$. Les solutions partielles associées à ces six labels sont représentées sur la Figure 5-12.

Pour les solutions partielles modélisées par les labels $L_1^{2^-}$ et $L_2^{2^-}$ représentées sur le haut de la Figure 5-12, seule l'affectation du véhicule diffère. Ces solutions sont caractérisées par une opération de transport réalisée par le véhicule T_1 ou T_2 entre les dates 0 et 2, permettant le transport d'une unité de ressources entre les activités A_0 et A_2. Cette quantité de ressources est imposée par la définition du flot. Dans ces deux solutions partielles, l'activité A_2.ne peut alors par être ordonnancée puisque celle-ci nécessite deux unités de ressources alors qu'une seule est livrée.

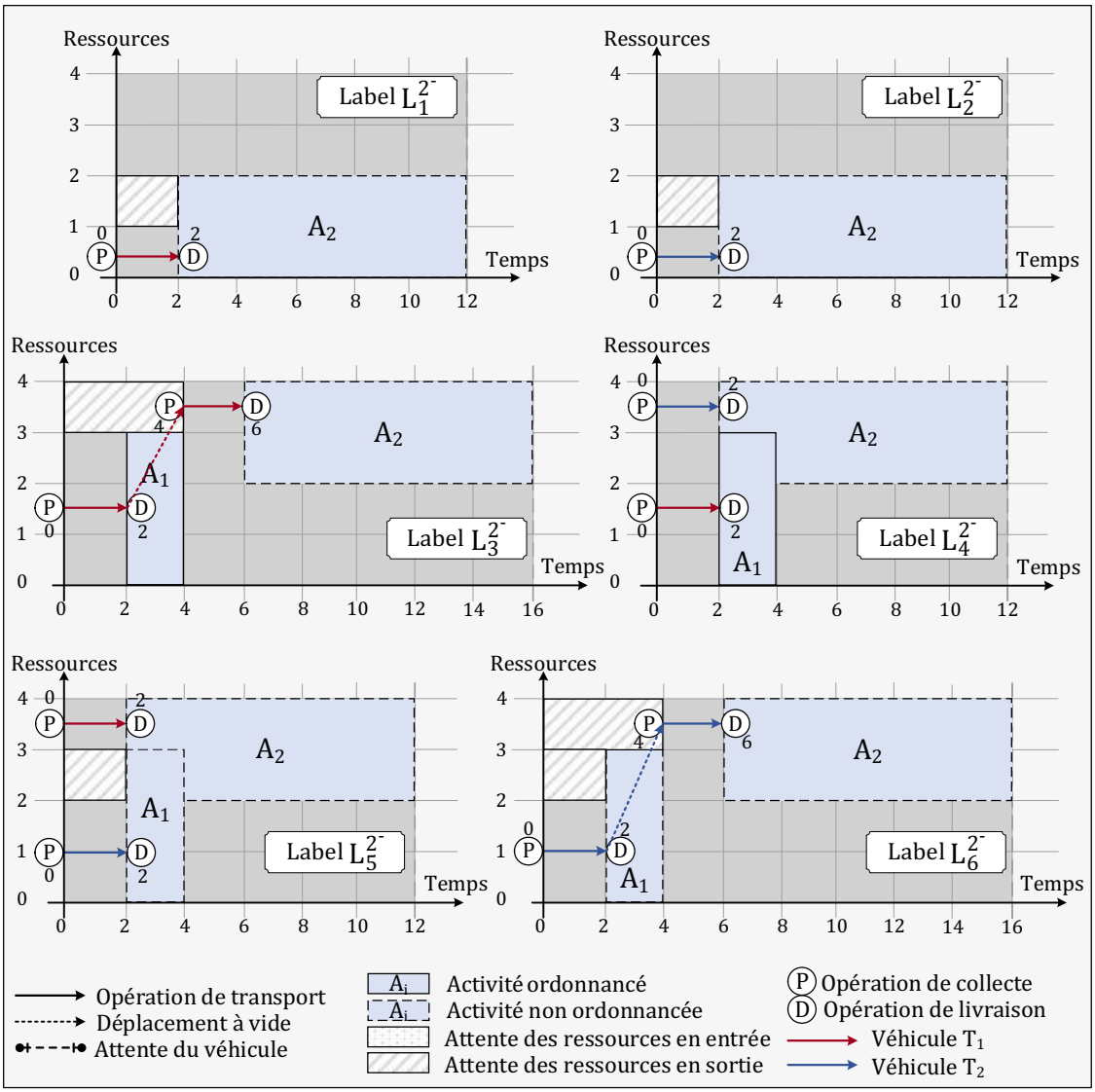


Figure 5-12. Solutions partielles associées aux labels L_1^{1-} et L_2^{1-}.

Pour les solutions partielles modélisées par les labels L_3^{2-} et L_4^{2-} représentées sur le milieu de la Figure 5-12, l'affectation du véhicule sur l'opération de transport entre les activités A_0 et A_2 diffère, ce qui implique des différences sur l'état des activités. Dans ces deux solutions partielles, c'est le véhicule T_1 qui effectue la transport entre les activités A_0 et A_1. Lorsque l'opération de transport entre A_0 et A_2 est effectuée par le véhicule T_1, et un déplacement à vide doit être pris en compte. Cette deuxième opération de transport peut donc avoir lieu entre les dates 4 et 6, permettant de définir la date d'ordonnancement de l'activité A_2 telle que $S_2 \geq 6$. Dans le deuxième cas, l'opération de transport entre A_0 et A_2 est effectuée par le véhicule T_2. Cette deuxième opération de transport peut donc avoir lieu dès la date 0 puisque le véhicule n'a pas encore effectué d'opération de transport. La date de début de l'activité A_2 pour les solutions générées à partir de ce label est supérieure ou égale à $t = 2$, $(S_2 \geq 2)$.

Enfin, pour les solutions partielles modélisées par les labels L_5^{2-} et L_6^{2-} représentées sur le bas de la Figure 5-12, c'est le véhicule T_2 qui effectue la première opération de transport. Cette différence engendre par rapport aux labels L_3^{2-} et L_4^{2-} un état des

ressources différent puisque les véhicules n'ont pas la même capacité. Dans ces deux solutions, l'activité A_1 ne reçoit pas la totalité de ces ressources.

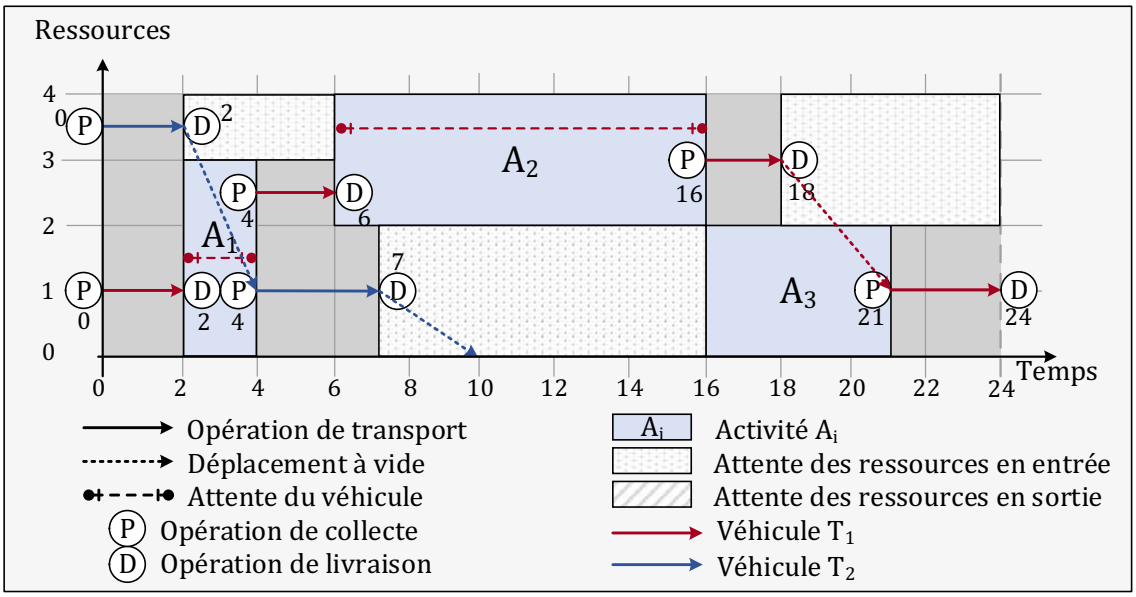


Figure 5-13. Solution optimale du RCPSPR en respectant le flot de l'exemple

À la fin de l'algorithme, lorsque tous les labels sont propagés (la pile est vide), un ensemble de labels non dominés est obtenu sur le nœud final $*$. Une des solutions minimisant le makespan peut alors être considérée comme solution optimale du RCPSPR respectant le flot défini préalablement. Une solution optimale est représentée sur la Figure 5-13. Cette solution est constituée de quatre opérations de transport pour le véhicule T_1 et de deux opérations de transport pour le véhicule T_2.

Les labels permettant la génération de cette solution sont explicités dans le Tableau 5-3. On peut vérifier que le label obtenu sur le nœud $*$ est conforme à la solution représentée sur la Figure 5-13.

Tableau 5-3
Détail des labels menant à la solution présentée dans la Figure 5-13

Noeud	Liste des labels
#	$(0\|0,0,0,0,0,0\|0,-\infty,-\infty,-\infty,-\infty\|3,1,1,2,2,2\|0,3,2,2,4)$
1^+	$(0\|0,0,0,0,0,0\|0,-\infty,-\infty,-\infty,-\infty\|3,1,1,2,2,2\|0,3,2,2,4)$
1^-	$(2\|1,0,2,0,2,0\|0,2,-\infty,-\infty,4\|0,1,1,2,2,2\|0,0,2,2,4)$
2^+	$(2\|1,0,2,0,2,0\|0,2,-\infty,-\infty,4\|0,1,1,2,2,2\|0,0,2,2,4)$
2^-	$(2\|1,2,2,2,2,2\|0,2,2,-\infty,12\|0,0,1,2,2,2\|0,0,1,2,4)$
3^+	$(7\|1,3,2,7,4,7\|0,2,2,7,12\|0,0,1,0,2,2\|0,0,1,0,4)$
3^-	$(7\|2,3,6,7,6,7\|0,2,6,16,16\|0,0,0,0,2,2\|0,0,0,0,4)$
4^+	$(4\|1,1,2,4,2,4\|0,2,2,-\infty,12\|0,0,1,2,2,2\|0,0,1,2,4)$
4^-	$(7\|1,3,2,7,2,7\|0,2,2,7,12\|0,0,1,0,2,2\|0,0,1,0,4)$
5^+	$(16\|2,3,6,7,16,7\|0,2,6,16,16\|0,0,0,0,2,2\|0,0,0,0,4)$
5^-	$(18\|*,3,18,7,18,7\|0,2,6,16,18\|0,0,0,0,0,2\|0,0,0,0,2)$
6^+	$(21\|3,3,20,7,21,7\|0,2,6,16,18\|0,0,0,0,0,2\|0,0,0,0,2)$
6^-	$(24\|*,3,24,7,24,7\|0,2,6,16,24\|0,0,0,0,0,0\|0,0,0,0,0)$
*	$(24\|*,*,24,10,24,10\|0,2,6,16,24\|0,0,0,0,0,0\|0,0,0,0,0)$

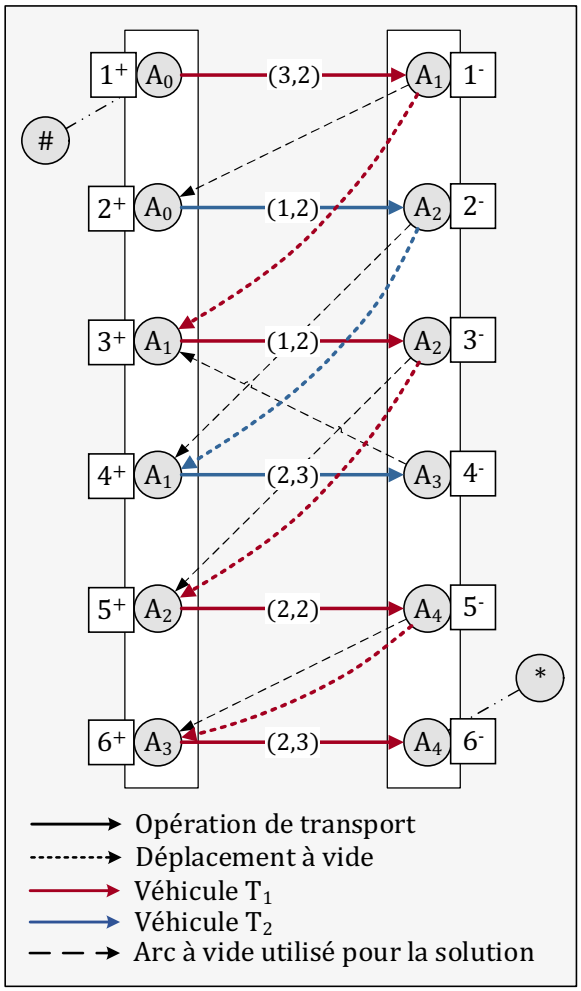


Figure 5-14. Graphe auxiliaire représentant le plus court chemin ainsi que l'affectation des véhicules permettant de modéliser les tournées

Cette solution correspond à un plus court chemin dans le graphe auxiliaire, dans lequel certains arcs sont affectés au véhicule T_1 et d'autres au véhicule T_2 afin de définir les tournées des deux véhicules. Ce plus court chemin est représenté sur la Figure 5-14, faisant apparaître la tournée des deux véhicules.

Les arcs non requis modélisés dans le graphe auxiliaire ne correspondent pas nécessairement à des déplacements à vide d'un véhicule dans la solution finale. Par exemple, l'arc entre le nœud 1^- et 2^+ ne correspond pas à un déplacement à vide d'un véhicule entre l'activité A_1 et A_0 mais à un déplacement à vide du véhicule T_2 de A_0 vers A_0 (pas de déplacement concret) puisque le véhicule T_2 est resté sur l'activité A_0 pendant le déplacement du véhicule T_1.

5.3.5 Discussion

L'algorithme proposé dans ce chapitre, pour résoudre le RCPSPR à partir d'un flot, repose sur la définition d'un plus court chemin à contraintes de ressources dans un graphe auxiliaire spécialement défini pour le problème du RCPSPR.

Cet algorithme permet la résolution d'un problème de tournées sur arcs avec des caractéristiques spécifiques dues à la nature du problème. En effet, ce problème intègre à la fois des contraintes liées au RCPSP et des contraintes liées au transport qui sont toutes deux liées par la gestion de ressources partagées. Dans le problème, après définition d'un flot, la quantité à transporter sur les arcs peut dépasser la capacité des véhicules, le problème appartient donc aux problèmes de type « split delivery ». À notre connaissance, il n'existe pas d'algorithme, sur les problèmes de tournées sur arc, permettant de répondre à l'ensemble de ces contraintes avec des temps de calcul raisonnables.

L'algorithme proposé s'intéresse à la minimisation du makespan, qui est le critère le plus couramment utilisé en ordonnancement ou en transport (on parle de coût total de la solution en termes de distance ou de temps). Il n'y a, a priori, pas de raison pour que des solutions de même makespan possèdent des caractéristiques équivalentes en mesurant d'autres critères liés au transport et à l'ordonnancement. Pour le RCPSPR on pourrait introduire des critères liés aux durées de séjour dans les stocks pour les ressources (ordonnancement), ou aux temps de transport ou encore aux temps d'attente des véhicules (transport).

Sur l'exemple de la section précédente, on peut comparer la solution représentée sur la Figure 5-13, avec la solution optimale pour le RCPSPR présentée sur la Figure 5-4 au début de ce chapitre. Ces deux solutions ont un makespan égal à $C_{max} = S_4 = 24$. On peut donc en déduire que la solution obtenue à l'issue du plus court chemin est une solution optimale du RCPSPR. De plus, ces deux solutions possèdent le même ordonnancement des activités, dans les deux cas, les dates sont identiques, on a : $S_1 = 2$, $S_2 = 6$ et $S_3 = 16$. Néanmoins, on peut remarquer que des critères (liés au transport et à l'ordonnancement) diffèrent entre ces deux solutions, comme par exemple, les temps de transport, les temps d'attente des véhicules et les temps d'attente des ressources. Le Tableau 5-4 permet de comparer ces deux solutions selon plusieurs critères.

Tableau 5-4
Comparaison de deux solutions optimales

	Solution Optimale CPLEX	Solution Optimale PCC
Temps de transport du véhicule T_1	6	12
Temps de transport du véhicule T_2	12	10
Temps total de transport	18	22
Temps d'attente du véhicule T_1	10	12
Temps d'attente du véhicule T_2	12	0
Temps d'attente total des véhicules	22	12
Temps d'attente des ressources en entrée	26	34
Temps d'attente des ressources en sortie	2	0
Temps total d'attente des ressources	28	34

Dans le Tableau 5-4, les temps d'attente des ressources sont exprimés par unité de temps et par unité de ressources. On peut noter, que la solution obtenue avec CPLEX possède un temps de transport total (égal à 18) inférieur à la solution obtenue avec l'algorithme de plus court chemin (PCC) avec un temps égal à 22. De même, les temps d'attente des ressources, en entrée et en sortie, valent 26 et 2 pour la solution avec CPLEX, et 34 et 0 pour la solution avec le PCC.

Néanmoins, le temps d'attente total des véhicules est inférieur pour la solution obtenue avec l'algorithme de plus court chemin. En effet, le temps d'attente total des véhicules vaut 22 dans la solution fournie par CPLEX, et 12 dans la solution fournie par l'algorithme de plus court chemin. Cependant, en s'intéressant à la notion d'équité, on peut noter que la solution obtenue avec CPLEX est plus équitable sur ce critère avec des temps d'attente proche pour les deux véhicules (égaux à 10 et 12), contrairement à la solution obtenue avec le PCC (égaux à 12 et 0). Il peut être intéressant de noter qu'une approche multi-objectifs pourrait être envisagée pour résoudre ce problème, afin de

minimiser plusieurs points cités dans le Tableau 5-4. Le temps d'attente des ressources peut être traduit par un coût de stockage, puisque les ressources doivent être stockées sur les activités.

5.4 Résultats numériques

Les expérimentations numériques sont réalisées sur les mêmes jeux d'instances que ceux présentés dans le chapitre 4 :

- le premier jeu d'instance regroupe 18 instances de petite taille, *i.e.*, avec 6 activités non fictives, pour lesquelles une résolution optimale utilisant CPLEX est possible ;
- le second jeu d'instance regroupe 9 instances de taille moyenne, *i.e.*, avec 30 activités non fictives, ces instances ne sont pas solvable par CPLEX.

Ces jeux d'instances sont disponibles sur la page web suivante :
http://fc.isima.fr/~vinot/Research/RCPSP&Routing.html

En raison de la nature combinatoire du problème de RCPSP, l'obtention de solutions optimales en utilisant des méthodes exactes pour des instances avec plus de 60 activités devient impossible (Valls *et al.*, 2005). Le RCPSPR étant une extension du RCPSP avec la gestion d'une flotte de véhicules, les variables de décision du problème englobent à la fois les variables RCPSP et les variables du problème de tournées de véhicules. Pour les instances de taille moyenne, avec plus de 60 opérations qui doivent être ordonnancées ($n_{tot} = 61$), la résolution optimale de cet ensemble d'instances est difficile à obtenir en raison, en premier lieu, du grand nombre d'opérations à planifier, avec à la fois les activités et les opérations de transport et, en deuxième lieu de la coordination entre les activités et les opérations de transport, avec l'affectation des véhicules aux opérations de transport.

Toutes les expérimentations numériques sont réalisées avec un programme C en utilisant Visual Studio et Windows 7 comme système d'exploitation sur un Dell Optiflex9020 avec un processeur Intel Core i7-4770 CPU 3.4 GHz et 16 Gb de RAM. Le nombre de Mflop/s de cet ordinateur peut être estimé à 2671 Mflops (Dongarra, 2014).

5.4.1 Les performances de l'algorithme de plus court chemin à contraintes de ressources

L'efficacité de l'algorithme de plus court chemin à contraintes de ressources peut-être évaluée sur les instances de petite taille, en considérant une restriction des ensembles Y_i à $N_L = 500$ labels avec $r = 10$ réplications par instance. Une réplication correspond à la résolution d'un flot aléatoirement généré. Le premier flot généré permet l'obtention d'une solution optimale pour le RCPSP, il est obtenu avec CPLEX. L'algorithme de plus court chemin à contraintes de ressources est comparé à la résolution optimale du sous-problème avec IBM ILOG CPLEX 12.6, en fixant le flot dans la formulation linéaire énoncée dans le chapitre précédent.

Pour chaque instance x, $\overline{h_C(x,r)}$ définit le coût moyen des solutions obtenues avec CPLEX sur les $r = 10$ réplications/flots, et $\overline{t_C(x,r)}$ définit le temps CPU moyen pour

obtenir les solutions optimales avec CPLEX. De la même façon, $\overline{h_{SP}(x,r)}$ définit le coût moyen des solutions obtenues avec l'algorithme de plus court chemin, et $\overline{t_{SP}(x,r)}$ définit le temps CPU moyen pour obtenir ces mêmes solutions. La notation $\overline{g(x,n)}$ permet d'exprimer l'écart entre $\overline{h_{SP}(x,r)}$ et $\overline{h_C(x,r)}$ pour l'instance x et les r réplications. Le nombre de solutions optimales obtenues par l'algorithme du plus court chemin sur l'instance x et les r réplications est noté $n_{SP}(x,r)$.

Notons que $\overline{g(.,r)}$ permet d'exprimer l'écart moyen entre le coût des solutions obtenues avec CPLEX et l'algorithme proposé sur toutes les instances. De même $\overline{t_*(.,r)}$ représente le temps CPU moyen sur l'ensemble des instances pour les deux méthodes (CPLEX et l'algorithme de plus court chemin). Le nombre moyen de solutions optimales trouvées sur les 18 instances est noté $\overline{n_{SP}(.,r)}$, et $n_{SP}(.,.) = \sum_{x=1}^{x=18} n_{SP}(x,r)$, représente le nombre total de solutions optimales trouvées sur les 18 instances avec les 10 réplications.

Tableau 5-5
Efficacité de l'algorithme avec une restriction sur le nombre de labels ($N_L = 500$) pour les instances de petite taille

	CPLEX Résolution optimale		Algorithme de plus court chemin à contraintes de ressources			
Instances	$\overline{h_C(x,r)}$	$\overline{t_C(x,r)}$	$\overline{h_{SP}(x,r)}$	$n_{SP}(x,r)$	$\overline{t_{SP}(x,r)}$	$\overline{g(x,r)}$
LMQV_U1	101.5	46.6	101.5	10/10	10.6	0.00 %
LMQV_U2	198.5	37.2	198.7	9/10	10.2	0.09 %
LMQV_U3	246.6	29.0	246.6	10/10	11.9	0.00 %
LMQV_U4	123.3	31.0	123.3	10/10	3.6	0.00 %
LMQV_U5	247.1	32.3	247.1	10/10	4.3	0.00 %
LMQV_U6	288.1	34.5	288.1	10/10	7.0	0.00 %
LMQV_U7	128.4	9.9	128.4	10/10	2.2	0.00 %
LMQV_U8	261.1	9.3	261.1	10/10	2.4	0.00 %
LMQV_U9	276.1	9.2	276.1	10/10	2.5	0.00 %
LMQV_C1	78.6	11.1	78.6	10/10	0.0	0.00 %
LMQV_C2	135.6	9.0	135.6	10/10	0.0	0.00 %
LMQV_C3	160.6	10.0	160.6	10/10	0.0	0.00 %
LMQV_C4	45.2	10.5	45.3	9/10	21.1	0.20 %
LMQV_C5	92.0	12.2	92.3	9/10	25.8	0.28 %
LMQV_C6	94.6	12.2	94.8	9/10	24.6	0.19 %
LMQV_C7	66.4	5.6	66.4	10/10	6.3	0.00 %
LMQV_C8	132.6	5.2	132.6	10/10	10.2	0.00 %
LMQV_C9	136.7	5.7	136.7	10/10	5.2	0.00 %
$\overline{g(.,r)}$						0.04 %
$\overline{t_*(.,r)}$		17.8			8.2	
$\overline{n_{SP}(.,r)}$				9.99		
$n_{SP}(.,.)$				176/180		

Le temps CPU moyen sur toutes les instances pour l'algorithme de plus court chemin est égal à 8.2 secondes, plus de deux fois plus rapide que CPLEX qui demande environ 17 secondes (Tableau 5-5). Le nombre moyen de solutions optimales retrouvées est supérieur à 99%. Le nombre total de solutions optimales retrouvées est égal à 176 sur les 180 problèmes correspondant aux 18 instances × 10 réplications avec un écart de 0.04% par rapport aux solutions optimales.

Le Tableau 5-6 donne le détail des solutions obtenues pour l'instance LMQV_U1 pour les 10 flots générés. Le premier flot indiqué dans le tableau permet l'obtention d'une solution optimale pour le problème du RCPSP avec un makespan égal à 19. Les flots suivants sont générés aléatoirement puis classés par ordre croissant sur le makespan de la solution du RCPSP (entre 19 et 24). Pour cette instance et sur les 10 flots générés, l'algorithme de plus court chemin est plus efficace que l'approche de résolution avec CPLEX puisque toutes les solutions optimales sont obtenues (les écarts sont tous nuls) et l'algorithme est quatre fois plus rapide que CPLEX qui nécessite 46.6 secondes en moyenne contre 10.6 secondes en moyenne pour l'algorithme de programmation dynamique.

D'autre part, dans le Tableau 5-6 on peut noter que la meilleure solution du RCPSPR a un coût égal à 87 et a été obtenue avec un flot donnant une solution du RCPSP avec un coût égal à 20 (flot numéro 3). De tels résultats mettent en évidence le fait que des flots permettant de modéliser des bonnes solutions pour le RCPSP n'engendrent pas nécessairement de bonnes solutions pour le RCPSPR. En effet, il n'y a aucune raison qu'une bonne solution pour un sous-problème permette l'obtention d'une bonne solution pour le problème dans sa globalité. De plus, plusieurs flots peuvent donner le même makespan pour le problème du RCPSP (flot 7 et 8 de makespan 23) mais peuvent générer des solutions du RCPSPR très différentes avec un makespan variant de 86 à 105. Toutes ces remarques mettent en avant le fait qu'il est difficile d'établir un lien entre les bonnes solutions du RCPSP et celle du RCPSPR.

Tableau 5-6
Comparaison des solutions du RCPSP ou du RCPSPR pour 10 flots aléatoires.

		Résolution du RCPSPR à partir d'un flot				
		Résolution avec CPLEX		Résolution avec l'algorithme de plus court chemin		
Flot	Solution du RCPSP	$\overline{h_C(x,1)}$	$\overline{t_C(x,1)}$	$\overline{h_{SP}(x,1)}$	$\overline{t_{SP}(x,1)}$	$\overline{h_C(x,1)}$
1	19*	95	20.4	95	22.5	0.0 %
2	19	115	97.1	115	25.0	0.0 %
3	**20**	**87**	20.3	**87**	0.1	0.0 %
4	20	117	115.7	117	22.3	0.0 %
5	21	96	22.8	96	1.3	0.0 %
6	22	99	27.7	99	5.8	0.0 %
7	23	86	26.1	86	5.2	0.0 %
8	23	105	40.7	105	1.5	0.0 %
9	24	102	30.5	102	1.0	0.0 %
10	24	113	65.2	113	23.4	0.0 %
Avg.		101.5	46.6	101.5	10.6	0.0 %

Le Tableau 5-7 donne les résultats obtenus par l'algorithme de plus court chemin à contraintes de ressources avec une restriction sur le nombre de labels tel que $N_L = 50$, pour chaque instance de taille moyenne avec un flot ($r = 1$). Le nombre d'arcs dans le graphe flot, pour chaque flot, est indiqué dans la deuxième colonne du Tableau 5-7 avec la notation n_φ. Malheureusement ces solutions ne peuvent pas être comparées à celles obtenues avec CPLEX. En effet, pour ces instances composées de 30 activités, le modèle linéaire ne peut être résolu par CPLEX, dû à la fois, au grand nombre de variables et de

contraintes. Par exemple, pour l'instance LMQV_J30_U1, aucune solution n'est fournie par CPLEX après deux jours d'exécution.

Connaissant le nombre d'arcs portant un flot générés pour chaque flot/instance, un nombre minimal d'opérations de transport peut être déduit et est égal à n_φ. Par exemple, pour l'instance LMQV_J30_CC1, 30 activités doivent être ordonnancées avec en plus, au moins, 60 opérations de transport, ce qui représente plus de 90 opérations à ordonnancer.

Tableau 5-7
Solution obtenues pour les instances de taille moyenne avec un flot ($r = 1$)

			Résolution avec l'algorithme de plus court chemin (N_L=50)	
Instance	n_φ	Solution du RCPSP	$\overline{h_{SP}(x,1)}$	$\overline{t_{SP}(x,1)}$
LMQV_J30_U1	56	96	250	80.4
LMQV_J30_U2	57	105	276	697.0
LMQV_J30_U3	50	82	129	5.3
LMQV_J30_C1	53	113	221	729.0
LMQV_J30_C2	56	81	231	336.9
LMQV_J30_C3	57	83	139	32.9
LMQV_J30_CC1	60	103	271	270.3
LMQV_J30_CC2	58	108	270	305.6
LMQV_J30_CC3	58	100	158	89.9
$\overline{t_*(.,n)}$				283.0

5.4.2 Les performances des règles de gestion des labels

Afin d'évaluer de façon plus poussée l'efficacité de l'algorithme, aucune restriction sur le nombre de labels n'est effectuée dans les tests reportés dans cette sous-partie. Tous les labels sont stockés sur les nœuds du graphe auxiliaire et le Tableau 5-8 permet de rendre compte :

- du nombre de labels insérés sur les nœuds avec la fonction `InsertLabel()`
- du nombre de labels supprimés par borne supérieure par la fonction `CheckUbLb()`
- du nombre de labels supprimés par dominance par la fonction `ApplyDominance()`

Les expérimentations suivantes sont effectuées sur les instances de petite taille en considérant les 10 réplications précédemment utilisées. Les notations suivantes sont utilisées dans le Tableau 5-8.

- $TNLG(x,j)$ nombre de labels générés pendant la résolution du flot numéro j;
- $\overline{TNLG(x,r)}$ nombre moyen de label générés sur les r réplications ;
- $\overline{TNLP(x,r)}$ nombre moyen de label supprimés sur les r réplications ;
- $P_p(x,j)$ pourcentage de label supprimés par borne supérieure par rapport au nombre de label générés ;
- $\overline{P_p(x,r)}$ pourcentage moyen de label supprimés par borne supérieure $\overline{P_p(x,r)} = Avg_{j=1,r}\left(P_p(x,j)\right)$;
- $\overline{TNLD(x,r)}$ nombre moyen de label supprimés par dominance ;
- $P_d(x,j)$ pourcentage de labels supprimés par borne supérieure par rapport

au nombre de label générés ;

- $\overline{P_d(x,r)}$ pourcentage moyen de labels supprimés par dominance, avec $\overline{P_p(x,r)} = Avg_{j=1,r}\left(P_p(x,j)\right)$;
- $\overline{P(.,.)}$ moyenne des pourcentages moyen ($\overline{P_p(x,r)}$ où $\overline{P_d(x,r)}$) sur toutes les instances ;
- $\overline{Avg(.,.)}$ nombre moyen de label générés, supprimés par borne supérieure ou par dominance.

Tableau 5-8
Efficacité des règles de gestion des labels sur les petites instances

Instance	$\overline{TNLG(x,r)}$	$\overline{TNLP(x,r)}$	$\overline{TNLD(x,r)}$	$\overline{P_P(x,r)}$	$\overline{P_d(x,r)}$
LMQV_U1	256 514.9	49 313.8	114 201.4	19.2	44.5
LMQV_U2	245 631.8	47 122.1	111 452.6	19.2	45.4
LMQV_U3	298 695.6	57 901.4	136 189.4	19.4	45.6
LMQV_U4	124 521.7	21 261.5	64 856.7	17.1	52.1
LMQV_U5	134 996.0	22 109.0	73 049.9	16.4	54.1
LMQV_U6	198 907.8	28 581.2	112 077.0	14.4	56.3
LMQV_U7	177 200.4	40 449.2	78 194.4	22.8	44.1
LMQV_U8	189 100.6	42 760.4	84 848.8	22.6	44.9
LMQV_U9	217 710.7	51 939.6	92 055.1	23.9	42.3
LMQV_C1	4.4	2.8	0.0	63.6	0.0
LMQV_C2	4.4	2.8	0.0	63.6	0.0
LMQV_C3	4.4	2.8	0.0	63.6	0.0
LMQV_C4	543 915.1	101 480.3	267 705.7	18.7	49.2
LMQV_C5	589 227.4	107 571.8	281 184.5	18.3	47.7
LMQV_C6	580 233.6	107 110.3	280 699.6	18.5	48.4
LMQV_C7	230 693.0	26 232.9	167 281.8	11.4	72.5
LMQV_C8	294 602.6	35 820.0	203 370.0	12.2	69.0
LMQV_C9	273 935.0	33 845.2	187 245.9	12.4	68.4
$\overline{Avg(.,.)}$	241 994.4	42 972.6	125 255.8		
$\overline{P(.,.)}$				25.4	43.6

D'après le Tableau 5-8, on peut noter que :

- 25% des labels générés sont supprimés par borne supérieure ;
- 43% des labels générés sont supprimés par dominance ;

Ces résultats prouvent que la règle de dominance introduite est très efficace et est un point-clé dans la définition de l'algorithme de plus court chemin.

5.4.3 Comparaison des deux approches de résolution du RCPSPR à partir d'un flot

Deux approches de résolution pour le RCPSPR à partir d'un flot sont présentées dans cette thèse. L'approche présentée dans ce chapitre, pouvant être optimale ou heuristique suivant la restriction définie sur le nombre de labels par nœud, et l'approche présentée dans le chapitre 4, basée sur la résolution séquentielle de plusieurs sous-problèmes grâce à des heuristiques.

Tableau 5-9
Comparaison de l'efficacité des deux approches pour les instances de petite taille

	Approche du chapitre 4		Approche du chapitre 5 (N_L=500)		Approche du chapitre 5 (N_L=50)		Approche du chapitre 5 (N_L=10)		Gap $\overline{g(x,r)}$		
Instance	$\overline{h_C(x,r)}$	$\overline{t_C(x,r)}$	$\overline{h_{SP}(x,r)}$	$\overline{t_{SP}(x,r)}$	$\overline{h_{SP}(x,r)}$	$\overline{t_{SP}(x,r)}$	$\overline{h_{SP}(x,r)}$	$\overline{t_{SP}(x,r)}$	500	50	10
LMQV_U1	112.5	<0.01	101.5	10.6	102.10	0.22	103.50	0.02	-9.78 %	-9.24 %	-8.00 %
LMQV_U2	223.6	<0.01	198.7	10.2	199.70	0.22	202.50	0.02	-11.14 %	-10.69 %	-9.44 %
LMQV_U3	273.1	<0.01	246.6	11.9	247.90	0.26	250.00	0.02	-9.70 %	-9.23 %	-8.46 %
LMQV_U4	135.1	<0.01	123.3	3.6	124.70	0.14	126.30	0.01	-8.81 %	-7.70 %	-6.51 %
LMQV_U5	272.7	<0.01	247.1	4.3	249.30	0.12	253.20	0.02	-9.39 %	-8.58 %	-7.15 %
LMQV_U6	320.2	<0.01	288.1	7.0	294.00	0.15	295.70	0.02	-10.02 %	-8.18 %	-7.65 %
LMQV_U7	138.4	<0.01	128.4	2.2	129.00	0.41	131.10	0.03	-7.23 %	-6.79 %	-5.27 %
LMQV_U8	283.2	<0.01	261.1	2.4	262.50	0.38	265.50	0.02	-7.80 %	-7.31 %	-6.25 %
LMQV_U9	320.8	<0.01	276.1	2.5	276.10	0.41	280.50	0.03	-13.93 %	-13.93 %	-12.56 %
LMQV_C1	78.6	<0.01	78.6	0.0	78.60	0.00	78.60	0.00	0.00 %	0.00 %	0.00 %
LMQV_C2	135.6	<0.01	135.6	0.0	135.60	0.00	135.60	0.00	0.00 %	0.00 %	0.00 %
LMQV_C3	160.6	<0.01	160.6	0.0	160.60	0.00	160.60	0.00	0.00 %	0.00 %	0.00 %
LMQV_C4	48.7	<0.01	45.3	21.1	45.50	0.30	46.20	0.02	-6.98 %	-6.57 %	-5.13 %
LMQV_C5	100.5	<0.01	92.3	25.8	93.20	0.29	94.60	0.02	-8.16 %	-7.26 %	-5.87 %
LMQV_C6	103.1	<0.01	94.8	24.6	95.60	0.32	96.40	0.02	-8.05 %	-7.27 %	-6.50 %
LMQV_C7	67.5	<0.01	66.4	6.3	66.80	0.25	67.10	0.01	-1.63 %	-1.04 %	-0.59 %
LMQV_C8	135	<0.01	132.6	10.2	133.50	0.21	133.80	0.01	-1.78 %	-1.11 %	-0.89 %
LMQV_C9	138.8	<0.01	136.7	5.2	137.40	0.15	137.00	0.01	-1.51 %	-1.01 %	-1.30 %
$\overline{g(.,r)}$									-6.44 %	-5.89 %	-5.09 %
$\overline{t_*(.,r)}$		0.002		8.2		0.21		0.02			

Afin de faciliter la lecture de cette sous-partie l'approche du chapitre 4 (respectivement 5) est notée f-4-RCPSPR (resp. f-2-RCPSPR). La notation f-n-RCPSPR caractérise une approche de résolution qui associe à un flot f une solution du RCPSPR en n étapes.Ces deux approches peuvent être comparées sur les instances de petite et de moyenne taille, en reprenant les jeux de données précédents, avec dix réplications pour chaque instance de petite taille et une seule réplication pour chaque instance de taille moyenne.

Le Tableau 5-9 permet de comparer les approches des deux chapitres sur les instances de petite taille. L'approche f-2-RCPSPR est déclinée sous trois versions, en fonction du nombre de labels maximal N_L sur chaque nœud (500, 50 ou 10). Plusieurs points sont à remarquer sur ce tableau.

Tout d'abord, en s'intéressant au temps d'exécution des différentes approches, on peut noter que f-4-RCPSPR est très efficace en temps et nécessite ~0.002 seconde en moyenne pour une évaluation. L'approche f-2-RCPSPR a des temps d'exécution nettement supérieurs, cependant, on remarque que plus la restriction sur le nombre de labels maximal par nœud est forte, plus le temps d'exécution diminue, passant de 8.2 s à 0.02 s, mais restant toutefois encore supérieur au temps moyen d'exécution de f-4-RCPSPR (qui est de 0.002 s).

D'autre part, en observant le coût moyen des solutions, ainsi que les écarts entre les approches, on remarque que l'approche f-2-RCPSPR améliore les solutions de l'approche f-4-RCPSPR de 6.4%, de 5.9% ou de 5.1% suivant sa version.

Malgré le fait que l'approche f-2-RCPSPR permette l'obtention de solutions strictement meilleures, l'approche f-4-RCPSPR peut être privilégiée dans les schémas de résolution de type métaheuristique. En effet, la différence sur les temps d'exécution de ces deux approches montre que dix flots peuvent être évalués par f-4-RCPSPR pendant qu'un seul flot est évalué par f-2-RCPSPR. De plus, il a été démontré dans le chapitre précédent que l'on ne pouvait pas prédire la qualité d'une solution du RCPSPR en n'évaluant que le flot. Il est donc préférable d'explorer au maximum l'espace de codage contenant les flots possibles d'un problème de RCPSPR.

Tableau 5-10
Comparaison de l'efficacité des deux approches pour les instances de moyenne taille

		Approche du chapitre 4		Approche du chapitre 5 (N_L=50)		Approche du chapitre 5 (N_L=10)		$\overline{g(x,1)}$	
Instance	n_φ	$\overline{h_{SP}(x,1)}$	$\overline{t_{SP}(x,1)}$	$\overline{h_{SP}(x,1)}$	$\overline{t_{SP}(x,1)}$	$\overline{h_{SP}(x,1)}$	$\overline{t_{SP}(x,1)}$	50	10
LMQV_J30_U1	56	278	0.02	250	80.4	278	6.7	-10.1 %	0.0 %
LMQV_J30_U2	57	298	0.00	276	697.0	298	23.9	-7.4 %	0.0 %
LMQV_J30_U3	50	144	0.00	129	5.3	144	2.1	-10.4 %	0.0 %
LMQV_J30_C1	53	232	0.00	221	729.0	232	20.4	-4.7 %	0.0 %
LMQV_J30_C2	56	244	0.00	231	336.9	244	41.3	-5.3 %	0.0 %
LMQV_J30_C3	57	155	0.02	139	32.9	155	5.0	-10.3 %	0.0 %
LMQV_J30_CC1	60	282	0.00	271	270.3	282	6.6	-3.9 %	0.0 %
LMQV_J30_CC2	58	284	0.00	270	305.6	284	5.4	-4.9 %	0.0 %
LMQV_J30_CC3	58	162	0.00	158	89.9	162	9.6	-2.5 %	0.0 %
$\overline{t_*(.,n)}$			0.004		283.0		13.4	-6.6 %	0.0 %

Pour les instances de moyenne taille, les différences entre les deux approches sont plus nettes (Tableau 5-10). On remarque que l'approche f-4-RCPSPR est encore très efficace en temps, avec ~0.004 seconde en moyenne pour une évaluation. L'approche f-2-RCPSPR permet d'améliorer les résultats obtenus avec l'approche f-4-RCPSPR de près de 7%, néanmoins les temps de calcul sont beaucoup plus grands. Il est intéressant de rappeler que l'approche f-4-RCPSPR fournit une borne supérieur à l'approche f-2-RCPSPR. On peut alors noter que l'approche f-2-RCPSPR n'est pas en mesure de trouver une meilleur solution avec $N_L = 10$.

5.4.4 Conclusion

Les résultats confirment l'efficacité de l'algorithme et prouvent que l'algorithme peut être utilisé en pratique, en étant intégré au sein d'une métaheuristique comme recherche locale (en limitant le nombre de labels par nœud) ou en post-optimisation. L'algorithme de plus court chemin à contraintes de ressources peut être utilisé comme fonction d'évaluation afin de définir une solution du RCPSPR à partir d'un flot.

Des expérimentations numériques plus poussées ont montré que l'algorithme peut être adapté en fonction du contexte, et permet de trouver des solutions du RCPSPR de qualité dans un temps de calcul acceptable, en faisant varier le nombre de labels stockés sur les nœuds. Une expérimentation avec18 flots a révélé que le temps de calcul peut être réduit de 15 secondes avec 500 labels par nœud à 0.4 seconde avec 50 labels par nœud, fournissant la solution optimale du RCPSPR dans 67% des cas (l'écart moyen par rapport à la solution optimale étant inférieur à 0.9%).

Étant donné que la fonction d'évaluation, permettant d'associer une solution à un flot, peut faire partie d'un processus global d'optimisation, la capacité de l'algorithme à assurer une évaluation quasi optimale, dans un temps de calcul très court est une caractéristique très intéressante de l'algorithme.

5.5 Conclusion du chapitre

Ce chapitre aborde un problème intégré composé d'un problème d'ordonnancement et d'un problème de tournées de véhicules. Ce nouveau problème, nommé RCPSPR, est présenté dans le chapitre précédent et est résolu à l'aide de plusieurs fonctions d'évaluation.

L'originalité de l'approche proposée dans ce chapitre repose sur la résolution optimale d'un sous-problème du RCPSPR, dans lequel les échanges de ressources sont connus et déterminés grâce à un flot respectant les contraintes du RCPSP. Cette approche repose sur l'utilisation d'une représentation indirecte de la solution à l'aide d'un flot et d'une fonction d'évaluation permettant l'obtention d'une solution du RCPSPR à partir d'un flot. Cette approche de résolution se situe dans la continuité des approches « route-first, cluster-second » mais avec la prise en compte de contraintes liées à un problème d'ordonnancement.

La fonction d'évaluation proposée repose sur la définition d'un graphe auxiliaire, sur lequel un algorithme de plus court chemin à contraintes de ressources est appliqué. Cet algorithme utilise des labels modélisant l'ensemble du système à un instant donné, et est géré par un ensemble de règles permettant l'obtention du plus court chemin tout en limitant le nombre de labels à gérer durant l'algorithme.

Cette méthode de résolution est testée sur des instances de petite et de moyenne taille. Cette méthode a prouvé son efficacité en comparant les résultats obtenus avec les résultats obtenus par un programme linéaire résolu par CPLEX.

Une comparaison entre les deux approches de résolution du RCPSPR proposées dans cette thèse permet de mesurer l'efficacité des deux approches. L'approche de résolution proposée dans le chapitre 4 est basée sur plusieurs étapes heuristiques. Cependant, elle permet de fournir des solutions du RCPSPR à partir d'un flot, ayant un écart inférieur à 10% par rapport à la solution optimale pour les instances de petite taille.

CONCLUSION GENERALE

Dans le cadre de cette thèse, nous nous sommes intéressé aux problèmes intégrés d'ordonnancement et de transport. Ces problèmes consistent à déterminer des ordonnancements et des tournées pour satisfaire des demandes de clients dont les produits doivent à la fois être fabriqués (ou les ressources utilisées) et transportés depuis un lieu d'origine vers un lieu de destination.

Une des difficultés de ces problèmes est de prendre en compte, de manière intégrée, les contraintes de l'ordonnancement et du transport. Après une introduction sur les problèmes d'ordonnancement et de tournée de véhicules, avec différentes méthodes de résolution, deux familles de problèmes intégrés sont étudiées.

Dans un *premier temps*, nous nous sommes intéressé au problème de type PTSP et plus particulièrement au problème proposé par (Geismar *et al.* 2008) qui introduit des contraintes spécifiques dues à la présence de produits périssables et dans lequel le transport est réalisé par un seul véhicule. Le problème de type PTSP a fait l'objet de deux études dans cette thèse :

- une étude basée sur le PTSP avec un seul véhicule (variante introduite par Geismar *et al.* en 2008) ;
- une étude basée sur une extension du PTSP avec une flotte homogène de véhicules.

Sur le PTSP défini par (Geismar *et al.*, 2008), une méthode de résolution basée sur une métaheuristique hybridée de type GRASP×ELS a été proposée. Cette méthode comprend différents points-clés :

- une fonction d'évaluation de type SPLIT ;
- une modélisation sous la forme d'un flow-shop à deux machines ;
- un algorithme de type Johnson ou de type Gilmore-Gomory.

Cette approche est ensuite étendue à la généralisation du PTSP avec une flotte homogène de véhicules. Une approche de résolution similaire, mais basée sur des mécanismes différents, est proposée. L'ensemble des travaux réalisés sur le PTPSP fait l'objet des chapitres 2 et 3.

Dans un *deuxième temps*, nous avons traité un problème de type RCPSPR défini comme une extension du RCPSP dans lequel des considérations liées au transport sont prises en compte.

Le terme de RCPSPR est introduit, pour définir un RCPSP avec routage. La partie routage est réalisée par une flotte hétérogène de véhicules permettant le transport des ressources entre les activités du RCPSP. Au problème classique du RCPSP s'ajoute donc :

- un problème de détermination des opérations de transfert afin de définir les quantités de ressources échangées entre les différentes activités ;
- un problème de détermination des opérations de transport à partir des opérations de transfert à effectuer entre les activités qui s'apparente à un problème de constitution de lots ;
- un problème d'affectation des opérations de transport aux véhicules ;
- un problème d'ordonnancement des opérations de transport pour créer des tournées des véhicules ;
- un problème d'intégration des opérations d'ordonnancement (activités) et de transport dans un graphe disjonctif pour gérer la coordination des activités, des véhicules et des ressources.

Le problème est traité, d'une part, avec une fonction d'évaluation « heuristique » et d'autre part, avec une méthode exacte de programmation dynamique. Ces deux méthodes ont fait l'objet des chapitres 4 et 5, et ont demandé la création de jeux de données spécifiques.

Dans cette thèse, nous nous sommes intéressé essentiellement, à des problèmes d'ordonnancement/transport de type PTSP et RCPSPR. Ces travaux offrent plusieurs perspectives et extensions possibles.

À ***court terme***, un certain nombre d'extensions peuvent être dégagées, comme par exemple, la résolution du RCPSPR avec plusieurs ressources, chacune pouvant être modélisée par un flot. Le problème d'intégration change alors de nature, puisque le début d'une activité est à la fois contraint, par le transport et par la nécessité, de disposer de tous les types de ressources nécessaires. Les différents types de ressources peuvent alors être traités différemment, par les véhicules, en incluant par exemple des contraintes interdisant le transport de certaines ressources dans un même véhicule ou encore en définissant des ressources ayant des poids différents.

À ***moyen terme***, l'utilisation du flot, comme information de niveau stratégique, pour la construction d'un planning intégré ordonnancement/transport pourrait être remise en cause. Retenir cette définition du « flot » comme étant la donnée d'entrée utilisée pour résoudre l'ordonnancement/transport, signifie qu'un véhicule n'est pas autorisé à se déplacer d'une activité i vers une activité j en transportant un volume supérieur à celui demandé par l'activité j. Ce type de contrainte se justifie pleinement, dans de nombreuses situations, en particulier, si le transport concerne des produits coûteux ou sensibles, pour lesquels aucune erreur de livraison n'est tolérée : on ne peut pas prendre le risque de livrer en j, un produit destiné à k et on adopte une approche de type 1-to-1 « classique ». La conséquence de cette hypothèse engendre des tournées qui sont du type Pickup-Delivery et qui possèdent un nombre d'aller/retours au dépôt important. Pour des produits standards, cette contrainte peut être relâchée et on pourrait alors rechercher des solutions (moins coûteuses en termes de temps de transport), dans lesquelles on remplacerait la notion de flot par la notion de flux. On entend par flux, le transport d'un volume de ressource qui pourrait être supérieur à la quantité demandée par l'activité.

Sur le ***long terme***, des axes de recherche nouveaux sont à définir dans la continuité des travaux actuels, en se basant essentiellement, sur les enjeux liés aux problèmes de tournées.

Dans cette optique, 3 axes prioritaires de recherches nous sembleraient intéressants à développer, puisqu'ils correspondent à des enjeux sociétaux majeurs liés à la qualité de service au niveau du transport ou à un aspect économique. On sait en effet, que l'acceptabilité des solutions, dépend de la qualité des solutions, la qualité perçue par le client est souvent très différente de celle considérée en optimisation. Trop souvent en effet, les critères traités sont le makespan pour l'ordonnancement, ou le coût de transport pour les problèmes de tournée de véhicule, coût qui se résume soit à un temps ou à une distance parcourue.

De ce point de vue, il y aurait un intérêt certain, à inclure l'aspect *qualité de service* dans la résolution des problèmes de tournées. Les notions de qualité de service sont prises en compte, dans les problèmes de type DARP, où l'on suppose que le transport concerne non pas des marchandises, mais des personnes. (Cordeau *et al.*, 2003) ont introduit la notion de Total Riding Time (TRT), de Total Duration (TD) et de Total Waiting Time (TWT) et ils ont proposé une approche, tirant profit de la notion de time-lags maximaux et minimaux, dans un algorithme en $O(n^2)$. Ces trois notions pourraient être redéfinies, en associant par exemple, la notion de TD à la durée d'utilisation des véhicules dans le RCPSPR, la notion de TRT à la durée de transport des véhicules, et la notion de TWT à la durée d'attente des véhicules. Le problème ne pourrait plus alors, se ramener à un « simple calcul » de plus court chemin dans un graphe et on ne chercherait plus alors, une solution semi-active, mais une solution de compromis, entre un coût et une qualité de service qu'on pourrait définir, en s'inspirant de ce que propose (Cordeau *et al.*, 2003). La qualité de service pourrait être également mesurée, grâce au temps d'attente des ressources, qui peuvent être livrées en plusieurs fois sur les activités dans le RCPSPR. Un coût supplémentaire, prenant en compte ces attentes comme étant des pénalités, pourrait être envisagé.

Un second axe prospectif pourrait s'organiser autour de la *notion d'équité*, souvent absente dans les problèmes de transport ou d'ordonnancement. Du point de vue du client, le retard ou délai moyen (ou la qualité de service) est, de fait, un indicateur pertinent. Chaque client est plus sensible à la qualité de service qu'il a eu, qu'à la qualité de service de l'ensemble des autres clients. On peut généraliser la réflexion, en pensant à la manière dont les usagers perçoivent la qualité d'un service public. Bien souvent, l'usager est sensible à la qualité de service dont il bénéficie, mais il peut aussi être agacé de constater que d'autres usagers ont eu un meilleur service. Les scènes de tension dans un bureau de poste, les scènes de réclamation dans les gares ou aéroports sont souvent liées, à la perception d'une qualité de service mesurée en temps d'attente. C'est pourquoi, on préfère souvent des solutions équitables entre les clients (ou usagers), *i.e.* avec des différences de qualité de service entre les clients faible. Dans les problèmes de tournée de véhicules de type « split delivery », cette notion d'équité est souvent prise en compte, en essayant de limiter le nombre de livraisons, pour un même client.

Finalement, il y aurait un intérêt à inclure un *aspect économique* dans la résolution des problèmes intégrés d'ordonnancement et de transport, non pas comme une nouvelle contrainte à inclure, mais comme un objectif à minimiser, ou comme une situation

d'équilibre à trouver. On peut penser par exemple, à un système de tarification variable, dans lequel le prix payé par le client dépendrait du niveau de qualité de service qu'il souhaite recevoir. On ferait alors le lien, entre d'une part, le fournisseur d'un service et d'autre part les clients, vus comme des acteurs économiques. Cette approche pourrait correspondre à des enjeux concrets au sein de la chaîne logistique de différentes entreprises.

BIBLIOGRAPHIE

Afsar, H. M., Lacomme, P., Ren, L., Prodhon, C., et Vigo, D. (2016) « Resolution of a Job-Shop problem with transportation constraints: a master/slave approach, » *IFAC-PapersOnLine*, Vol. 49, No.12, pp. 898–903.

Aggarwal, A., Schieber, B., et Tokuyama, T. (1994) « Finding a minimum-weight k-link path in graphs with the concave Monge property and applications, » *Discrete & Computational Geometry*, Vol. 12, No.3, pp. 263–280.

Ahuja, R. K., Magnanti, T. L., et Orlin, J. B. (1993) *Network Flows*, Prentice hall, Upper Saddle River, New Jersey.

Alvarez-Valdes, R., et Tamarit, J. M. (1993) « The project scheduling polyhedron: dimension, facets and lifting theorems, » *European Journal of Operational Research*, Vol. 67, No.2, pp. 204–220.

Armstrong, R., Gao, S., et Lei, L. (2008) « A zero-inventory production and distribution problem with a fixed customer sequence, » *Annals of Operations Research*, Vol. 159, No.1, pp. 395–414.

Artigues, C., Michelon, P., et Reusser, S. (2003) « Insertion techniques for static and dynamic resource-constrained project scheduling, » *European Journal of Operational Research*, Vol. 149, No.2, pp. 249–267.

Artigues, C., et Roubellat, F. (2000) « A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes, » *European Journal of Operational Research*, Vol. 127, No.2, pp. 297–316.

Balas, E. (1970) « Project scheduling with resource constraints, » *Applications of Mathematical Programming Techniques*, American Elsevier, pp. 187–200.

Ballou, R. H. (1992) *Business Logistics Management*, Prentice-Hall, Englewood Cliffs.

Bardossy, M. G., et Shier, D. R. (2006) « Label-Correcting Shortest Path Algorithms Revisited, » *Perspectives in Operations Research*, Operations Research/Computer Science Interfaces Series, Springer, Boston, MA, pp. 179–197.

Baum, E. B. (1986) « Iterated Descent: A Better Algorithm for Local Search in Combinatorial Optimization Problems, » Caltech, Pasadena.

Beasley, J. (1983) « Route first—Cluster second methods for vehicle routing, » *Omega*, Vol. 11, No.4, pp. 403–408.

Beasley, J. E., et Christofides, N. (1989) « An algorithm for the resource constrained shortest path problem, » *Networks*, Vol. 19, No.4, pp. 379–394.

Bellman, R. (1958) « On a routing problem, » *Quarterly of Applied Mathematics*, Vol. 16, No.1, pp. 87–90.

Bierwirth, C. (1995) « A generalized permutation approach to job shop scheduling with genetic algorithms, » *Operations-Research-Spektrum*, Vol. 17, No.2–3, pp. 87–92.

Binato, S., Hery, W. J., Loewenstern, D. M., et Resende, M. G. C. (2002) « A Grasp for Job Shop Scheduling, » *Essays and Surveys in Metaheuristics*, Operations Research, Springer, Boston, MA, pp. 59–79.

Blazewicz, J., Lenstra, J. K., et Rinnooy Kan, A. H. . (1983) « Scheduling subject to resource constraints: classification and complexity, » *Discrete Applied Mathematics*, Vol. 5, No.1, pp. 11–24.

Blum, C., et Roli, A. (2003) « Metaheuristics in combinatorial optimization: Overview and conceptual comparison, » *ACM Computing Surveys (CSUR)*, Vol. 35, No.3, pp. 268–308.

Boudia, M., Prins, C., et Reghioui, M. (2007) « An Effective Memetic Algorithm with Population Management for the Split Delivery Vehicle Routing Problem | SpringerLink, » Springer, Dortmund, Germany, pp. 16–30.

Brucker, P., Drexl, A., Mohring, R., Neumann, K., et Pesch, E. (1999) « Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods, » *European Journal of Operational Research*, Vol. 112, No.3, p. 41.

Brucker, P., Knust, S., Roper, D., et Zinder, Y. (2000) « Scheduling UET task systems with concurrency on two parallel identical processors, » *Mathematical Methods of Operations Research*, Vol. 52, No.3, pp. 369–387.

Caumond, A., Lacomme, P., Moukrim, A., et Tchernev, N. (2009) « An MILP for scheduling problems in an FMS with one vehicle, » *European Journal of Operational Research*, Vol. 199, No.3, pp. 706–722.

Chang, Y.-C., et Lee, C.-Y. (2004) « Machine scheduling with job delivery coordination, » *European Journal of Operational Research*, Vol. 158, No.2, pp. 470–487.

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

Chassaing, M., Fontanel, J., Lacomme, P., Ren, L., Tchernev, N., et Villechenon, P. (2014) « A GRASPxELS Approach for the Job-shop with a Web Service Paradigm Packaging, » *Expert Systems with Applications*, Vol. 41, No.2, pp. 544–562.

Chen, P., Huang, H., et Dong, X.-Y. (2010) « Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem, » *Expert Systems with Applications*, Vol. 37, No.2, pp. 1620–1627.

Chen, Z.-L. (2010) « Integrated Production and Outbound Distribution Scheduling: Review and Extensions, » *Operations Research*, Vol. 58, No.1, pp. 130–148.

Chen, Z.-L., et Vairaktarakis, G. L. (2005) « Integrated Scheduling of Production and Distribution Operations, » *Management Science*, Vol. 51, No.4, pp. 614–628.

Cheng, R., Gen, M., et Tsujimura, Y. (1996) « A Tutorial Survey of Job-shop Scheduling Problems Using Genetic Algorithms—I: Representation, » *Computers & Industrial Engineering*, Vol. 30, No.4, pp. 983–997.

Cheref, A., Artigues, C., et Billaut, J.-C. (2016) « A new robust approach for a production scheduling and delivery routing problem, » Troyes, France.

Christopher, M. (1992) *Logistics and Supply Chain Management*, Financial Times Prentice Hall, London.

Cook, S. A. (1971) « The Complexity of Theorem-proving Procedures, » *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, ACM, New York, NY, USA, pp. 151–158.

Corberan, A., Erdogan, G., Laporte, G., Plana, I., et Sanchis, J. M. (In Press) « The Chinese Postman Problem with load-dependent costs, » *Transportation Science*.

Damay, J., Quilliot, A., et Sanlaville, E. (2007) « Linear programming based algorithms for preemptive and non-preemptive RCPSP, » *European Journal of Operational Research*, Vol. 182, No.3, pp. 1012–1022.

Dantzig, G. B., et Ramser, J. H. (1959) « The truck dispatching problem, » *Management science*, Vol. 6, No.1, pp. 80–91.

Dantzig, G., Fulkerson, R., et Johnson, S. (1954) « Solution of a large-scale traveling-salesman problem, » *Journal of the operations research society of America*, Vol. 2, No.4, pp. 393–410.

De Reyck, B., et Herroelen, W. (1998) « A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations, » *European Journal of Operational Research*, Vol. 111, No.1, pp. 152–174.

Dell'Amico, M., et Trubian, M. (1993) « Applying tabu search to the job-shop scheduling problem, » *Annals of Operations Research*, Vol. 41, No.3, pp. 231–252.

Demeulemeester, E., et Herroelen, W. (1992) « A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem, » *Management Science*, Vol. 38, No.12, pp. 1803–1818.

Demeulemeester, E. L., et Herroelen, W. (2002) *Project Scheduling - A Research Handbook*, Springer.

Desrochers, M. (1986) « La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes, » Centre de recherche sur les Transport, Université de Montréal, Canada.

Desrochers, M. (1988) *An algorithm for the shortest path problem with resource constraints*, *Technical Report G-88-27, GERAD*, Montreal, Canada.

Devapriya, P. (2008) « Optimal Fleet Size of an Integrated Production and Distribution Scheduling Problem for a Single Perishable Product, » Clemson University.

Devapriya, P., Ferrell, W., et Geismar, N. (2017) « Integrated production and distribution scheduling with a perishable product, » *European Journal of Operational Research*, Vol. 259, No.3, pp. 906–916.

Dijkstra, E. W. (1959) « A Note on Two Problems in Connexion with Graphs, » *Numerische Mathematik*, Vol. 1, No.1, pp. 269–271.

Dongarra, J. J. (2014) *Performance of various computers using standard linear equations software*, University of Tennessee, Computer Science Department.

Duhamel, C., Gouinaud, C., Lacomme, P., et Prodhon, C. (2013) « A Multi-thread GRASPxELS for the Heterogeneous Capacitated Vehicle Routing Problem, » *Hybrid Metaheuristics*, Studies in Computational Intelligence, Springer, Berlin, Heidelberg, pp. 237–269.

Duhamel, C., Lacomme, P., Prins, C., et Prodhon, C. (2010) « A GRASPxELS Approach for the Capacitated Location-Routing Problem, » *Computers and Operations Research*, Vol. 37, No.11, pp. 1912–1923.

Duhamel, C., Lacomme, P., et Prodhon, C. (2011) « Efficient Frameworks for Greedy Split and New Depth First Search Split Procedures for Routing Problems, » *Computers and Operations Research*, Vol. 38, No.4, pp. 723–739.

Duhamel, C., Lacomme, P., et 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*, Vol. 25, No.2, pp. 345–358.

Edmonds, J., et Johnson, E. L. (1973) « Matching, euler tours and the chinese postman, » *Mathematical Programming*, Vol. 5, pp. 88–124.

Eiselt, H. A., Gendreau, M., et Laporte, G. (1995) « Arc Routing Problems, Part II: The Rural Postman Problem, » *Operations Research*, Vol. 43, No.3, pp. 399–414.

Feillet, D., Dejax, P., Gendreau, M., et Gueguen, C. (2004) « An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems, » *Networks*, Vol. 44, No.3, pp. 216–229.

Feo, T. A., et Resende, M. G. C. (1989) « A probabilistic heuristic for a computationally difficult set covering problem, » *Operations Research Letters*, Vol. 8, No.2, pp. 67–71.

Feo, T. A., et Resende, M. G. C. (1995) « Greedy Randomized Adaptive Search Procedures, » *Journal of Global Optimization*, Vol. 6, No.2, pp. 109–133.

Fondrevelle, J., Oulamara, A., et Portmann, M.-C. (2004) « Permutation Flowshop Scheduling Problems with Maximal and Minimal Time Lags, » *Computers and Operations Research*, Vol. 33, No.6, pp. 1540–1556.

Ford Jr., L. R. (1956) *Network Flow Theory*, Santa Monica, California, RAND Corporation.

Gandibleux, X., Beugnies, F., et Randriamasy, S. (2006) « Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function, » *4OR: A Quarterly Journal of Operations Research*, Vol. 4, No.1, pp. 47–59.

Geismar, H. N., Laporte, G., Lei, L., et Sriskandarajah, C. (2008) « The Integrated Production and Transportation Scheduling Problem for a Product with a Short Lifespan, » *INFORMS Journal on Computing*, Vol. 20, No.1, pp. 21–33.

Gilmore, P. C., et Gomory, R. E. (1964) « Sequencing a One State-Variable Machine: A Solvable Case of the Traveling Salesman Problem, » *Operations Research*, Vol. 12, No.5, pp. 655–679.

Gilsinn, J., et Witzgall, C. (1973) *A performance comparison of labeling algorithms for calculating shortest path trees*, *National Bureau of Standards*, U.S. Department of Commerce, Washington.

Glabowski, M., Musznicki, B., Nowak, P., et Zwierzykowski, P. (2014) « Review and Performance Analysis of Shortest Path Problem Solving Algorithms, » *International Journal on Advances in Software*, Vol. 7, No.1–2, pp. 20–30.

Glover, F. (1986) « Future paths for integer programming and links to artificial intelligence, » *Computers & Operations Research*, Applications of Integer Programming, Vol. 13, No.5, pp. 533–549.

Golden, B. L., et Wong, R. T. (1981) « Capacitated arc routing problems, » *Networks*, Vol. 11, No.3, pp. 305–315.

Grabowski, J., Nowicki, E., et Zdrzalka, S. (1986) « A Block Approach for Single-Machine Scheduling with Release Dates and Due Dates, » *European Journal of Operational Research*, Vol. 26, No.2, pp. 278–285.

Hartmann, S., and Briskorn, D. (2010) « A survey of variants and extensions of the resource-constrained project scheduling problem, » *European Journal of Operational Research*, Vol. 207, No.1, pp. 1–14.

Herroelen, W., De Reyck, B., et Demeulemeester, E. (1998) « Resource-constrained project scheduling: a survey of recent developments, » *Computers & Operations Research*, Vol. 25, No.4, pp. 279–302.

Hurink, J., et Knust, S. (2002) « A tabu search algorithm for scheduling a single robot in a job-shop environment, » *Discrete applied mathematics*, Vol. 119, No.1–2, pp. 181–203.

Hurink, J., et Knust, S. (2005) « Tabu search algorithms for job-shop problems with a single transport robot, » *European Journal of Operational Research*, Vol. 162, No.1, pp. 99–111.

Jackson, J. R. (1955) *Scheduling a production line to minimize maximum tardiness*, Management research project, *Technical Report 43*, University of California, Los Angeles.

Jamili, N., Ranjbar, M., et Salari, M. (2016) « A bi-objective model for integrated scheduling of production and distribution in a supply chain with order release date restrictions, » *Journal of Manufacturing Systems*, Vol. 40, pp. 105–118.

Johnson, S. M. (1954) « Optimal two- and three-stage production schedules with setup times included, » *Naval Research Logistics Quarterly*, Vol. 1, No.1, pp. 61–68.

Karaoğlan, İ., et Kesen, S. E. (2017) « The coordinated production and transportation scheduling problem with a time-sensitive product: a branch-and-cut algorithm, » *International Journal of Production Research*, Vol. 55, No.2, pp. 536–557.

Kimms, A. (2001) *Mathematical Programming and Financial Objectives for Scheduling Projects*, International Series in Operations Research & Management Science, Springer, Boston, MA.

Knust, S. (1999) « Shop-Scheduling Problems with Transportation, » *Fachbereich Mathematik/ Informatik*, Universität Osnabrück.

Kolisch, R. (1996) « Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation, » *European Journal of Operational Research*, Vol. 90, No.2, pp. 320–333.

Kolisch, R., et Hartmann, S. (2006) « Experimental investigation of heuristics for resource-constrained project scheduling: An update, » *European journal of operational research*, Vol. 174, No.1, pp. 23–37.

Kolisch, R., et Padman, R. (2001) « An integrated survey of deterministic project scheduling, » *Omega*, Vol. 29, No.3, pp. 249–272.

Krüger, D., et Scholl, A. (2010) « Managing and modelling general resource transfers in (multi-)project scheduling, » *OR Spectrum*, Vol. 32, No.2, pp. 369–394.

Lacomme, P., Larabi, M., et Tchernev, N. (2007) « A disjunctive graph for the job-shop with several robots, » *MISTA conference*, pp. 285–292.

Lacomme, P., Larabi, M., et Tchernev, N. (2013) « Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles, » *International Journal of Production Economics*, Vol. 143, No.1, pp. 24–34.

Lacomme, P., Prins, C., et Ramdane-Chérif, W. (2001) « A Genetic Algorithm for the Capacitated Arc Routing Problem and Its Extensions, » *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 473–483.

Lacomme, P., Toussaint, H., et Duhamel, C. (2013) « A GRASPxELS for the vehicle routing problem with basic three-dimensional loading constraints, » *Engineering Applications of Artificial Intelligence*, Vol. 26, No.8, pp. 1795–1810.

Lenstra, J. K., et Kan, A. H. G. R. (1976) « On general routing problems, » *Networks*, Vol. 6, No.3, pp. 273–280.

Li, C.-L., Vairaktarakis, G., et Lee, C.-Y. (2005) « Machine scheduling with deliveries to multiple customer locations, » *European Journal of Operational Research*, Vol. 164, No.1, pp. 39–51.

Li, K., Zhou, C., Leung, J. Y.-T., et Ma, Y. (2016) « Integrated Production and Delivery with Single Machine and Multiple Vehicles, » *Expert Syst. Appl.*, Vol. 57, No.C, pp. 12–20.

Lourenço, H. R., Martin, O. C., et Stützle, T. (2003) « Iterated Local Search, » *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, Springer, Boston, MA, pp. 320–353.

Lummus, R. R., Vokurka, R. J., et Albert, K. L. (1998) « Strategic supply chain planning, » *Production and Inventory Management Journal*, Vol. 39, No.3, pp. 49–58.

Matsuo, H., Sush, C. J., et Sullivan, R. S. (1988) *A controlled search simulated annealing method for the general job-shop scheduling problem*, Graduate School of Business, University of Texas at Austin, Texas, USA.

Mei-Ko, K. (1962) « Graphic programming using odd or even point s, » *Chinese Mathematics*, Vol. 1, pp. 273–277.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., et Bianco, L. (1998) « An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation, » *Manage. Sci.*, Vol. 44, No.5, pp. 714–729.

Minty, G. J. (1957) « A Comment on the Shortest-Route Problem, » *Operations Research*, Vol. 5, No.5, pp. 724–724.

Moons, S., Ramaekers, K., Caris, A., et Arda, Y. (2017) « Integrating Production Scheduling and Vehicle Routing Decisions at the Operational Decision Level, » *Computers & Industrial Engineering*, Vol. 104, No.C, pp. 224–245.

Moukrim, A., et Quilliot, A. (2005) « Optimal Preemptive Scheduling on a Fixed Number of Identical Parallel Machines, » *Operations Research Letters*, Vol. 33, No.2, pp. 143–150.

Orloff, C. S. (1974) « A fundamental problem in vehicle routing, » *Networks*, Vol. 4, No.1, pp. 35–64.

Osman, I. H., et Laporte, G. (1996) « Metaheuristics: A bibliography, » *Annals of Operations Research*, Vol. 63, No.5, pp. 511–623.

Papadimitriou, C. H. (1976) « On the Complexity of Edge Traversing, » *Journal of the Association for Computing Machinery*, Vol. 23, No.3, pp. 544–554.

Parragh, S. N., Doerner, K. F., et Hartl, R. F. (2008a) « A survey on pickup and delivery problems, » *Journal für Betriebswirtschaft*, Vol. 58, No.1, pp. 21–51.

Parragh, S. N., Doerner, K. F., et Hartl, R. F. (2008b) « A survey on pickup and delivery problems, » *Journal für Betriebswirtschaft*, Vol. 58, No.2, pp. 81–117.

Pearn, W.-L., Assad, A., et Golden, B. L. (1987) « Transforming Arc Routing into Node Routing Problems, » *Computers and Operations Research*, Vol. 14, No.4, pp. 285–288.

Pinedo, M. L. (2008) *Scheduling: Theory, Algorithms, and Systems*, Springer New York, New York, NY.

Poirier, C., et Reiter, S. (2001) *La Supply Chain : Optimiser la chaîne logistique et le réseau interentreprises*, Realites De L'entreprise, Dunod, Paris.

Poppenborg, J., et Knust, S. (2016) « A Flow-based Tabu Search Algorithm for the RCPSP with Transfer Times, » *OR Spectrum*, Vol. 38, No.2, pp. 305–334.

Prins, C. (2004) « A simple and effective evolutionary algorithm for the vehicle routing problem, » *Computers & Operations Research*, Vol. 31, No.12, pp. 1985–2002.

Prins, C. (2009) « A GRASP × Evolutionary Local Search Hybrid for the Vehicle Routing Problem, » *Bio-inspired Algorithms for the Vehicle Routing Problem*, Studies in Computational Intelligence, Springer, Berlin, Heidelberg, pp. 35–53.

Prins, C., Lacomme, P., et Prodhon, C. (2014) « Order-first split-second methods for vehicle routing problems: A review, » *Transportation Research Part C: Emerging Technologies*, Vol. 40, pp. 179–200.

Pritsker, A. A. B., Watters, L. J., et Wolfe, P. M. (1969) « Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach, » *Management Science*, Vol. 16, No.1, pp. 93–108.

Quilliot, A., et Toussaint, H. (2013) « Flow Models for Project Scheduling with Transfer Delays, Recent Advances in Computational Optimization, » *Studies in Computational Intelligence*, Springer.

Ramdane-Cherif, W. (2002) « Problèmes d'optimisation en tournées sur arcs, » Université de Technologie de Troyes.

Reeves, C. R. (Ed.). (1993) *Modern Heuristic Techniques for Combinatorial Problems*, Wiley, New York.

Reghioui, M., Prins, C., et Labadi, N. (2007) « GRASP with Path Relinking for the Capacitated Arc Routing Problem with Time Windows, » *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 722–731.

Ren, L. (2012) « Conception et évaluation d'outils décisionnels pour des systèmes réactifs d'aide à la mobilité, » Université Blaise Pascal-Clermont-Ferrand II.

Roy, B. (1970) *Algèbre moderne et théorie des graphes. Tome 2, Chapitre 8*, Dunod, Paris.

Roy, B., et Sussmann, B. (1964) *Les problèmes d'ordonnancement avec contraintes disjonctives*, *Technical Report, Note DS no 9 bis, SEMA*, Paris.

Sachs, H., Stiebitz, M., et Wilson, R. J. (1988) « An historical note: Euler's Königsberg letters, » *Journal of Graph Theory*, Vol. 12, No.1, pp. 133–139.

Sarmiento, A. M., et Nagi, R. (1999) « A review of integrated analysis of production–distribution systems, » *IIE Transactions*, Vol. 31, No.11, pp. 1061–1074.

Sprecher, A. (2000) « Scheduling Resource-Constrained Projects Competitively at Modest Memory Requirements, » *Management Science*, Vol. 46, No.5, pp. 710–723.

Stützle, T. G. (1998) « Local search algorithms for combinatorial problems - Analysis, Improvements, et New Applications, » Darmstadt University of Technology.

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

Ulusoy, G. (1985) « The fleet size and mix problem for capacitated arc routing, » *European Journal of Operational Research*, Vol. 22, No.3, pp. 329–337.

Valls, V., Ballestín, F., et Quintanilla, S. (2005) « Justification and RCPSP: A technique that pays, » *European Journal of Operational Research*, Vol. 165, No.2, pp. 375–386.

Van Laarhoven, P. J. M., Aarts, E. H. L., et Lenstra, J. K. (1992) « Job Shop Scheduling by Simulated Annealing, » *Operations Research*, Vol. 40, No.1, pp. 113–125.

Vidal, T. (2015) « Technical Note: Split Algorithm in O(n) for the Vehicle Routing Problem, » *arXiv:1508.02759 (cs)*.

Weglarz, J. (1999) *Project Scheduling - Recent Models, Algorithms and Applications*, International Series in Operations Research & Management Science, Springer US.

Wilbert, R. (1988) « The Concave Least-Weight Subsequence, » *Journal of Algorithms*, pp. 418–425.

Wolf, S., et Merz, P. (2007) « Evolutionary Local Search for the Super-Peer Selection Problem and the p-Hub Median Problem, » *Hybrid Metaheuristics*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 1–15.

Zhang, Q., Manier, H., et Manier, M.-A. (2012a) « A Modified Disjunctive Graph for Job-Shop Scheduling Problems with Bounded Processing Times and Transportation Constraints, » *IFAC Proceedings*, Vol. 45, No.6, pp. 1377–1382.

Zhang, Q., Manier, H., et Manier, M.-A. (2012b) « A Genetic Algorithm with Tabu Search Procedure for Flexible Job Shop Scheduling with Transportation Constraints and Bounded Processing Times, » *Computers and Operations Research*, Vol. 39, No.7, pp. 1713–1723.