Modélisation et algorithmes pour le dimensionnement et l'ordonnancement cyclique d'atelier de traitement de surface

Emna Laajili

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE A L'UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

École doctorale n°37

SPIM (Sciences Pour l'Ingénieur et Microtechniques)

Doctorat en Automatique

Par

Mme LAAJILI Emna

Modélisation et algorithmes pour le dimensionnement et l'ordonnancement cyclique d'atelier de traitement de surface

Thèse présentée et soutenue à Belfort, le 31 Mars 2021

Composition du Jury :

Mme, PIAT Nadine	Professeure, ENSMM Besançon	Présidente
Mme, BRIL-EL HAOUZI Hind	Professeure, ENSTIB Epinal	Rapporteuse
Mr, KACEM Imed	Professeur, Université de Lorraine	Rapporteur
Mr, LACOMME Philippe	Maître de Conférences, Université Clermont-Auvergne	Examinateur
Mme, YALAOUI Alice	Maître de Conférences, UTT Troyes	Examinatrice
Mme, MANIER Marie-Ange	Maître de Conférences HDR, UTBM Belfort	Directrice de thèse
Mr, NICOD Jean-Marc	Professeur, ENSMM Besançon	Co-directeur de thèse
Mr, LAMROUS Sid	Maître de Conférences, UTBM Belfort	Co-encadrant de thèse

N° | 2 | 0 | 2 | 1 | U | B | F | C | A | 0 | 0 | 6 |

Je dédie ce travail

À MOI MÊME ET À TOUTES CES ANNÉES D'ÉTUDE

À MA CHÈRE MÈRE

Source inépuisable d'amour, de tendresse et de patience. Même dans la distance, tu étais et tu es, tous les jours un pilier de ma vie. Ta prière et tes bénédictions m'ont été d'un grand secours tout au long de mon parcours. Les mots ne sauront jamais décrire ma grande affection et ma profonde reconnaissance. Mon adorée !

Puisse Dieu tout puissant, te préserver et t'accorder santé, longue vie et bonheur.

À MON CHER PÈRE

Le meilleur de tous les pères. Pour tes années de sacrifices et efforts, et pour les rides qui embellissent ton front, je t'offre ce travail et toute ma réussite. Je sais à quel point tu en es fier et ça me rend encore plus radieuse. J'espère avoir apporté le fruit de tes longues années de sollicitudes, d'encouragement et de prières. Les mots ne te suffiront jamais pour exprimer mon grand amour !

Puisse Dieu te préserver et te procurer santé, longue vie et bonheur.

À LA MÉMOIRE DE MA GRAND-MÈRE

À MA SŒUR

La meilleure des sœurs. Tu as su être présente quand il fallait. Tes mots tendres et tes encouragements ont su avoir le chemin à mon cœur et à mon esprit. Je te dédie ce travail en témoignage de mon amour et de ma reconnaissance. Dans la distance, nous avons su être là l'une pour l'autre. Nous en garderons de beaux souvenirs de notre indéfectible union qui s'est tissée au fil de ces années !

À MON FRÈRE

Le plus cher des frères. Même avec tes rares expressions, tu sais toujours cibler tes mots d'affection et d'encouragements. Je sais que tu es fier de moi ! Tu es et tu resteras l'homme que j'aime le plus après mon père. A toi, je dédie ce travail pour essayer de te traduire toute l'affection que j'ai pour toi, et pour te remercier de la joie de vivre que tu m'apportes tous les jours.

À TOUS CEUX QUI M'AIMENT ET AIMENT ME VOIR RÉUSSIR

Remerciements

Dans le cadre de cette thèse, je tiens à remercier plusieurs personnes qui ont contribué au bon déroulement de ce projet :

Tout d'abord, ma directrice de thèse Marie-Ange MANIER, mon co-directeur de thèse Jean-Marc NICOD et mon co-encadrant Sid LAMROUS, qui m'ont bien accueillie au sein de l'équipe de recherche OMNI du département DISC de l'institut FEMTO-ST, et encadrée tout au long de cette thèse. Je les remercie pour leur suivi, leurs conseils et le partage de leurs brillantes intuitions. Qu'ils soient aussi remerciés pour leur gentillesse, leur disponibilité permanente et pour les nombreux encouragements qu'ils m'ont prodigués.

Je remercie également, Madame Nadine PIAT, Professeure à l'ENSMM de Besançon, d'avoir accepté de présider le jury de ma thèse, tout en partageant de sa douceur et de sa rigueur, et de me faire une belle délibération qui m'a envahie d'émotions.

J'adresse sans doute mes précieux remerciements à Madame Hind BRIL EL-HAOUZI (Professeure à l'ENSTIB d'Épinal), à Madame Alice YALAOUI (Maître de Conférences à l'UTT de Troyes), à Monsieur Imed KACEM (Professeur à l'université de Lorraine), ainsi qu'à Monsieur Philippe LACOMME (Maître de Conférences à l'université de Clermont-Auvergne), de l'honneur qu'ils m'ont accordé en acceptant de rapporter ou d'examiner ma thèse. Leurs questions et suggestions constructives m'ont bien éclairée sur de belles autres perspectives de la thèse.

Enfin, je veux exprimer mes sincères remerciements à toute personne qui a donné de son temps et/ou de son âme pour m'aider dans la réalisation de ce projet de recherche et l'obtention de ce précieux diplôme. Je ne mentionne pas des noms mais je n'oublie sans doute personne ! Chaque individu qui m'a connu durant ces années de thèse se voit concerné de ma gratitude !

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Dans un environnement industriel concurrentiel, les entreprises se battent toujours pour produire des biens de meilleure qualité, avec des coûts moindres et dans des délais les plus raccourcis. En accentuant l'attractivité technologique et esthétique de leurs produits pour satisfaire au mieux leurs clients, elles doivent continuellement persévérer pour augmenter leur rentabilité en jouant sur les deux facteurs : temps et coûts. Vers cet objectif, une des pistes les plus prometteuses est l'optimisation des chaînes de production. Cette optimisation concourt à l'amélioration de la productivité, autrement dit, produire plus dans un temps plus court. Par ailleurs, elle vise à une meilleure gestion des ressources disponibles pour épargner sur le plan humain, matériel ou encore énergétique. Par conséquent, cette optimisation contribue à favoriser le rendement des entreprises, renforcer leurs revenus économiques et les aider à mieux se positionner par rapport à leurs concurrents sur le marché.

Dans ce contexte, l'ordonnancement des opérations s'offre comme une solution nécessaire et impérative pour l'optimisation des processus de production. Il y joue un rôle primordial car il maximise l'efficacité des opérations en éliminant les pertes inutiles de temps non productif. Il a donc un impact fondamental sur l'amélioration de la productivité en réduisant le temps et les coûts de production. Cet ordonnancement, en outre, dépend de plusieurs paramètres tels que le type d'atelier, la nature et le nombre des ressources mises à disposition, la gamme opératoire et le nombre des opérations à accomplir, ou encore les contraintes de production, etc. Le croisement et l'accroissement de ces facteurs, ce qui est le cas dans les systèmes industriels contemporains, rend les problèmes d'ordonnancement plus complexes et leur résolution d'autant plus difficile.

Dans les ateliers robotisés, le problème d'ordonnancement devient encore plus critique. L'automatisation des processus a envahi les systèmes de production depuis l'aube des années 70 avec la troisième révolution industrielle. De nos jours, nous vivons l'essor de l'usine connectée, au coeur de la quatrième révolution industrielle, où machines, progiciels et produits doivent communiquer en permanence. Grâce à l'arrivée de la numérisation, l'industrie devient un système global interconnecté. Dans cette nouvelle ère 4.0, les processus ont moins droit à l'erreur, d'autant plus les méthodes d'ordonnancement. Faire le nécessaire dans les meilleurs délais permettra de remonter les bonnes informations juste à temps, notamment celles liées à l'avancement dans le temps des processus, produits et machines de la chaîne de production. Ainsi, le développement de nouvelles méthodes de résolution des problèmes d'ordonnancement, devient une nécessité impérieuse non seulement pour miser sur l'efficacité et la productivité de la ligne, mais aussi, pour appuyer sa traçabilité dans l'usine connectée.

Un atelier de traitement de surface fait en général partie de la famille des ateliers robotisés, dans lequel des procédés de galvanoplastie sont réalisés sur un ensemble de pièces. Comme exemple bien connu de ces procédés, nous citons la production de cartes de circuits imprimés (PCB : *Printed Circuit Boards*). Le traitement de surface tel que la galvanoplastie est une opération intermédiaire des processus de production, qui s'effectue souvent entre l'usinage et le montage. C'est un procédé qui consiste à revêtir une pièce d'une fine couche de métal, pour lui donner une propriété de surface désirée, que ce soit mécanique comme la conductivité électrique, la résistance à la rouille, la protection contre l'oxydation et la corrosion, ou même des qualités esthétiques, notamment la dorure, le chromage, le dépôt d'argent, etc.

Dans les installations de galvanoplastie, les pièces doivent tremper dans plusieurs cuves contenant des solutions chimiques. Chacune est nécessaire pour une étape spécifique du traitement de la pièce, comme le nettoyage, le décapage à l'acide, le placage, le rinçage, etc. Des robots, commandés par ordinateur, sont responsables de la manutention de ces pièces. Le traitement des pièces, à la différence d'autres ateliers de production, ne se réalise pas à durées fixes mais plutôt dans des intervalles de temps bien définis, dans lesquelles se situent les durées de trempage. Dans le cas où les durées de traitement sont assez importantes, les cuves sont considérées comme les ressources critiques de la ligne. En contrepartie, les robots peuvent aussi devenir les ressources critiques de la ligne, quand le nombre des cuves et/ou le nombre des produits à manipuler devient assez élevé. Dans ces cas, il peut apparaître un ou plusieurs goulets d'étranglement sur la ligne de production. D'où le rôle accentué des méthodes d'ordonnancement qui viennent lisser la cadence de production pour en tirer un gain sur la productivité de la ligne.

Ces lignes de traitement de surface sont souvent des systèmes de production de masse où un nombre important de pièces doit être traité, suivant la même séquence de traitement. Les robots de manutention peuvent alors répéter indéfiniment une même séquence de mouvements pour transférer les pièces tout au long de la ligne. La répétition de la même séquence constitue le cycle de production et la séquence fixe de mouvements répétée par les robots est appelée la séquence cyclique. Le problème qui recherche la séquence cyclique qui minimise la période du cycle est connu sous le nom de problème d'ordonnancement cyclique des ateliers de traitement de surface (CHSP : *Cyclic Hoist Scheduling Problem*) [Manier & Baptiste 1994].

Le travail présenté dans cette thèse s'intéresse à une variante du CHSP assez peu étudiée, qui englobe en amont une partie de la conception de l'atelier, plus précisément le dimensionnement des ressources de manutention. En effet, la plupart des approches précédentes se sont très majoritairement concentrées sur l'ordonnancement des déplacements des robots en supposant que la conception de la ligne est déjà fournie et que le nombre de robots est une donnée fixe. Toutefois, dans notre étude, nous supposons que le nombre de moyens de transport peut aussi être optimisé et est donc inconnu. Nous cherchons alors les meilleurs couples (nombre de robots,

temps de cycle optimal associé). Nous traitons ainsi de manière innovante à la fois le problème de conception et d'ordonnancement cyclique de l'atelier de traitement de surface. Nous l'appelons CHDSP pour *Cyclic Hoist Design and Scheduling Problem*. Nous souhaitons apporter une contribution vers l'élaboration à terme d'un système d'aide à la décision qui permettra à des industriels de choisir certains paramètres appropriés pour leur chaîne de production tout en tenant compte des coûts d'investissement et des objectifs de productivité. En outre, la complexité du problème augmente puisque nous élargissons le périmètre du problème classiquement étudié et nous lui ajoutons une nouvelle variable (le nombre des ressources de transport). Par conséquent, pour appréhender cette difficulté additionnelle, nous développons des méthodes de résolution adaptées et efficaces.

La suite de ce manuscrit s'articule de la manière suivante, afin de rendre compte du travail de thèse effectué et de présenter les approches et méthodes proposées pour traiter conjointement les deux aspects de dimensionnement et d'ordonnancement du problème.

Dans le premier chapitre, nous décrivons les lignes de traitement de surface ainsi que les problèmes d'ordonnancement d'atelier, notamment ceux liés à ces lignes spécifiques. Nous présentons un état de l'art des travaux de recherche associés, dont une classification des problèmes d'ordonnancement des robots dans les ateliers de galvanoplastie, et une énumération des différentes méthodes de résolution appliquées au CHSP. Nous nous focalisons en particulier sur les recherches associées aux ateliers comportant plusieurs ressources de manutention, ainsi qu'aux rares études dédiées à la conception de ces lignes. Cela nous permet d'introduire et de situer nos travaux dans la littérature du domaine.

Dans le second chapitre, nous explicitons le problème faisant l'objet de nos travaux, le CHDSP. Nous décrivons la méthode de codage adoptée pour représenter une solution, basée sur une amélioration d'un codage issu de travaux précédents. Cette amélioration autorise une exploration plus complète de l'espace de recherche. Nous présentons ensuite une première méthode de résolution qui fournit des résultats préliminaires confirmant l'efficacité de cette stratégie d'amélioration.

Dans le troisième chapitre, nous fournissons une description exhaustive d'une seconde méthode de résolution adaptée et plus efficace que la précédente, pour résoudre le CHDSP. Cette dernière repose sur la recherche à voisinage variable adaptative (aVNS : *adapted Variable Neighborhood Search*). Nous déployons la procédure et les résultats menant à l'ajustement des paramètres de notre algorithme de résolution, et nous finissons par une analyse et une comparaison des résultats de cette approche par rapport à d'autres résultats de la littérature.

Dans le quatrième chapitre, nous proposons un nouveau modèle mathématique pour l'évaluation des solutions du CHDSP, avec prise en compte des contraintes

spatiales liées aux déplacements des ressources de manutention. Les contraintes spatiales imposent une interdiction des collisions entre les robots qui partagent la même voie de déplacement sur la ligne de production multi-robot. Le modèle est formulé par un programme linéaire en nombres entiers mixtes (MILP : *Mixed Integer Linear Programming*). Il est validé par des expérimentations menées sur les meilleures solutions issues de la mise en oeuvre de nos algorithmes.

Nous terminons ce manuscrit par une conclusion qui propose un bilan de nos travaux et ouvre quelques perspectives de recherche.

Chapitre 1

Ordonnancement d'atelier de traitement de surface

Les travaux de cette thèse s'inscrivent dans le cadre des problèmes d'ordonnancement. Ce chapitre propose un état de l'art du domaine. Il introduit les notions liées à ces problèmes et à leur complexité. Il se concentre sur l'ordonnancement dans les ateliers, en particulier dans les lignes de traitement de surface, connu dans la littérature sous le nom Hoist Scheduling Problem, ou HSP. Il définit les classes de problèmes d'ordonnancement spécifiques rencontrées dans ce type d'atelier, ainsi que la notation associée. Il décrit les méthodes de résolution déjà développées pour la variante cyclique qui nous intéresse dans ses deux déclinaisons, monorobot et multirobot. Ainsi, ce chapitre permet de positionner nos travaux par rapport aux contributions de la littérature, notamment en termes de spécificités du problème étudié.

Sommaire

1.1 Les problèmes d'ordonnancement

Respect des délais et réactivité sont devenus des armes de compétitivité industrielle et de croissance. Les entreprises doivent être en perpétuelle évolution pour pouvoir affronter les vicissitudes et le dynamisme du marché, et concurrencer leurs rivaux avec de meilleurs biens ou services. Il peut s'agir de petites améliorations continues de toutes les activités de l'entreprise pour tenter d'obtenir un avantage concurrentiel [Hitomi 1996]. Cela passera sans doute par l'optimisation de la productivité, qui ne sera garantie que par un ordonnancement sûr et performant des opérations de production. Des méthodes de résolution efficaces sont alors nécessaires pour résoudre les problèmes d'ordonnancement qui sont rencontrés dans tous les secteurs d'activité économique, aussi bien ceux offrant des services tels que la santé, l'informatique, la distribution, l'administration, etc., que ceux produisant des biens comme l'industrie manufacturière. Nous nous intéressons particulièrement dans cette thèse aux problèmes d'ordonnancement présents dans les chaînes de production industrielles. Dans la suite, nous présentons quelques notions de base, utiles pour comprendre le concept général de l'ordonnancement.

1.1.1 Concept et définitions

L'ordonnancement est le procédé par lequel on donne des priorités successives à des tâches différentes. Selon [Carlier *et al.* 1993], « Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début ». Antérieurement, dans [Carlier & Chretienne 1982], les auteurs donnent une définition un peu plus détaillée : « Ordonnancer c'est programmer dans le temps l'exécution d'une réalisation décomposable en tâches, en attribuant des ressources à ces tâches et en fixant en particulier leurs dates de début d'exécution tout en respectant des contraintes données ».

Dans le domaine industriel, l'ordonnancement consiste à affecter une suite de tâches à des ressources disponibles et à organiser leur réalisation dans le temps, en respectant les contraintes de production, et en optimisant des critères définis en fonction des objectifs à atteindre. Une solution à un problème d'ordonnancement peut être exprimée sous forme d'un planning d'exécution des tâches et d'allocation des ressources qui cible la satisfaction d'un ou de plusieurs objectifs. Notons que pour caractériser un problème d'ordonnancement, quatre éléments sont à définir : les tâches, les ressources, les contraintes et les objectifs. Nous les expliquons un par un dans les paragraphes suivants.

1.1.1.1 Les tâches

Une tâche est un travail élémentaire localisé dans le temps par deux dates réelles de début et/ou de fin. Sa réalisation nécessite une durée (un certain nombre d'unités

de temps) et aussi un certain nombre d'unités de chaque ressource. Souvent, d'autres dates liées aux tâches sont définies : la date de disponibilité qui correspond à la date à laquelle l'exécution de la tâche peut commencer, et la date échue (ou aussi dite date « due »), qui n'est que la date à laquelle la tâche doit être achevée. La première induit une contrainte sur le début de la tâche, tandis que la seconde conduit à une contrainte de finalisation de celle-ci, qui non respectée, mène à des pénalités.

Deux tâches peuvent être liées entre elles par des relations d'antériorité. On dit aussi qu'elles sont soumises à des contraintes de précédence. Dans le cas où ces relations sont absentes, les tâches sont dites indépendantes. En outre, d'un problème à l'autre, une tâche peut être exécutée par morceaux ou, doit être exécutée sans interruption ; dans le premier cas, la tâche est dite préemptive ou encore morcelable ; dans le second, elle est dite non préemptive.

1.1.1.2 Les ressources

Une ressource est un moyen, humain ou technique, nécessaire dans la réalisation d'une tâche. On reconnaît deux types de ressources : les ressources renouvelables et les ressources consommables [Słowinski 1981]. La ressource renouvelable est celle qui, après avoir été allouée à une tâche, redevient à nouveau disponible pour les autres tâches. On en donne comme exemples, les hommes, les machines, les équipements, etc. La ressource consommable est celle qui, après avoir été attribuée à une tâche, n'est plus disponible pour les autres tâches. Comme exemples, on cite les matières premières, l'argent, etc. Qu'elle soit renouvelable ou consommable, une ressource a une disponibilité qui peut varier au cours du temps. On connaît alors a priori sa courbe de disponibilité.

Parmi les ressources renouvelables, on distingue encore principalement deux types : les ressources disjonctives et les ressources cumulatives. Les premières sont celles qui ne peuvent exécuter qu'une seule tâche à la fois telles que la machine-outil, le robot manipulateur, etc. Quant aux ressources cumulatives, en revanche, elles peuvent exécuter plusieurs tâches simultanément, mais en nombre limité, telles qu'une équipe d'ouvriers ou un poste de travail, etc.

1.1.1.3 Les contraintes

Une contrainte est une règle qui régit les tâches et/ou les ressources. Elle exprime une restriction sur les valeurs que peuvent prendre simultanément les variables de décision [Esquirol & Lopez 2001]. On distingue des contraintes temporelles et des contraintes de ressources.

Dans la première catégorie, on trouve les contraintes de temps alloué. Elles sont généralement liées aux impératifs de la production (délais requis, retards, priorités) et sont relatives aux dates limites des tâches, à savoir, les dates de disponibilité et les

dates dues. On trouve aussi les contraintes de précédence qui traduisent des relations d'ordre entre les tâches, et elles dépendent notamment de la gamme opératoire.

Dans la seconde catégorie, les contraintes d'utilisation expriment la nature et la quantité des ressources utilisées par tâche, ainsi que leurs caractéristiques d'utilisation. On y trouve des contraintes disjonctives liées aux ressources disjonctives, et des contraintes cumulatives imposant, à une ressource cumulative, le nombre possible de tâches à réaliser simultanément et dans la limite de sa capacité. Il y a, en outre, les contraintes de disponibilité qui définissent la nature et la quantité des ressources disponibles au cours du temps.

1.1.1.4 Les objectifs

Les objectifs d'un problème d'ordonnancement sont soit maximiser soit minimiser la fonction économique, qui est une fonction regroupant un ou plusieurs critères d'évaluation. D'après [Carlier *et al.* 1993], les critères les plus courants dans l'évaluation d'une solution sont :

- le délai global de l'ordonnancement ou sa durée totale, également appelé *makespan* et noté souvent C_{max}. Il est égal à la date d'achèvement de la tâche qui finit le plus tardivement ;
- le plus grand retard résultant, lors de la réalisation de toutes les tâches ;
- le retard moyen pondéré, qui est la somme pondérée de tous les retards ;
- le nombre de tâches en retard ;
- les stocks d'encours.

Il en découle des objectifs tels que la minimisation du *makespan*, qui n'est rien que la maximisation du débit ou taux de rendement, et le respect des dates dues qui se traduit par la minimisation du plus grand retard, ou du retard moyen pondéré, ou du nombre de tâches en retard, etc. Notons que tous ces objectifs sont liés à la notion du temps. Par ailleurs, nous pouvons bien évidemment rencontrer des problèmes d'ordonnancement ([Carlier *et al.* 1993]) avec des objectifs liés aux ressources. Ces derniers peuvent cibler la minimisation de la quantité totale ou pondérée de ressources nécessaires pour réaliser un ensemble de tâches ou la maximisation du taux d'utilisation de chaque ressource. D'autres objectifs peuvent être liés aux coûts de production tels que la minimisation du coût des stocks d'encours, ou aux coûts de lancement ou de transport, etc.

A partir de ces données, nous pouvons nous rendre compte de la grande diversité des problèmes d'ordonnancement. La table 1.1 propose une typologie de ces problèmes.

1.1.2 Complexité et approches de résolution

Une fois le problème d'ordonnancement défini avec ses objectifs et contraintes, et une fois sa fonction économique à optimiser déterminée, il convient de considérer

TABLE 1.1 – Typologie des problèmes d'ordonnancement [Carlier *et al.* 1993].

Caractéristiques	Choix possibles (non exclusifs)				
Quantité de moyens (nature)	illimitée		limitée (renouvelables)		limitée (renouvelables)
Organisation des moyens	une machine	m machines en parallèle	machines en série "flow-shop"	"job-shop"	gammes non linéaires
Contraintes de fabrication	préemption autorisée	chevauchement		temps de transit	encours limités
Types de fabrication	unitaire	atelier (petite série)		masse (grande série)	continue
Répétition des travaux	cyclique ou répétitif			à la commande ou non répétitif	
Hypothèses sur la demande	statique (connue a priori)			dynamique (arrivée continue en temps réel)	
Critères principaux d'appréciation des solutions	minimisation du cycle de fabrication		minimisation des encours		minimisation des retards

sa taille et sa complexité selon lesquelles il sera possible de choisir la méthode de résolution à mettre en œuvre, ou du moins, la catégorie des méthodes de résolution la plus adéquate pour résoudre ce problème.

En recherche opérationnelle, il existe une variété de méthodes de résolution pour l'optimisation combinatoire, qui sont réparties en deux grandes familles : les méthodes exactes et les méthodes approchées (voir figure 1.1). La première catégorie est qualifiée d'exacte car elle garantit la complétude (l'optimalité) de la résolution. Néanmoins, elle peut parfois nécessiter des temps de calcul prohibitifs. Nous citons comme exemples la méthode de séparation et évaluation (Branch and Bound) [Land & Doig 1960] ou sa version améliorée, la méthode de Branch and Cut [Mitchell 2010]. La deuxième catégorie, quant à elle, ne peut pas garantir cette complétude, mais elle est souvent capable de fournir des solutions approchées de bonne qualité, et ce en temps raisonnable, comparée à la première famille de méthodes. Cette catégorie est divisée, elle-même, en deux familles : les heuristiques, qui sont conçues et appliquées pour résoudre un type spécifique de problèmes, et les métaheuristiques, dont l'application est plus générale. Cette dernière famille est aussi subdivisée en deux catégories : les méthodes de recherche locale [Aarts *et al.* 2003] et les méthodes évolutionnaires [Coello 2005]. La première regroupe des méthodes à solution unique telles que la méthode de descente (Hill Climbing), la recherche tabou [Glover 1990b], le recuit simulé [Kirkpatrick *et al.* 1983], etc. La deuxième caractérise les méthodes à base de population de solutions, dont une des plus populaires est l'algorithmique génétique [Coello 2005].

Les méthodes de résolution des problèmes d'ordonnancement puisent dans l'éventail des techniques de l'optimisation combinatoire. Quand le problème est de complexité réduite (de petite taille), l'application de l'une des méthodes exactes peut suffire pour aboutir à une solution optimale. Dans le cas contraire, le recours aux méthodes approchées et sous-optimales est nécessaire pour obtenir des solutions.

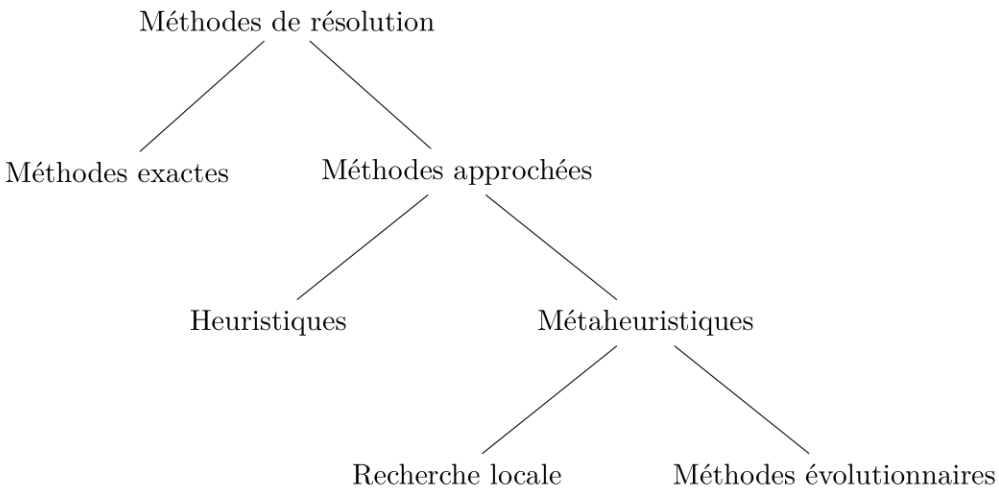

FIGURE 1.1 – Classification des méthodes d'optimisation combinatoire.

L'objectif ainsi est de trouver des solutions proches de l'optimal, sans qu'il soit toujours possible de le garantir. Dans le cas de problèmes de grande taille, les algorithmes de complexité non-polynomiale ne peuvent pas être utilisés, d'où le besoin de construire ces méthodes approchées, de complexité polynomiale, pour la résolution des problèmes NP-difficiles [Garey & Johnson 1979].

La complexité des problèmes d'ordonnancement est définie suivant la complexité des méthodes de résolution et celle des algorithmes utilisés [Tangour 2007]. Quand la taille de ces problèmes devient relativement importante, la combinatoire devient aussi très importante, ce qui rend la résolution très difficile. Nous distinguons deux catégories de complexité : la complexité du problème et la complexité algorithmique.

1.1.2.1 Complexité du problème

La complexité d'un problème est un résultat du problème à résoudre et de la méthode de résolution à mettre en œuvre pour construire la solution optimale. Opérationnellement parlant, la complexité d'un problème est une estimation du nombre d'instructions à exécuter, pour résoudre les instances de ce problème [Solnon 2010]. Cette estimation est un ordre de grandeur par rapport à la taille de l'instance la plus difficile de ce problème.

Selon la littérature [Charon *et al.* 1996], les problèmes d'ordonnancement se divisent en deux catégories qui caractérisent leur degré de complexité :

— Les problèmes indécidables, qui sont les problèmes d'ordonnancement les plus difficiles. Aucune méthode de résolution n'existe pour ce type de problèmes ;

— Les problèmes décidables, pour lesquels il existe les classes P et NP.

Selon [Papadimitriou 1994], différentes classes de problèmes peuvent être identifiées en fonction de la complexité de leur résolution. Nous distinguons :

— Les problèmes de décision, pour lesquels les classes de complexité ont été introduites. Ce sont les problèmes qui comprennent deux parties : une partie données du problème et une question binaire dont la réponse possible est « oui » ou « non ».

— Les problèmes de recherche qui sont constitués d'ensembles de données. À chaque ensemble de données correspond un ensemble de solutions. Pour résoudre un problème de recherche, il faut calculer, pour chaque ensemble de données D, l'ensemble des solutions $S(D)$ lui sont associées.

— Les problèmes d'optimisation qui sont à la base des problèmes de recherche, mais qui affectent à chaque solution une valeur quantitative. On parle d'une valeur minimale si on cherche à minimiser la fonction économique, ou d'une valeur maximale dans le cas contraire. À chaque problème d'optimisation on peut associer un problème de décision [Charon *et al.* 1996]. Ainsi, par inférence, la complexité d'un problème de décision reflète la complexité relative au problème d'optimisation associé.

Comme déjà évoqué, les classes de complexité ont été introduites pour les problèmes de décision :

— Un problème de décision est dit polynomial et est de classe P, s'il existe un algorithme polynomial pour le résoudre.

— Un problème de décision est dit polynomial non déterministe et est de classe NP, s'il ne peut pas être résolu en un temps polynomial par les algorithmes déterministes [Carlier & Chrétienne 1988], mais peut être résolu en un temps polynomial par des méthodes approchées. Autrement dit, pour résoudre ce type de problèmes, on doit examiner un nombre grandissant de cas, éventuellement exponentiel, qui ne peut pas être fait dans un temps polynomial, mais l'examen de chacun de ces cas reste faisable en temps polynomial.

En outre, un problème de décision est dit *NP-complet*, s'il appartient à la classe *NP* et s'il est résolu, au mieux, en un temps exponentiel. Enfin, un problème d'optimisation est dit *NP-difficile* si le problème de décision associé est *NP-complet*.

1.1.2.2 Complexité algorithmique

La théorie de la complexité vise à analyser les coûts de résolution des problèmes d'optimisation combinatoire, notamment en termes de temps de calcul. Elle permet de classer ces problèmes en plusieurs niveaux de difficulté, et fait la distinction entre un problème d'optimisation et un problème de décision [Carlier & Chrétienne 1988]. Il a été montré que la plupart des problèmes d'ordonnancement sont difficiles [Lopez & Roubellat 2001].

La complexité algorithmique, ainsi, se mesure en fonction du temps alloué à l'exécution de l'algorithme ou encore en fonction de l'espace mémoire requis. Le

temps de calcul est une fonction du nombre d'instructions à exécuter et de la taille des données manipulées.

1.1.3 Ordonnancement d'atelier

Dans la littérature, il existe différents types de problèmes d'ordonnancement dits d'atelier. [Esquirol & Lopez 2001] considèrent que dans ce type de problèmes, les ressources sont des machines ne pouvant réaliser qu'une seule tâche ou opération à la fois. Les problèmes d'ordonnancement d'atelier se distinguent par plusieurs facteurs comme le type d'atelier considéré, la nature ou le nombre des ressources déployées, la manière d'enchaîner les opérations, etc. Ainsi, plusieurs classifications de ces problèmes ont été établies dans la littérature en se référant à différents critères. La figure 1.2 schématise une classification de ces problèmes en fonction des ressources, proposée par [Esquirol *et al.* 1999]. Notons que la majorité des problèmes d'atelier étudiés sont à ressources renouvelables, disjonctives, uniques ou multiples.

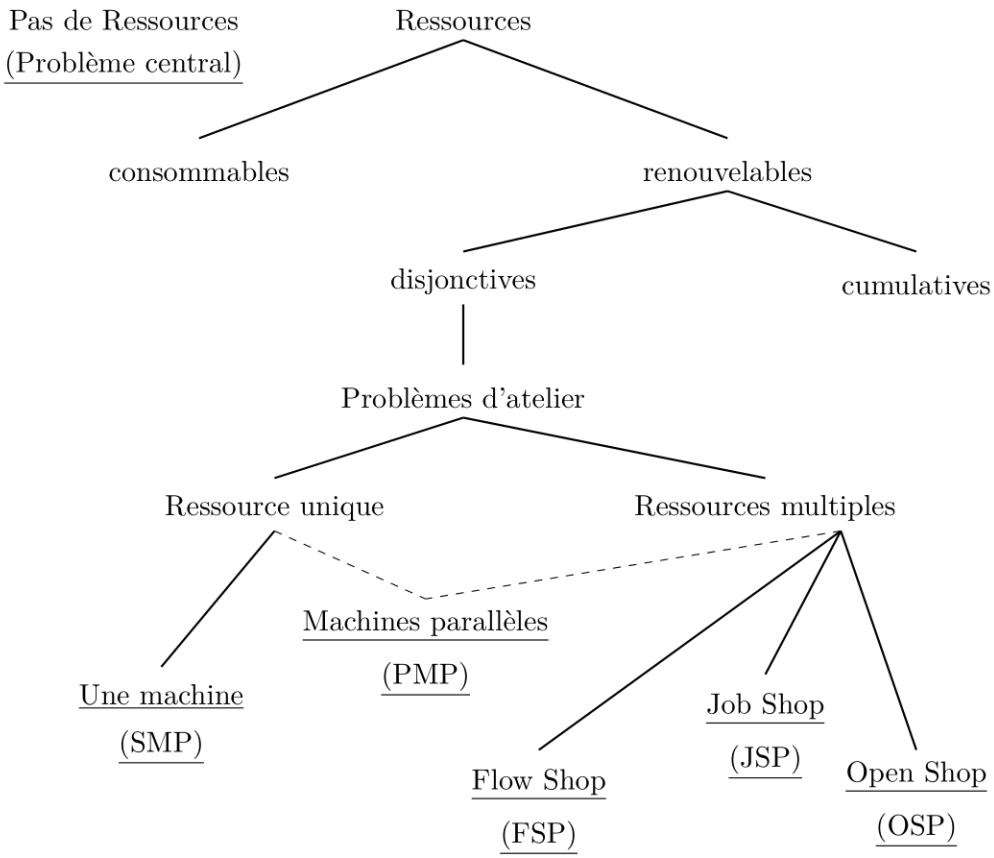

FIGURE 1.2 – Classification des problèmes d'ordonnancement d'atelier en fonction des ressources.

1.1.3.1 Ateliers à ressources uniques ou multiples

Certains types d'ateliers ont été définis selon le nombre de machines qu'ils contiennent [Chen *et al.* 1998a] et le type de travaux à réaliser pour fabriquer un produit. Pour ces différents types d'ateliers possibles, différents problèmes peuvent être posés [Carlier & Chrétienne 1988] :

— Problème à une machine [Gupta & Kyparisis 1987, Koulamas 2010] : chaque travail n'est constitué que d'une seule opération, réalisable sur une seule machine (ressource de traitement). Ce sont les problèmes d'ateliers à ressources uniques (voir figure 1.2).

— Problème à machines parallèles [Cheng & Sin 1990, Mokotoff 2001] : une ou plusieurs machines, effectuant les mêmes opérations, sont dupliquées. Elles remplissent donc les mêmes fonctions. On parle de systèmes avec flexibilité de ressources, puisque plus d'une seule ressource peut exécuter une même opération. Cependant, il en résulte un problème supplémentaire qui est celui de l'affectation, et qui pose la question : quelle ressource parmi les exemplaires identiques sera la responsable de quelle opération ? Ainsi, le problème d'ordonnancement consiste en un problème d'affectation et un problème de séquencement, souvent résolus l'un après l'autre. Sur la figure 1.2, ce type de problèmes apparaît bien en croisement entre les problèmes d'ateliers à ressources uniques et ceux à ressources multiples.

Dans les problèmes à machines parallèles, il existe encore une classification des ateliers en fonction de la vitesse de ces machines :

— Ateliers à machines identiques [Tahar *et al.* 2006, Biskup *et al.* 2008] : toutes les machines disposent de la même vitesse d'exécution pour tous les travaux.

— Ateliers à machines uniformes [Koulamas & Kyparisis 2000, Yeh *et al.* 2015] : les vitesses de ces machines sont proportionnelles, avec un coefficient de proportionnalité. Cependant, chaque machine a une vitesse d'exécution propre et constante pour tous les travaux.

— Ateliers à machines indépendantes [Kim *et al.* 2002, Logendran *et al.* 2007] : chaque machine a une vitesse d'exécution différente et indépendante des autres, pour chaque travail. Cela influence donc la durée opératoire d'un travail qui dépend du travail réalisé lui-même et de la machine qui en est responsable.

1.1.3.2 Ateliers à cheminements uniques ou multiples

Par ailleurs, d'autres types d'ateliers ont été définis selon le type d'enchaînement ou du cheminement des opérations, ce qui a donné naissance à une autre classification des problèmes d'ordonnancement d'atelier :

— Problème de *flow-shop* [Cheng *et al.* 2000, Reza Hejazi & Saghafian 2005, Yenisey & Yagmahan 2014] : la ligne de production se compose de plusieurs machines en série. Tous les produits visitent les machines dans le même et unique ordre.

Les durées opératoires peuvent être différentes, mais le flux des produits est unidirectionnel. Ce type d'atelier est aussi dit à cheminement unique. Chaque machine n'exécute qu'une seule opération à la fois, et chaque opération ne peut s'exécuter à la fois que sur une seule machine.

— Problème de *job-shop* [Manne 1960, Błażewicz *et al.* 1996, Zhang *et al.* 2019] : les opérations sont réalisées sur les machines selon un ordre fixé bien déterminé, qui dépend de la gamme de chaque produit. Cet ordre varie ainsi d'un produit à l'autre, donc plusieurs changements d'outils sont à envisager. Ce type d'atelier est aussi dit à cheminements multiples.

— Problème d'*open-shop* [Gonzalez & Sahni 1976, Anand *et al.* 2015] : les gammes de fabrication ne sont pas linéaires, et il n'y a aucun ordre de fabrication qui est imposé. Le cheminement de toutes les opérations est ainsi libre et multiple. L'exécution de ces dernières peut se faire dans n'importe quel ordre.

Suivant le nombre et le type des produits à traiter, on distingue différents types de production, d'où la différence des méthodes de gestion de l'atelier [Mangione 2003].

D'autres extensions de ces problèmes ont aussi été étudiées dans la littérature, telles que le problème de *flow-shop* hybride [Ruiz & Vázquez-Rodríguez 2010, Ribas *et al.* 2010] ou encore le problème de *job-shop* flexible [Chaudhry & Khan 2016, Gao *et al.* 2019]. Le *flow-shop* hybride est une extension du *flow-shop* classique, dans lequel les machines sont disponibles en plusieurs exemplaires. Chaque travail contient des opérations différentes, et chaque opération doit être affectée à une machine parmi l'ensemble des mêmes exemplaires. Cet ensemble est souvent appelé étage, contenant des machines parallèles dédiées à cette opération. Ainsi, chaque machine ne peut appartenir qu'à un seul étage. Le *job-shop* flexible est une extension du problème *job-shop* classique. Il est défini pour un atelier où plusieurs machines sont capables de réaliser un sous-ensemble d'opérations. Autrement dit, une opération est associée à un ensemble de machines qui sont toutes capables de la réaliser.

1.1.3.3 Ateliers avec ressources de transport

Dans certains types d'ateliers, les durées de traitement des pièces priment sur les durées de transport de ces pièces entre les postes de travail. Par conséquent, les durées de transport sont négligées et ne rentrent pas dans les modélisations élaborées pour résoudre le problème. C'est d'ailleurs le cas des problèmes rencontrés dans les classes d'ateliers citées précédemment.

Or, de manière générale, dans les ateliers automatisés, le transfert des produits entre machines ou postes de travail requiert des moyens de transport tels que des robots, des chariots filoguidés, etc. Dans ces ateliers avec ressources de transport, les durées de manutention ne peuvent pas être négligées car elles sont parfois au moins aussi importantes que les durées opérationnelles sur les machines. Elles doivent donc être prises en compte dans la modélisation du problème considéré. Dans la

littérature associée, nous citons trois problèmes : le problème d'ordonnancement dans les Systèmes Flexibles de Production (SFP) ou (FMSSP pour la version anglaise : *Flexible Manufacturing Systems Scheduling Problem*), le problème d'ordonnancement dans les cellules robotisées (*Robotic Cells Scheduling Problem*) et le problème d'ordonnancement dans les ateliers de traitement de surface (*Hoist Scheduling Problem* ou *HSP*). Nous détaillons ci-après quelques-unes des principales caractéristiques des systèmes concernés :

— Système flexible de production ou *Flexible Manufacturing System* [Buzacott & Yao 1986, Basnet & Mize 1994, Candan & Yazgan 2015] : le déplacement des produits se fait en général via des chariots automatiquement guidés ou chariots autonomes (figure 1.3) dits AGV (*Automated Guided Vehicle* [Thomas 1983]). Une spécificité de ce type d'atelier concerne les durées opératoires sur les machines qui sont soit non bornées, soit bornées seulement par une borne inférieure. Il faut noter que certains FMSs sont constitués de plusieurs cellules robotisées, entre lesquelles se déplacent les AVGs.

FIGURE 1.3 – Un atelier de type FMS.

— Cellule robotisée [Dawande *et al.* 2005, Gultekin *et al.* 2008] : des robots sont implantés pour le chargement et le déchargement des pièces sur et depuis les machines (figure 1.4). Ces robots sont soit fixes, soit mobiles sur des rails pour le transport des pièces entre machines.

— Lignes de traitement de surface [Bloch *et al.* 1997, Manier & Bloch 2003] : des robots, pouvant être suspendus à un rail unique et unidirectionnel, sont en charge du transfert des pièces entre les postes de travail (cuves) (figure 1.5). Ici, en revanche, les durées opératoires (trempe dans les cuves) sont bornées par limites inférieures et supérieures.

Dans cette thèse, nous nous intéressons particulièrement au problème dernièrement mentionné, le *Hoist Scheduling Problem* (HSP), et plus spécifiquement, à sa

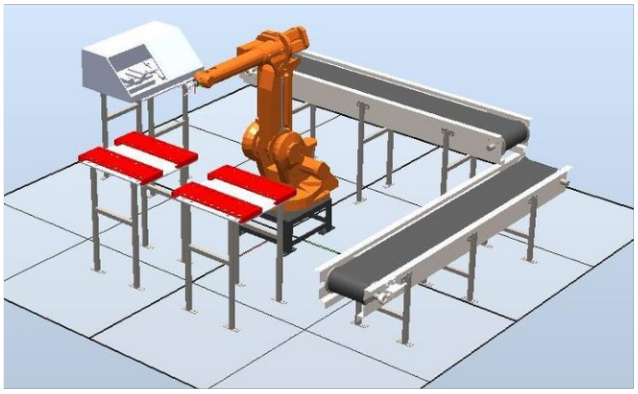

FIGURE 1.4 – Une cellule robotisée.

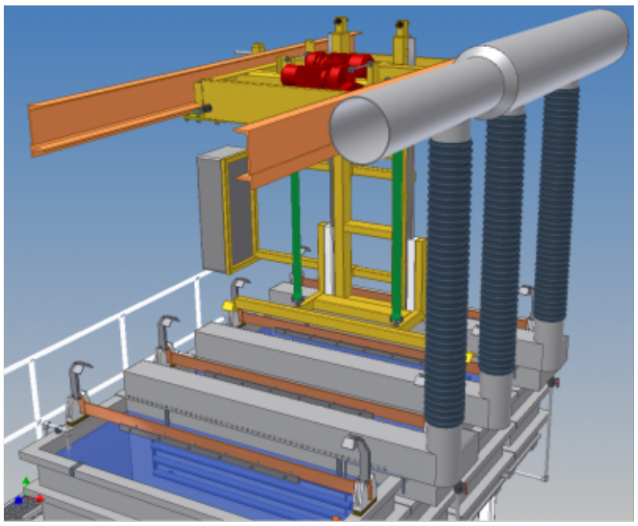

FIGURE 1.5 – Une ligne de traitement de surface.

variante cyclique (CHSP). Le HSP est décrit plus en détails dans le paragraphe 1.3. Les ateliers de traitement de surface sont présentés dans le paragraphe suivant.

1.2 Les ateliers de traitement de surface

Dans la fabrication des pièces, une gamme de fabrication décrit les phases et les étapes par lesquelles les pièces doivent passer pour leur élaboration. Ces phases peuvent agir sur les propriétés physiques, mécaniques, électriques, chimiques ou électrochimiques des pièces. Comme les cellules robotisées sont principalement dédiées à l'usinage des pièces, les ateliers de traitement de surface sont destinés à des opérations de traitement chimiques ou électrochimiques, qui modifient l'état surfacique des pièces. Ces traitements sont généralement réalisés au milieu de la gamme de fabrication d'un produit, entre l'usinage et le montage [Mangione 2003]. Les opérations accomplies dans ce type d'atelier consistent à tremper des pièces, de

manière successive, dans des cuves contenant des solutions chimiques (figure 1.6). Ces dernières agissent sur les caractéristiques du matériau dont se composent les pièces, permettant ainsi de réaliser différents traitements désirés, tels qu'un dépôt de cuivre, d'argent ou d'or, une attaque acide, un rinçage, etc.

FIGURE 1.6 – Ateliers de traitement de surface.

Les lignes de traitement de surface peuvent se trouver dans l'industrie mécanique pour effectuer des opérations de trempe, dans l'industrie électronique pour la fabrication des cartes de circuits imprimés, ou tout simplement dans les cas où il est nécessaire de revêtir une pièce d'une fine couche de métal tel que cuivre, argent ou autres. Ces traitements offrent ainsi de nouvelles propriétés désirées de la surface des pièces, comme la conductivité électrique, la résistance à la rouille, la protection contre l'oxydation et la corrosion, ou même des qualités esthétiques grâce notamment à la dorure, au chromage, au dépôt d'argent, etc.

En général, une ligne de traitement de surface se compose de plusieurs cuves qui contiennent différents bains de traitement chimiques et/ou électrolytiques. Pour traiter une pièce, il faut suivre sa gamme opératoire qui consiste à enchaîner des traitements spécifiques réalisés dans les cuves correspondantes. Pour déplacer les pièces entre cuves, un ou plusieurs robots sont nécessaires, selon le besoin de la production et/ou la configuration de la ligne. Les pièces sont généralement chargées sur des porteurs, pour être traitées par lots. Le ou les robots circulent le long d'un rail, suspendu au-dessus de la ligne des cuves, permettant ainsi les déplacements

requis. Un robot ici est aussi appelé palan ou *hoist* en anglais (figure 1.7).

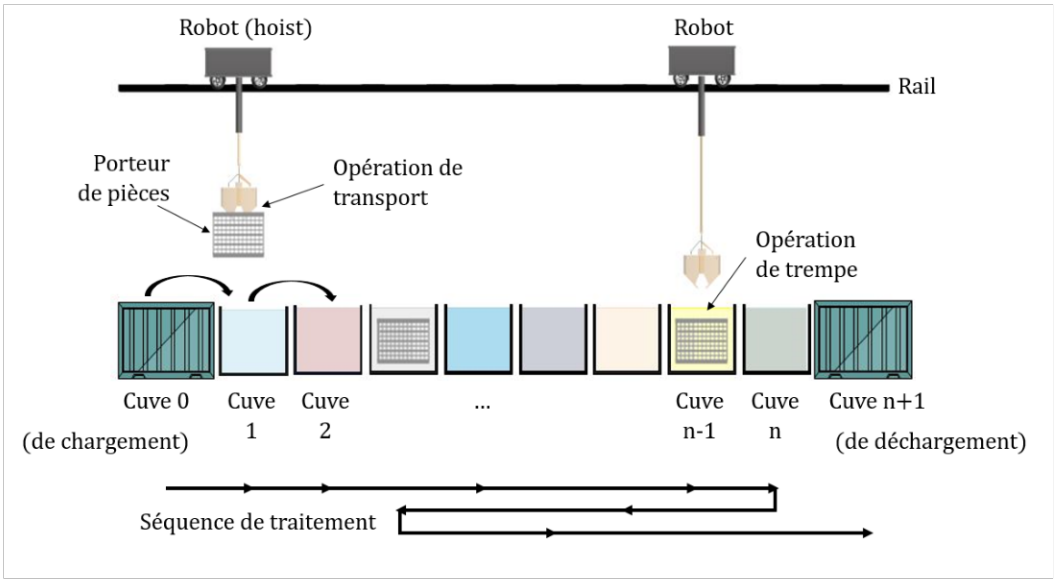

FIGURE 1.7 – Schéma d'une ligne de traitement de surface.

Les ateliers de traitement de surface peuvent être composées de lignes simples ou complexes. Selon le nombre de cuves, le nombre de robots, ou la taille de l'atelier lui-même, la disposition de ces lignes s'adapte. Une ligne *simple* est celle qui emploie un seul robot circulant sur une seule rangée de cuves de traitement. Ses ressources de traitement sont dites *monobacs* (cuves de capacité unitaire) et *monofonctions* (un produit ne passe au plus qu'une seule fois dans chaque cuve). Une ligne complexe peut être une ligne simple comportant plusieurs robots (figure 1.7), dans l'objectif d'accroître la productivité. On parle ici de ligne *multirobot*. Dans ce cas, les robots se partagent les tâches de transfert des produits entre les cuves, et donc doivent se déplacer sur le même rail au-dessus de la ligne, ce qui implique d'éventuelles collisions. Ainsi, il s'avère nécessaire de gérer le problème supplémentaire de collisions et d'éviter tout croisement entre robots. Parfois, le nombre de cuves peut être important et leur disposition en une seule ligne simple n'est donc pas possible, en tenant compte de l'espace alloué. Alors, les lignes sont disposées en parallèle les unes aux autres, formant ainsi des lignes complexes. La liaison entre ces lignes parallèles se fait via des systèmes de transport éventuellement autres que les robots. Il s'agit de chariots de transfert ou tout simplement des cuves de transfert, contenant souvent des produits neutres, et servant à transférer les porteurs de pièces entre les lignes parallèles. Dans ce cas, il est parfois nécessaire de synchroniser l'arrivée du système de transfert utilisé avec l'arrivée d'un des robots pour permettre à ce dernier de déposer ou de collecter un des porteurs. Le porteur collecté doit continuer son traitement sur la ligne desservie par ce robot, tandis que le porteur posé est transporté via le chariot ou la cuve de transfert pour continuer à être traité sur une autre ligne de cuves. [Jullien 1999] a étudié un atelier réel complexe avec trois lignes parallèles,

sept robots et trois cuves de transfert. La figure 1.8 [Zhang 2012] montre un exemple de lignes complexes.

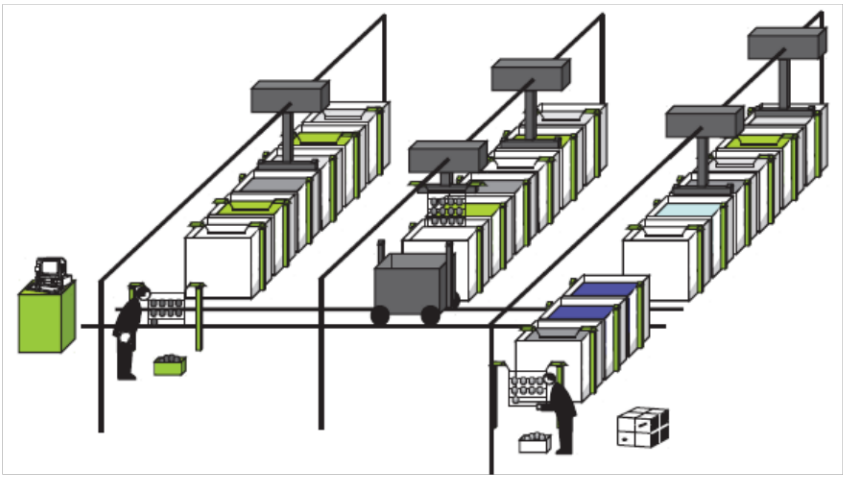

FIGURE 1.8 – Un atelier de traitement de surface complexe.

Dans les ateliers complexes, on peut rencontrer trois implantations différentes : en forme de U, de O ou de H, comme l'expose la figure 1.9.

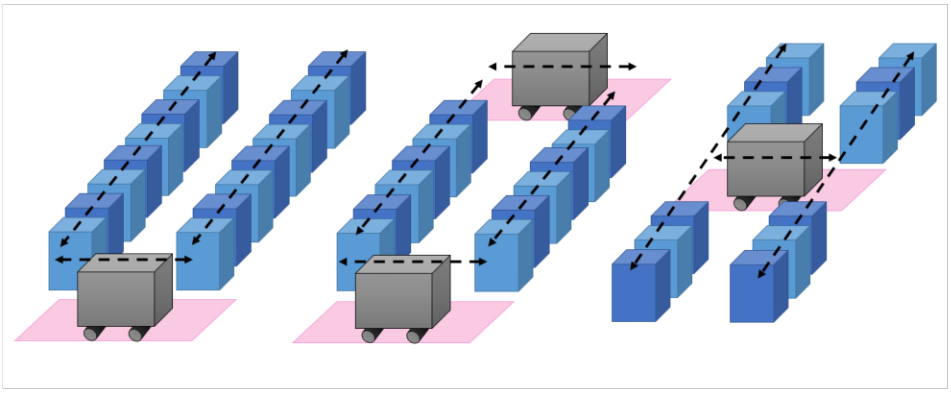

FIGURE 1.9 – Différents types d'implantation.

1.2.1 Les porteurs

Dans les ateliers de traitement de surface, les pièces peuvent être traitées en très grandes quantités, ce qui fait que leur traitement par lots est beaucoup plus raisonnable et économique en temps qu'un traitement pièce par pièce. Les pièces sont aussi souvent de petites tailles, ce qui rend leur préhension par les robots difficile. Ainsi, pour faciliter leur transport tout au long de la ligne, leur immersion et leur évacuation des bains mis à disposition, elles sont chargées sur des porteurs en entrée de ligne, pour être traitées par lots. On trouve plusieurs types de porteurs suivant la nature des pièces à transporter :

— Des cadres, utilisés pour le transport de pièces de taille moyenne telles que les supports de cartes pour les circuits imprimés, ou encore les tiges d'amortisseurs de voiture. Ces cadres disposent de multiples crochets sur lesquels les pièces sont accrochées ou vissées.

— Des tonneaux, qui servent pour le déplacement des pièces de petite taille telles que les antennes de téléphone portable, les boulons, ou les vis. Ces tonneaux sont des cylindres permettant de contenir les pièces, et une fois immergés dans des bains, ils se mettent en rotation autour de leur axe, pour permettre un traitement efficace de toute la surface des pièces.

— Des paniers, aussi destinés pour le transfert des pièces de petite taille. Il s'agit simplement de récipients remplis de pièces qui vont être immergés dans les cuves.

1.2.2 Les cuves

Les cuves sont les ressources qui contiennent les solutions chimiques nécessaires au traitement des pièces. Les porteurs y sont immergés pour une durée donnée, qui est souvent bornée par une ou deux limites temporelles. Un porteur doit impérativement passer un temps de traitement minimal dans chaque cuve avant d'en être retiré. Ceci conditionne le bon déroulement des opérations et garantit l'efficacité du traitement souhaité. Également, dans certains cas, le porteur ne doit pas excéder une durée maximale de séjour dans les cuves, pour éviter ainsi un sur-traitement. Dans les deux cas, le non-respect de ces durées risque de produire des pièces défectueuses et donc de dégrader la qualité attendue du traitement. Ces limites temporelles changent bien évidemment d'une cuve à une autre et elles sont donc dépendantes du type de traitement réalisé et de la gamme opératoire. Elles sont généralement définies par des experts électriciens ou chimistes. Les traitements réalisés sont de type galvanoplastie, qui visent à appliquer des dépôts sur les pièces. Ces dépôts peuvent avoir un objectif de protection (chromage, galvanisation), de décoration (argenture, dorure, nickelage) ou de reproduction de pièces métalliques à partir d'un moule en cire, qui est rendu conducteur au moyen d'un vernis [Mangione 2003]. Il y a recours à la galvanoplastie dans plusieurs domaines, tels que l'industrie électrique (bras de disjoncteurs) ou électronique (antennes de téléphones portables), la bijouterie (bouchons de parfums) ou la protection anti-corrosion (visserie).

La complexité des lignes se caractérise également par le type de cuves qui la constituent. Ainsi, on distingue la notion de capacité des cuves et de polyvalence des traitements qui y sont effectués :

— Les cuves sont dites *monobacs*, si elles ne peuvent recevoir qu'un seul porteur à la fois. Sinon elles sont *multibacs*, et peuvent accueillir plus d'un porteur en même temps. En effet, certains traitements peuvent être plus longs que d'autres. Si c'est le cas, les cuves ayant d'importants temps de traitement forment des goulots d'étranglement, qui peuvent ralentir la production et

réduire la productivité de la ligne. Ainsi, il est donc raisonnable d'accroître la capacité de réception de ces cuves pour lisser leurs charges. Ces cuves multibacs sont des cuves disposant de plusieurs bacs similaires, contenant tous la même solution pour offrir un même traitement. Les traitements peuvent ainsi se dérouler dans les bacs en parallèle. Dans un atelier de traitement de surface, les cuves multibacs restent tout de même moins nombreuses que les cuves monobacs. Cependant, le nombre de bacs peut être assez élevé dans une cuve multibac (une dizaine). La figure 1.10 illustre une ligne ayant une cuve multibac.

— Des cuves dites *monofonctions* si elles ne doivent assurer qu'un seul traitement dans la gamme de fabrication de chaque produit. Sinon, elles sont appelées cuves *multifonctions*, et elles peuvent assurer plus d'un seul traitement dans la gamme opératoire d'un produit. Un porteur peut alors visiter ce type de cuves plus d'une fois. Cette différence n'est pas liée à un changement de la solution chimique que contient ce type de cuves, mais à la durée du traitement. Le porteur peut passer par la même cuve deux fois et chaque fois y rester pendant une durée différente. Dans le cas général, ces cuves assurent des traitements de type rinçage qui ne demandent pas des durées opératoires importantes. Logiquement, deux traitements réalisés dans la même cuve multifonction ne doivent pas être successifs, sinon on ne parle plus de traitements différents. Notons que la présence de cuves multifonctions implique que le problème d'ordonnancement à résoudre peut-être qualifié de réentrant, selon le vocabulaire utilisé en ordonnancement de manière générale. Pour le HSP, on parlera de problème avec recirculation.

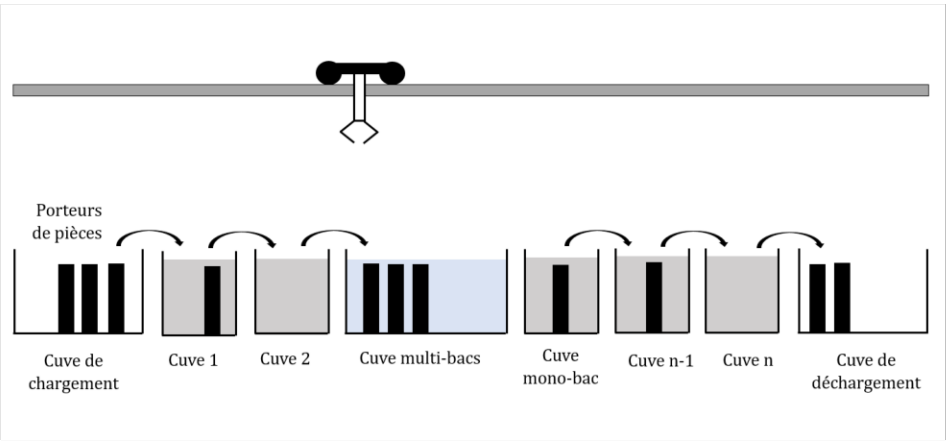

FIGURE 1.10 – Une ligne de traitement de surface avec une cuve multibac.

Le chargement et le déchargement des pièces sur des porteurs se font, en général, de façon manuelle. Pour cela, chaque ligne dispose d'un poste de chargement et d'un poste de déchargement respectivement en entrée et en sortie de ligne. Ils sont en général considérés comme des cuves fictives, de capacité infinie, et peuvent même

parfois être réellement composés de cuves, vides ou remplies de solutions neutres pour préserver les pièces de la détérioration. Les postes de chargement sont équivalents aux stocks d'entrée dans les lignes de production traditionnelles et les postes de déchargement sont assimilés aux stocks de produits finis de ces lignes. Ces deux postes peuvent exister sous deux configurations différentes :

— dissociées : dans ce cas, les postes sont indépendants et en général situés respectivement aux deux extrémités de la ligne.

— associées : un seul poste/cuve fictive, appelée station de chargement-déchargement. On rencontre ce cas dans les ateliers à espace limité ou s'il n'y a qu'un seul opérateur responsable des deux opérations de chargement et déchargement.

1.2.3 Les robots

Les robots, aussi appelés palans, sont les ressources responsables du transport des porteurs d'une cuve à une autre. Ils représentent le système de manutention de ce type de ligne. Ils se déplacent sur des rails installés au-dessus ou le long de la ligne des cuves. Chaque robot est lié à un rail bien précis, duquel il ne peut pas changer. Les palans sont amenés à effectuer deux types de déplacement lors des tâches de transport :

— Des déplacements en charge, quand le robot transporte un porteur d'une cuve à une autre. Lors de ce type de déplacement, le robot effectue quatre mouvements élémentaires :

  - Il se positionne au-dessus de la cuve contenant le porteur à déplacer, dès que celui-ci finit son traitement,
  - Il descend et récupère le porteur, puis s'élève et laisse égoutter le porteur au-dessus de la même cuve,
  - Il se déplace avec le porteur égoutté jusqu'à la cuve dans laquelle le traitement suivant sera effectué,
  - Il stabilise, descend, puis dépose le porteur dans cette cuve.

— Des déplacements à vide, quand le robot vient d'accomplir un déplacement en charge et se déplace vers une autre cuve pour entamer le déplacement en charge suivant. Ce mouvement part de la cuve d'arrivée d'un mouvement en charge, vers la cuve de départ du mouvement en charge suivant.

Les mouvements élémentaires assurant un déplacement en charge d'un robot, sont visualisés sur la figure 1.11 avec des flèches continues. Les déplacements à vide apparaissent avec des flèches en pointillés. En général, les déplacements en charge sont moins rapides que les déplacements à vide.

Les robots ont des caractéristiques connues à l'avance, comme les durées de déplacement à vide, les temps de montée, de descente, de déplacement en charge, ainsi que leurs vitesses maximales à vide et en charge. Ils ne doivent pas s'arrêter au cours de leurs déplacements, sauf dans des cas stricts et bien définis : lors de

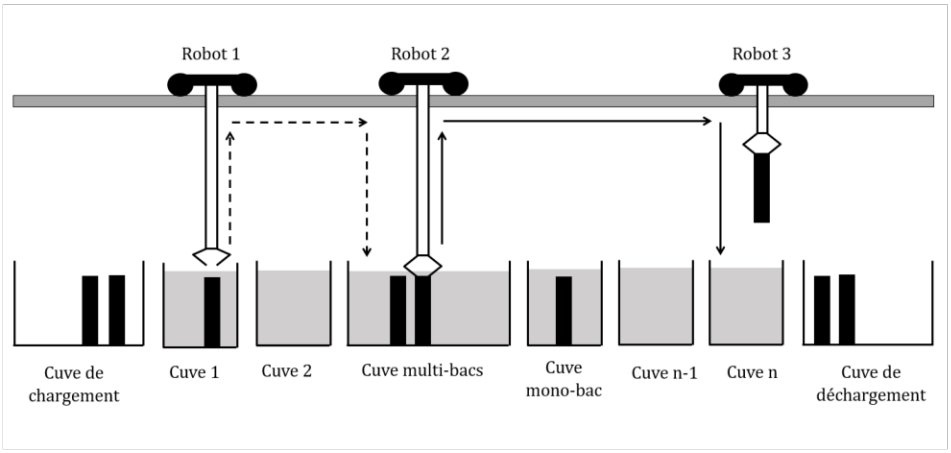

FIGURE 1.11 – Décomposition des mouvements effectués par un robot.

l'égouttage des pièces ou la stabilisation au-dessus des cuves, ou encore quand ils n'ont aucune tâche de transport à assurer. Tout autre arrêt des robots pourrait affecter la qualité des pièces (exemple d'oxydation) et induire des retards sur la production.

1.3 Le Hoist Scheduling Problem (HSP)

1.3.1 Le problème

Les ateliers de traitement de surface comportent trois catégories de ressources, dont deux responsables du bon déroulement de la ligne : les cuves, responsables des traitements à apporter aux pièces, et les robots, responsables des déplacements de ces pièces entre les cuves pour permettre la continuité de la production. Les porteurs ne sont employés que pour constituer des lots de pièces pouvant être traités simultanément. Ces deux ressources primordiales impliquent ainsi deux types d'opérations : la trempe réalisée dans les cuves et le transport réalisé par les robots. Ces deux types d'opérations sont étroitement liés et l'ordonnancement des unes conduit à l'ordonnancement des autres. Ceci est dû à la nature des contraintes sur ce type d'ateliers (temps opératoires limités, interdiction de stocks intermédiaires, etc.), mais aussi et simplement, car les opérations de trempe ne peuvent pas être abordées sans les transports effectués par les robots et vice versa (les robots ne sont là que pour un besoin de manutention, qui est impossible sans leur présence).

En général, les problèmes d'ordonnancement d'ateliers consistent à affecter aux ressources disponibles un ensemble de tâches (opérations) [Rodammer & White 1988]. Chaque opération doit être effectuée sur une ressource de l'atelier de production. Par ailleurs, l'ordonnancement consiste aussi à déterminer le séquencement de ces opérations, en respectant les différentes contraintes (comme celles liées aux durées d'exécution des opérations, à l'utilisation des ressources, à la disponibilité des tâches,

etc.). Dans les problèmes d'optimisation, le bon ordonnancement est celui qui optimise (minimise ou maximise) le critère de performance escompté, par exemple la minimisation de la durée, du coût, du retard, ou la maximisation de la productivité.

Ainsi, le problème d'ordonnancement des ateliers de traitement de surface consiste à affecter et séquencer les opérations de trempe aux ressources traitement (cuves), ou également, affecter et séquencer les opérations de transport aux robots. La majorité des travaux conduits sur ce problème a choisi d'ordonnancer les opérations de transport, ce qui implique automatiquement l'ordonnancement des tâches de trempe sur les cuves. Ce problème est connu dans la littérature sous l'appellation *Hoist Scheduling Problem* ou HSP [Lei & Wang 1989b, Serafini & Ukovich 1989, Crama & Van de Klundert 1997, Bloch *et al.* 1997, Bloch & Manier 1999, Manier & Bloch 2003, Bloch *et al.* 2008]. Il s'agit ici de fournir l'ordonnancement qui optimise un ou plusieurs critères, en respectant un ensemble de contraintes (fenêtres de temps sur les durées de traitement, disponibilité du(des) robot(s) et/ou des cuves, etc.) [Shapiro 1985a, Shapiro & Nuttle 1988, Manier *et al.* 2000].

1.3.2 Les contraintes

Les contraintes liées à l'ordonnancement des ateliers de traitement de surface (HSP) sont nombreuses [Rodošek & Wallace 1998]. Certaines concernent la nature des ressources employées, d'autres sont imposées soit par la gamme de fabrication, ou pour respecter des caractéristiques du processus chimique mis en œuvre, afin d'assurer son bon déroulement.

1.3.2.1 Les contraintes liées aux ressources

Ces contraintes concernent soit les ressources de traitement (les cuves) ou les ressources de transport (les robots) :

— Les contraintes relatives aux cuves :

- Contraintes dites de capacité et de disponibilité : chaque cuve a une capacité définie et ne peut donc pas recevoir en même temps qu'un nombre limité de porteurs. Ainsi, une cuve *monobac* ne peut recevoir qu'un seul porteur, alors qu'une cuve *multibac* peut en recevoir autant que le nombre de bacs dont elle dispose. Cela implique qu'il faut s'assurer de la disponibilité d'au moins un des emplacements de la cuve avant d'y destiner un porteur de pièces.

— Les contraintes relatives aux robots :

- Contraintes de capacité : chaque robot ne peut déplacer en même temps qu'un seul porteur de pièces.
- Contraintes de disponibilité : un robot affecté à une tâche de transport doit être disponible au plus tard à la fin de l'opération de trempe qui précède ce transfert. Alors, entre deux mouvements de transport consécutifs associés

à un même robot, celui-ci doit avoir le temps suffisant pour se déplacer à vide entre la cuve d'arrivée du premier transport et la cuve origine du second.

- Contraintes temporelles : pendant les transferts de porteurs entre deux cuves, le robot doit avoir suffisamment de temps pour charger le porteur de la première cuve, le laisser égoutter, se déplacer vers la deuxième cuve, se stabiliser au-dessus d'elle, puis y décharger le porteur.
- Contraintes de non-attente : une fois qu'une opération de transport est commencée, elle ne doit pas être interrompue, le robot associé ne doit donc pas s'arrêter avant la fin de celle-ci (après avoir déposé le porteur dans la cuve destinataire). Ainsi, toute pause pendant une opération de transfert est interdite afin d'éviter l'oxydation des pièces.
- Contraintes de non-collision : dans le cas des lignes multirobots, les ressources de transport se déplacent sur le même rail, il faut donc conjurer le risque de collision entre les robots.

1.3.2.2 Les contraintes liées à la gamme de production

La gamme de fabrication implique un ordre opérationnel à suivre, affecte les postes de charge (ressources) aux différentes opérations, et précise les durées opératoires. Ainsi, dans les ateliers de traitement de surface, les contraintes relatives à la gamme de production sont :

— Contraintes d'ordre opératoire : les porteurs de pièces doivent respecter l'ordre d'exécution des opérations de trempe, imposé par la gamme.

— Contraintes de fenêtres de temps (time window constraints) : les opérations de traitement dans les cuves ont des durées opératoires bornées, avec une limite temporelle inférieure et une limite supérieure. Ces durées bornées dépendent de la nature du traitement chimique effectué dans chacune des cuves. Elles sont imposées pour respecter des impératifs d'ordre chimique et pour garantir une bonne qualité des produits. Par exemple, quand le traitement est une attaque acide, une durée minimale est nécessaire pour assurer que le traitement soit réalisé correctement, néanmoins, les pièces ne doivent pas rester trop longtemps pour éviter tout risque de détérioration ou des défauts de qualité. Quant à un traitement de type dépôt de métaux précieux (argent, or, etc.), la durée minimale est requise pour obtenir un dépôt suffisant, alors que la durée maximale évitera un dépôt de matière supplémentaire et donc des coûts de matière plus importants.

Avec ces durées strictes, les marges sur les temps d'immersion sont de 0% pour les traitements de type attaque acide ou dépôt de métaux précieux. En revanche, pour des opérations de type rinçage, les durées d'immersion peuvent s'allonger en ne causant aucune dégradation des pièces. Dans ce cas, les marges sur les temps d'immersion peuvent être très grandes, voire des marges infinies.

1.3.2.3 Les contraintes liées au processus de production

— Contraintes de non-préemption : toutes les opérations, qu'elles soient de trempe ou de transport, ne doivent pas être interrompues une fois commencées. En effet, les produits chimiques utilisés dans les bains peuvent continuer à attaquer les pièces pendant ces interruptions.

— Contraintes de non-attente : les pièces ne peuvent pas attendre dans les cuves une fois leur traitement est terminé. Cette contrainte est en liaison étroite avec la limite de temps supérieure imposée par la contrainte de fenêtres de temps qui veut éviter toute exagération sur les opérations de trempe (excès de dépôt ou d'acide).

— Contraintes de non-stock : toujours dans l'objectif d'empêcher la dégradation des pièces, celles-ci ne doivent pas être stockées entre deux opérations de trempe. Ainsi, tout stock intermédiaire entre les cuves est interdit.

En tenant compte de toutes ces contraintes, une fois entré sur la ligne de traitement de surface, un porteur de pièces doit enchaîner alternativement les opérations de trempe et de transport, sans préemption et sans attente. Cette continuité entre les opérations implique que la date de fin d'une opération de trempe est égale à la date de début de l'opération de transport, et la date de fin de l'opération de transport est égale à la date de début de l'opération de trempe suivante. Cela implique aussi que l'ordonnancement des opérations de trempe permet de déduire l'ordonnancement des opérations de transport, et vice-versa, comme évoqué dans le paragraphe 1.3.1. Néanmoins, les robots représentent les ressources critiques de la ligne, et la plupart des travaux réalisés sur le HSP cherchent à ordonnancer les mouvements des robots.

1.3.3 La classification

Le problème d'ordonnancement des mouvements de robots dans les ateliers de traitement de surface (HSP) est apparu dans la littérature en 1976, avec les chercheurs Phillips et Unger [Phillips & Unger 1976], puis en 1988, avec Shapiro et Nuttle [Shapiro & Nuttle 1988]. Dès lors, le problème a continué de préoccuper les chercheurs, qui l'ont étudié sous différentes situations, en particulier dans ses aspects statique et dynamique :

— Lors d'une production statique, les pièces à produire sont connues à l'avance, sur une période de temps donnée. Il faut donc trouver à quel moment et dans quel ordre les faire entrer sur la ligne, et déterminer les durées effectives de trempe dans les cuves, l'agencement des mouvements du(des) robot(s), etc.

— Lors d'une production dynamique, les pièces à produire ne sont pas connues à l'avance et arrivent sur le système à des moments aléatoires. Il faut ainsi à chaque arrivage de nouvelles pièces, décider si l'on doit commencer à les traiter ou si on doit attendre avant de lancer leur traitement, en tenant compte de l'état du système et des autres arrivées éventuelles de pièces.

La variété des études sur le HSP a contribué à la proposition d'une classification pour ce problème [Bloch *et al.* 1997, Bloch & Manier 1999, Manier & Bloch 2003], sous quatre grandes classes (ou variantes) : Les HSP cyclique (CHSP), prédictif (PHSP), dynamique (DHSP) et réactif (RHSP).

1.3.3.1 Le HSP Cyclique

Ce type de problème est rencontré dans les productions en grande série, où la ligne produit généralement les mêmes types de pièces (pièces identiques), dans des quantités qui ne varient presque pas sur la période étudiée. Ce type de production est dit *monoproduit*. Cette production peut aussi s'appliquer dans le cas où les gammes de traitement de différents types de produits sont les mêmes et que leurs temps de trempe sont similaires. Ces différents types de produits peuvent ainsi être considérés comme un seul lors de l'ordonnancement.

Dans le HSP cyclique, on cherche une séquence de mouvements du(des) robot(s) à répéter indéfiniment. Chaque répétition est appelée cycle ou période. La durée d'un cycle est appelée temps de cycle. La séquence de mouvements donne un ordonnancement cyclique. Un cycle est dit n-périodique ou de degré n, si n pièces sont introduites sur la ligne au cours d'une période et également, n pièces la quittent [Lei & Wang 1994b, Levner *et al.* 1996, Kats *et al.* 1999, Che *et al.* 2003, El Amraoui *et al.* 2013b, Elmi & Topaloglu 2017]. Avec ce mode de fonctionnement, la première catégorie cyclique du problème d'ordonnancement des lignes de traitement de surface (1-périodique), a été introduite par [Phillips & Unger 1976] et connue sous le *Cyclic Hoist Scheduling Problem* ou *CHSP* [Lei & Wang 1989a, Che *et al.* 2015]. Son objectif est de déterminer la séquence des mouvements des robots de durée (période T) minimale pour un degré de cycle n fixé, en cherchant à maximiser la productivité de la ligne [Che *et al.* 2001]. Cette séquence doit être faisable, à savoir respecter les contraintes, en particulier les fenêtres de temps spécifiées pour les opérations.

Le HSP est un problème fortement contraint de manière générale, et sa variante cyclique, même la plus simple (monoproduit et monorobot), a été démontrée comme étant un problème NP-complet [Lei & Wang 1989b]. La majorité des scientifiques qui s'y sont intéressés a résolu la version du problème 1-cyclique, et dans le cas le plus simple d'une ligne de galvanoplastie, c'est-à-dire avec des cuves monobacs et/ou monofonctions, desservies par un seul robot [Baptiste *et al.* 1992, Song *et al.* 1993, Armstrong *et al.* 1994, Chen *et al.* 1994, Song *et al.* 1995, Levner *et al.* 1997, Lim 1997, Ng & Leung 1997, Levner & Kats 1998, Chen *et al.* 1998b, Mateo *et al.* 2002, Zhou & Li 2003, Mangione *et al.* 2003a], etc. En revanche, l'ordonnancement *multicyclique* a été relativement moins étudié que la version 1-périodique. Nous pouvons citer les travaux suivants pour la version multicycle du problème : [Lei & Wang 1994b, Kats *et al.* 1999, Che *et al.* 2003, Che *et al.* 2011, Li & Fung 2013, Li & Fung 2014, Li *et al.* 2015, Li & Fung 2017, Mao *et al.* 2018].

1.3.3.2 Le HSP Prédictif

Dans le cas des productions en grandes séries et multigammes, à deux ou trois gammes qui diffèrent très peu, il reste possible d'appliquer le mode de gestion cyclique. Sinon, d'autres modes de gestion existent tels que la production par campagne. Il s'agit d'un enchaînement de phases de production monogammes. Même si la solution la plus simple consiste à achever la production d'une campagne avant de passer à une autre, il demeure plus pertinent de déterminer un ordonnancement permettant de faire cohabiter les deux types de gammes au cours d'une même phase transitoire [Varnier 1996], avec un objectif de minimiser la durée de cette phase.

Quand la ligne traite des produits différents et en petites séries, les modes de gestion cycliques ou par campagne ne sont plus à envisager. Ainsi, si la production reste stable et prévisible sur un horizon relativement petit, un autre mode de gestion est à appliquer. Il s'agit de déterminer un ordonnancement pour les différents produits à traiter avant leur entrée en ligne [Caux *et al.* 1995].

Ces deux cas de figure correspondent à des problèmes statiques et non cycliques. Ils ont été appelés *Predictive Hoist Scheduling Problem* ou PHSP [Varnier & Baptiste 1995,Fleury 1995,Fleury *et al.* 1996,Caux & Pierreval 1997,Lacomme 1998,Bloch *et al.* 1999].

1.3.3.3 Le HSP Dynamique

Il existe aussi le cas de productions en petites quantités d'une grande variété de produits, qui doivent être traités en temps limité. C'est une situation inévitable pour la plupart des compagnies, surtout celles offrant de la sous-traitance. En effet, quand la taille des séries diminue et les délais sont réduits, les responsables de production doivent combiner différents produits dans une même ligne pour augmenter la flexibilité de leurs systèmes. Les clients de plus en plus exigeants réclament la diminution des délais de production, et la planification doit alors suivre le même rythme. En revanche, il n'est pas facile d'obtenir une planification fiable sur quelques heures. Face à cette situation, les responsables de fabrication doivent être capables d'agir très rapidement suite à une commande urgente. La planification doit alors gérer un compromis entre trois objectifs : la productivité, la qualité et la flexibilité.

Pour le HSP, quand la qualité prime, toutes les opérations de transport sont affectées aux robots au moment où un nouveau porteur entre dans la ligne, en tenant compte de l'état de la ligne, et en particulier des porteurs en cours de traitement. L'ordonnancement doit respecter toutes les contraintes et permettre l'entrée du nouveau porteur, le plus tôt possible. L'objectif à optimiser est généralement la minimisation du temps total de réalisation de toutes les opérations de trempe (*makespan*). Cette approche est dédiée aux productions à forte valeur ajoutée et offre l'avantage de pouvoir faire face aux perturbations externes, telles que la réception d'une commande urgente ou le retard d'une livraison. Le problème

d'ordonnancement correspondant est appelé *Dynamic Hoist Scheduling Problem* ou DHSP [Yih 1994, Ge & Yih 1995, Cheng & Smith 1995, Lamothe 1996, Lamothe *et al.* 1996, Rosse Bloch 1999, Spacek *et al.* 1999, Fargier & Lamothe 2001, Hindi & Fleszar 2004, Paul *et al.* 2007, Feng *et al.* 2015, El Amraoui & Elhafsi 2016, Ramin *et al.* 2020].

1.3.3.4 Le HSP Réactif

Si la productivité et la flexibilité sont les objectifs primordiaux de l'entreprise, alors après chaque opération de transport, on cherche à déterminer quel prochain porteur il faut déplacer, tout en tenant compte de l'état de la ligne. Aucun ordonnancement prévisionnel n'est mis en œuvre. Le choix du porteur à déplacer dépend de la nature du bain dans lequel il se trouve et du temps restant dans ce bain. Parfois, il est difficile de respecter les temps de traitement, surtout en termes de borne supérieure. Cette approche est destinée aux productions à faible valeur ajoutée. Le problème d'ordonnancement associé est le *Reactive Hoist Scheduling Problem* ou RHSP [Thesen & Lei 1990, Yih & Chiu 1993, Yih *et al.* 1993, Lamothe 1996, Mak *et al.* 2000, Jégou *et al.* 2006].

1.3.3.5 Notation pour le HSP

Une étude intéressante sur la classification des problèmes HSP est celle de Manier et Bloch [Manier & Bloch 2003]. Les auteurs ont également fourni une typologie des différentes classes du HSP et proposé une notation générique qui regroupe à la fois : la classe du problème (α), les paramètres liés au système physique (β), les paramètres logiques liés à l'environnement de production (δ) et les critères d'optimisation (γ). Chaque domaine se compose lui-même de plusieurs paramètres :

— α, pour la variante du HSP où α=XHSP : la variable X dans XHSP donne le type de HSP considéré. X $\in \{C, P, D, R\}$ pour respectivement le HSP cyclique, prédictif, dynamique et réactif ;

— β, pour les paramètres physiques du système où $\beta = \beta1$, $\beta2$, $\beta3$, $\beta4/\beta5$, $\beta6$, $\beta7$, $\beta8/\beta9$. Ce domaine inclut respectivement :

- $\beta1$= nl : le nombre de lignes ;
- $\beta2$= ntransfer : le nombre de moyens de transfert reliant les lignes,
- $\beta3 \in \{\phi, synchro\}$: indique le besoin de synchronisation entre les robots et les moyens de transfert. ϕ pour le cas de non-synchronisation ;
- $\beta4 = (\beta41, \beta42, \beta43)$: indique respectivement pour chaque ligne : le nombre de robots (mh), le nombre de cuves (mt) et la capacité maximale des cuves (ct). Pour des lignes monorobots et des cuves de capacité unitaire, alors $(\beta41, \beta42, \beta43) = (\phi, \text{mt}, \phi)$;
- $\beta5$= nc : le nombre de porteurs ($\beta5 = \phi$, par défaut, s'il existe une infinité de porteurs) ;

- $\beta 6 \in \{\phi, circ\}$: indique s'il existe une circulation (circ) ou non (ϕ) entre les porteurs. La notion de circulation [Shapiro & Nuttle 1988] signifie que les porteurs déchargés à la fin d'un cycle doivent être de nouveau chargés pour le cycle suivant ;
- $\beta 7 \in \{\phi, ret\}$: indique la manière dont les porteurs vides sont transportés de la cuve de déchargement vers la cuve de chargement ; ret désigne qu'il existe un autre système assurant ce transport, alors que (ϕ) indique que c'est soit un robot qui assure cette fonction ou les deux cuves de chargement-déchargement sont associées ;
- $\beta 8 \in \{\phi, empty\}$: est associée à la gestion des porteurs vides qui ne peuvent être évacués de la ligne. $\beta 8$= empty, si les porteurs vides ne peuvent pas être stockés hors ligne ou au poste de chargement-déchargement, sinon $\beta 8 = \phi$;
- $\beta 9 \in \{\phi, ass, diss\}$: décrit la configuration des postes de chargement et de déchargement (ce sont respectivement des configurations quelconques, associées ou dissociées) ;

— δ, pour les paramètres logiques de l'environnement de production, où $\delta = \delta 1/\delta 2$, $\delta 3$, $\delta 4$, $\delta 5$. Ils incluent :

- $\delta 1$= nparts : quantité de pièces (ou porteurs de pièces) qui sont à traiter dans la ligne. Par défaut, s'il existe une infinité de pièces ($\delta 1 = \phi$) ;
- $\delta 2$= nps : nombre de types de produits à traiter. Si tous les produits sont identiques alors $\delta 2 = \phi$,
- $\delta 3$= nop : nombre maximal d'opérations dans les gammes de production à mettre en œuvre ;
- $\delta 4 \in \{\phi, clean\}$: indique si les porteurs doivent être nettoyés ou pas après l'opération de déchargement. $\delta 4 = \phi$ quand l'opération de déchargement est la dernière opération réalisée ;
- $\delta 5 \in \{\phi, recrc\}$: pour la re-circulation [Pinedo 1996], qui concerne les cuves multifonctions. $\delta 5 = \phi$ si toutes les cuves sont monofonctions ;

— γ, pour les critères d'optimisation. Les objectifs considérés sont :

- la maximisation de la productivité, qui se traduit par la minimisation de la période du cycle pour un CHSP, ou par la minimisation du *makespan* pour les autres variantes ;
- la minimisation du nombre de robots ;
- la minimisation du nombre de pièces défectueuses, etc.

Ainsi, la notation générique complète est :

$$XHSP|nl, ntransfer, synchro, (mh, mt, ct)_{i=1nl}/nc, circ, ret, empty/$$
$$load - unload|nparts/nps, nop, clean, recrc|criteria$$

Dans cette thèse, nous nous intéressons à une nouvelle variante cyclique du HSP, qui correspond à résoudre conjointement un CHSP et le dimensionnement des robots de la ligne. En ajoutant la valeur CHDSP au champ α de la notation, désignant le *Cyclic Hoist Design and Scheduling Problem*, nous pouvons alors exprimer la variante que nous étudions au moyen de cette notation :

$$CHDSP|mh, mt, 1//ass|/nop|T_{min}, mh_{min}$$

Il s'agit du problème de conception et ordonnancement cyclique des lignes de traitement de surface, dans leur configuration *multirobot*, avec des cuves *monobacs* et *monofonctions*, dans le cas d'une production de masse, et où on cherche à minimiser à la fois le temps de cycle et le nombre de robots.

Comme nos travaux portent sur une variante cyclique du HSP, nous nous concentrons davantage, dans la suite de ce chapitre, sur les recherches appartenant à cette classe, sachant que plusieurs études, telles que [Manier & Baptiste 1994, Bloch *et al.* 1997, Bloch *et al.* 2008], ont fourni un état de l'art des travaux réalisés sur le HSP. Ils ont aussi présenté différentes méthodes de résolution mises en œuvre, avec leurs avantages et inconvénients.

1.4 Cyclic Hoist Scheduling Problem : CHSP

Le CHSP, variante cyclique du HSP, est la variante la plus étudiée dans la littérature avec différents paramètres physiques du système et spécifications de production, et résolue avec diverses méthodes et approches. Le CHSP consiste à affecter et ordonnancer les mouvements des robots, en déterminant la ou les séquences des mouvements du ou des robots, et les dates de début d'exécution de ces mouvements. L'objectif du CHSP est de minimiser la durée du cycle.

Le CHSP a été résolu pour la première fois en 1976, par Phillips et Unger [Phillips & Unger 1976], puis, environ une décennie plus tard, en 1988, par Shapiro et Nuttle [Shapiro & Nuttle 1988]. Par la suite, le CHSP a continué à recevoir beaucoup d'attention de la part des scientifiques qui ont proposé divers modèles et méthodes pour résoudre ce problème, tout en tenant compte de divers objectifs et contraintes. Certaines de ces études ont examiné le cas de lignes *monorobots*, alors que d'autres se sont intéressées aux systèmes *multirobots*, qui sont plus complexes que les premiers. Dans la suite, nous donnons un état de l'art utile sur le *CHSP monorobot*, et un plus approfondi sur le *CHSP multirobot*.

1.4.1 Le CHSP monorobot

L'ordonnancement cyclique des ateliers de traitement de surface à robot unique a continué de susciter l'intérêt des chercheurs depuis la première étude qui a été mené par Phillips et Unger en 1976. Plusieurs approches ont été proposées pour le résoudre, notamment dans le cas 1-cyclique, à savoir :

— La programmation linéaire ([Phillips & Unger 1976, Song *et al.* 1993, Ng & Leung 1997, Liu *et al.* 2002, Zhou & Li 2003, Feng *et al.* 2018]) ;
— Les méthodes de type séparation-évaluation "Branch and Bound" ([Shapiro & Nuttle 1988, Lei & Wang 1989a, Armstrong *et al.* 1994, Chen *et al.* 1998b, Che & Chu 2007, Che *et al.* 2010]) ;
— La programmation logique sous contraintes ([Baptiste *et al.* 1992, Baptiste *et al.* 1993, Manier 1994]) ;
— Les heuristiques ([Lei 1993, Levner *et al.* 1997, Che & Chu 2005]) ;
— Les métaheuristiques ([Lim 1997]).

Nous donnons plus de détails sur ces approches, dans ce qui suit.

*** La programmation linéaire**

Phillips et Unger [Phillips & Unger 1976] ont été les premiers à s'intéresser à la modélisation du CHSP. Leur étude a servi de référence pour les travaux postérieurs. Ils ont proposé un modèle de programmation linéaire en nombres entiers mixtes pour résoudre le problème d'une ligne à un seul robot et à cuves monobacs, ayant une cuve *fictive* qui représente le poste de chargement-déchargement. Un porteur entre dans la ligne après la fin de l'opération de son chargement qui s'effectue dans la cuve de chargement-déchargement. Il en sort au moment où le robot le dépose dans cette cuve pour être déchargé. La solution optimale correspond à un ordre donné d'exécution des opérations de transport du robot au cours d'un cycle, et pour chacune de ces opérations sa date de début. L'objectif de l'optimisation est de maximiser le rendement de la production (minimiser la durée du cycle T). Pour résoudre le problème, les chercheurs ont adopté une méthode (MPSX-MIP) qui utilise une approche arborescente de type séparation-évaluation. Ils l'ont appliquée sur un exemple numérique industriel de 13 cuves. Néanmoins, ils ont fixé a priori le nombre de porteurs dans la ligne pendant un cycle, ce qui les a empêchés de trouver la solution optimale pour cette instance. Dans la même étude, ces chercheurs ont fourni une extension du problème au cas des lignes à cuves multifonctions.

[Song *et al.* 1993] ont traité le cas d'une ligne de traitement de surface avec des ressources restreintes, et l'ont modélisé sous forme de Programme Linéaire Mixte (PLM). Ensuite, une heuristique a été appliquée pour la résolution de ce modèle. Elle permet de déterminer la date de début au plus tôt de traitement de chacune des tâches. Le calcul de cette date dépend des dates de début antérieures et de la satisfaction des contraintes de capacité du robot.

Contrairement aux études précédentes, NG et Leung [Ng & Leung 1997] ont considéré les temps de déplacement entre les cuves (mouvements en charge) comme des variables de décision, c'est-à-dire qu'ils ont supposé que des pauses sont autorisées pour les robots pendant le déplacement des pièces. Ils ont proposé un modèle mathématique de programmation en nombres entiers qui prend en compte le cas où

le robot peut attendre après la période d'égouttage du porteur pendant le mouvement inter-cuves, avant de partir en charge vers la cuve suivante pour y déposer le porteur pour son opération de trempe suivante. Les auteurs ont dérivé une condition nécessaire pour trouver la solution optimale. Sur la base de cette condition, un algorithme a été proposé pour trouver les temps de déplacement optimaux.

[Liu *et al.* 2002] ont développé un modèle de programmation linéaire à nombres entiers mixtes pour résoudre le CHSP, avec l'objectif de minimiser la durée du cycle. Ce modèle prend en compte la possibilité d'attente d'un porteur au-dessus d'une cuve, et donc, les auteurs, ont introduit une variable qui calcule la différence entre le temps réel et le temps minimum d'un déplacement de robot entre deux cuves de traitement successives. Selon eux, l'introduction de cette variable dans le programme linéaire a permis d'améliorer la durée du cycle T par rapport au modèle de [Phillips & Unger 1976]. Les auteurs ont aussi été parmi ceux qui ont examiné d'autres extensions des systèmes de galvanoplastie. Ils en ont proposé deux : une première extension qui considère les systèmes avec les cuves multibacs, et une deuxième qui prend en charge des systèmes avec des cuves à fonctions multiples.

Zhou et Li [Zhou & Li 2003] ont également traité le HSP cyclique monorobot et ont examiné les phases à *goulet d'étranglement* (*bottleneck stages*). Ils ont proposé une programmation linéaire en nombres entiers mixtes basée sur le modèle de [Phillips & Unger 1976] et une procédure pour estimer le temps de traitement dans une phase de goulet d'étranglement avec plusieurs cuves.

Récemment, Feng et al. [Feng *et al.* 2018] se sont attaqués au problème de l'ordonnancement cyclique de robots où des pièces de types multiples sont considérées avec des cuves à capacités et fonctions multiples. Le problème a également rassemblé les contraintes de fenêtres temporelles. Les auteurs ont développé un modèle de programmation linéaire en nombres entiers mixtes pour résoudre le problème.

*** Les méthodes Branch and Bound (B&B)**

En 1988, Shapiro et Nuttle [Shapiro & Nuttle 1988] ont décrit un modèle de programmation linéaire et un algorithme associé pour trouver un cycle optimal pour un système à robot unique. Ils l'ont appliqué pour résoudre le même exemple numérique proposé par [Phillips & Unger 1976]. Dans leur modèle, ils prennent en compte les temps réels de trempe dans les cuves. La seule exception est de considérer les systèmes avec postes de chargement-déchargement dissociés. Dans ce cas le porteur, après avoir été déchargé, doit retourner de la cuve de déchargement vers celle de chargement, pour être chargé de nouveau et transporter de nouvelles pièces. Pour la résolution, les auteurs ont proposé un algorithme basé sur la méthode de *séparation & évaluation* (*Branch & Bound* ou B&B). Ils ont aussi évoqué le cas de cuves dupliquées.

Un an plus tard, [Lei & Wang 1989a] ont aussi proposé une méthode utilisant

la procédure de *séparation & évaluation* (B&B), pour la résolution du CHSP. La méthode permet de chercher des cycles n-périodiques pour des lignes de traitement de surface simples. L'objectif pris en compte était comme avant, la minimisation de la durée du cycle T, en considérant un ensemble de contraintes, comme celles liées à l'ordre d'exécution des opérations de transport.

Armstrong et al. [Armstrong *et al.* 1994] ont aussi utilisé la méthode de *séparation & évaluation progressive* (B&B), en se basant sur le *Minimal Time Span* qui est une limite inférieure du temps de cycle d'un programme candidat de mouvements du robot.

[Chen *et al.* 1998b], quant à eux, ont développé un modèle et un algorithme qui permettaient de trouver un ordonnancement cyclique optimal des mouvements du robot. Ils ont proposé deux procédures de type B&B : la première procédure permet d'énumérer toutes les distributions possibles des produits traités (les cartes de circuits imprimées) dans un cycle, c'est à dire, si chacune des cuves contient un porteur de produits au début du cycle ou non. La deuxième procédure permet d'énumérer les séquences possibles des mouvements du robot pour chaque distribution de produits initialement proposée, et qui est potentiellement réalisable.

Che et Chu [Che & Chu 2007], ont également étudié le HSP cyclique à robot unique avec deux extensions. Ils ont considéré que ces systèmes avec les cuves multifonctions ou/et les postes à cuves multiples comme les « grandes lignes de galvanoplastie les plus proches de la réalité ». Cependant, pour résoudre le problème, ils ont étudié les propriétés analytiques de ces systèmes et ont proposé un algorithme de *séparation & évaluation* (B&B) basé sur ces analyses, pour éviter les solutions dominées ou irréalisables. Ils ont comparé leurs résultats à ceux de Liu et al. [Liu *et al.* 2002] et ont constaté que leur approche était plus performante que celle basée sur la programmation linéaire en nombres entiers mixtes.

Che et al. [Che *et al.* 2010] ont étudié le cas d'un atelier de galvanoplastie avec des pièces de plusieurs types et des temps de traitement fixes. Ils ont supposé que les pièces ne devaient pas attendre après avoir été traitées ou pendant leur déplacement. Pour résoudre le problème, ils ont utilisé une procédure B&B dynamique et l'ont testée sur des instances de test générées de manière aléatoire, où elle s'est avérée efficace.

*** La programmation Logique sous contraintes**

Baptiste et al [Baptiste *et al.* 1992] ont proposé un modèle de programmation logique par contraintes qui a été simplement mis en œuvre et en mesure de fournir la solution optimale comme dans [Shapiro & Nuttle 1988]. Baptiste et al. [Baptiste *et al.* 1993] ont employé le même modèle de base et ont comparé ses performances avec deux solveurs différents (Prolog III et CHIP). Ils ont souligné que leur modèle est extensible aux cycles n-périodiques et aux cas de systèmes à robots multiples.

Marie-Ange Manier [Manier 1994] a aussi proposé une approche basée sur la programmation logique par contraintes, pour la recherche d'une solution optimale et s'est basée dans sa modélisation du problème sur le modèle proposé par Shapiro et Nuttle [Shapiro & Nuttle 1988]. Elle a défini un ensemble de disjonctions concernant les contraintes liées à la disponibilité du robot. Elle a imposé des intervalles de temps associés à un déplacement en charge d'un porteur. Elle a aussi défini la condition qui, entre deux opérations de transport, oblige d'avoir le temps nécessaire au robot pour se déplacer à vide entre la cuve de dépôt du premier porteur (la cuve de départ du mouvement à vide) et la cuve de retrait du second porteur (la cuve d'arrivée du mouvement à vide). D'autres contraintes disjonctives relatives à ces intervalles de temps ont été étudiées, selon les relations de précédence possibles entre deux opérations de transport. L'approche de résolution proposée par [Manier 1994] consiste à déterminer des ordonnancements réalisables, en se basant sur une procédure arborescente de type séparation & évaluation. L'auteur a aussi fourni des extensions de son modèle initial a plusieurs cas de lignes : à cuves multibacs, à cuves multifonctions, des lignes multirobots, ou des lignes avec différentes configurations de gammes et de postes de chargement-déchargement. Elle a proposé également une extension au cas n-cyclique.

Très récemment, Wallace et al. [Wallace & Yorke-S. 2020] ont réexaminé la modélisation du CHSP et proposé une nouvelle formulation de la programmation par contraintes. Ils considèrent que leur nouveau modèle est nettement plus simple que ceux de la littérature, tout en gardant la capacité de gérer plusieurs variantes du CHSP avec un minimum de modifications. Ils ont résolu ce modèle avec différentes approches d'optimisation en utilisant des techniques simples et multiples : programmation en nombres entiers (Integer Programming, IP), génération de clauses paresseuses (Lazy Clause Generation, LCG) et un exemple hybride qui combine les deux approches. Pour soutenir cette flexibilité, le modèle a été implémenté dans un langage de modélisation générique, le MiniZinc [Nethercote *et al.* 2007]. Les auteurs ont comparé ses performances sur les problèmes standards et les nouveaux benchmarks aux résultats rapportés dans la littérature. Ils sont arrivés à montrer que, grâce à leur modèle, les solveurs IP et LCG actuels de pointe peuvent être utilisés de manière impromptue pour obtenir des résultats de calcul qui surpassent les modèles de la littérature.

*** Les heuristiques**

Lei [Lei 1993] a suggéré une procédure de recherche binaire pour trouver les valeurs entières optimales des dates de début des opérations dans un programme de transport cyclique avec un itinéraire défini et des fenêtres de temps dépendantes. Pour cela, l'auteur a défini un programme linéaire en nombres entiers, pour minimiser le temps de cycle T. La procédure de recherche binaire proposée vise à trouver une valeur pour T qui vérifie les contraintes du modèle. Elle consiste à déterminer une borne inférieure et une borne supérieure pour T, et de définir les différentes valeurs pour T afin de trouver la valeur minimale qui satisfait toutes les contraintes.

Levner et al. [Levner *et al.* 1997] ont proposé un algorithme exact pour résoudre le problème cyclique monorobot. Ce dernier fournit le résultat en un temps polynomial $O(n^3 \log n)$. Ils ont aussi développé un programme mathématique avec des contraintes linéaires et quadratiques. Ils ont considéré une relation entre les variables, de façon que l'ordre des opérations du robot dépende de la valeur de T. Cependant, ce programme ne peut pas être résolu en un temps polynomial. Ainsi, les auteurs l'ont modélisé de nouveau en éliminant les contraintes quadratiques et la dépendance entre quelques indices avec T. Ils ont utilisé une approche par intervalle pour la détermination de T en un temps polynomial. Une fois la durée du cycle T trouvée, les valeurs des dates de début des opérations de transport sont calculées, et l'ordonnancement des opérations du robot est défini. L'algorithme a été testé sur un exemple de ligne dans l'industrie électronique à 7 cuves.

Che et Chu [Che & Chu 2005] ont aussi développé un algorithme polynomial pour résoudre le cas simple du CHSP, ainsi que des extensions pour des lignes à cuves multibacs et multifonctions.

*** Les métaheuristiques**

Lim [Lim 1997] a proposé une approche basée sur un algorithme génétique pour résoudre le CHSP. Cet algorithme est capable de fournir un ordonnancement proche de l'optimum. Il part d'une population initiale composée d'un ensemble de solutions (chromosomes), et sélectionne les parents selon leurs forces (fitness). Le principe de la sélection se base sur la roulette, qui attribue à chaque individu, une probabilité proportionnelle à sa force. Le chercheur utilise l'algorithme proposé par Lei [Lei 1993], pour calculer la force de chaque individu. Il s'agit de la valeur de la durée de cycle de chaque solution. Les enfants sont obtenus avec un opérateur appelé LOX (Line Order Crossover). A chaque génération, de nouveaux enfants viennent remplacer les mauvais individus (solutions) se trouvant dans la génération actuelle, pour former, ainsi, la nouvelle génération de solutions. Lim [Lim 1997] a testé son algorithme sur l'exemple de Phillips et Unger [Phillips & Unger 1976]. Il a conclu que son algorithme converge rapidement vers la solution optimale, quand la taille de la population est de 30. Cependant, Lim [Lim 1997] ne donne pas de précisions sur la manière dont il gère les nombreux chromosomes correspondant à des solutions non faisables.

Plus récemment, Lei et al [Lei *et al.* 2017] ont développé un algorithme basé sur une autre classe de métaheuristiques, appelée *Quantum-Inspired Evolutionary Algorithm* (QEA), et chargée de faire évoluer une population initiale de solutions. L'algorithme intègre trois stratégies de décodage pour convertir des individus quantiques en séquences de mouvements de robots. En outre, des opérateurs de croisement et de mutation sont utilisés avec des probabilités adaptatives pour augmenter la diversité de la population. Une procédure de réparation est proposée pour traiter

les individus infaisables. L'efficacité de cet algorithme est montrée sur des instances générées aléatoirement.

La figure 1.12 fournit une représentation de la typologie du *CHSP monorobot*, qui contient les références citées, et d'autres non détaillées ici.

1.4.2 Le CHSP multirobot

Lorsque le nombre de cuves de traitement devient très grand, les distances à parcourir pour assurer la manutention et le transport des pièces deviennent plus importantes. Un seul robot semble alors insuffisant pour gérer toute cette charge dans un temps de cycle idéal. Il risque ainsi de représenter la ressource critique de la ligne et empêcher l'optimisation de la production. Par conséquent, pour améliorer la productivité de la ligne, accroître le nombre de robots devient une évidence pour assurer un transfert de pièces plus rapide et efficace [Karzanov & Livshits 1978, Kats 1982]. Le nombre de ces robots peut être déterminé d'une manière empirique [Leung & Levner 2006].

Dans une ligne unidirectionnelle, si le nombre de robots augmente, des contraintes additionnelles sont générées pour permettre d'assurer la bonne gestion des mouvements des robots [Manier 1994, Kharrat 2012] :

— Pour assurer une meilleure affectation des opérations de transport en charge aux robots,

— Pour gérer le risque de collision entre les robots qui circulent sur le même rail. Une collision peut se produire entre deux robots adjacents :

- si les deux robots sont en déplacement que ce soit pour un mouvement en charge ou un mouvement à vide,
- ou quand un robot est en déplacement et l'autre est en position d'arrêt.

La multiplication du nombre de robots, ainsi que le nombre de situations conflictuelles possibles rendent le problème plus difficile à résoudre. Comme dans le cas de CHSP monorobot, la version du problème à robots multiples a attisé la curiosité de plusieurs chercheurs. Karzanov et Livshits [Karzanov & Livshits 1978] semblent être les premiers auteurs à avoir étudié le CHSP à plusieurs robots. Cependant, ils ont étudié le système avec des voies de circulation parallèles pour les robots et non pas une seule voie que partagent les robots. Ils ont traité le problème avec des durées de traitement fixes. On peut alors considérer que ce problème est en réalité connexe au HSP, de type *Robotic Cell Scheduling Problem* (RCSP). Ils ont proposé un algorithme de complexité $O(N^3)$ pour trouver le nombre minimal de robots pour un temps de cycle donné, où N est le nombre de cuves dans une ligne de production. Par ailleurs, dans le cas d'un CHSP avec une seule voie de circulation partagée, Shapiro [Shapiro 1985b], dans sa thèse, a choisi une stratégie de partition appelée *zoning approach*, pour empêcher les collisions entre robots. Cette même

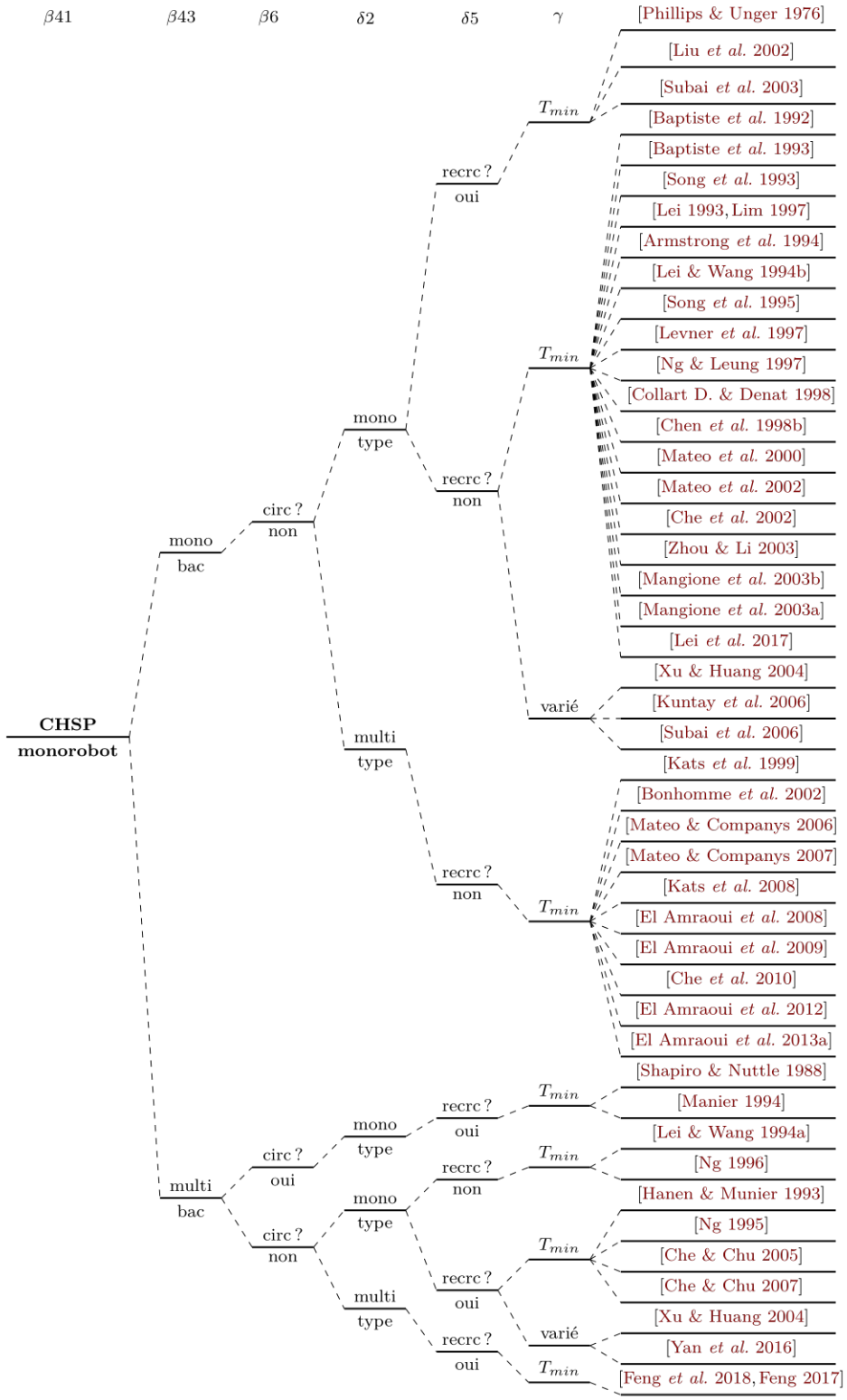

FIGURE 1.12 – Typologie des problèmes HSP cycliques à robot unique.

stratégie a été davantage développée subséquemment par [Lei & Wang 1991]. Elle se base sur la division de l'ensemble des cuves en sous-ensembles adjacents de cuves, dont le nombre est égal au nombre de robots employés sur la ligne. A chacun de ces sous-ensembles est affecté un robot. Les cuves frontières de ces sous-ensembles sont desservies par deux robots : un qui vient déposer un porteur de pièces et un autre qui vient le retirer. Ces dernières sont considérées comme des points de transfert [Manier 1994], et appelées cuves pivot [Lei & Wang 1991]. Ce type de résolution du problème écarte tout risque de chevauchement entre les mouvements des robots. Cependant, dans sa thèse, [Manier 1994] a étudié un scénario plus réel où les robots peuvent partager des zones communes sur la ligne, autres que les cuves pivot. Dans ce cas, le robot n'est plus affecté à un sous-ensemble de cuves, mais plutôt à un ensemble d'opérations de transport. Cette affectation est unique car chaque opération n'est affectée qu'à un seul robot. Ce type de résolution du problème considère ainsi les zones de chevauchements qui peuvent se produire entre les robots.

Pour les deux cas de figures, plusieurs méthodes ont été développées pour la résolution du *CHSP multirobot*, voire pour d'autres problèmes connexes multirobots (tels que le RCSP), à savoir :

- Des heuristiques ([Lei & Wang 1991, Liu & Jiang 2005, Zhou & Liu 2008, Chtourou *et al.* 2013]) ;
- La programmation linéaire ([Leung & Zhang 2003, Leung *et al.* 2004, Che & Chu 2008, Zhou & Li 2009, Che *et al.* 2013, Li *et al.* 2015, Mao *et al.* 2018]) ;
- Des méthodes de type séparation-évaluation "Branch and Bound" ([Hanen & Munier 1994, Che & Chu 2004, Jiang & Liu 2014]) ;
- La programmation logique sous contraintes ([Varnier *et al.* 1997, Manier *et al.* 2000]) ;
- Des métaheuristiques ([Yang *et al.* 2001, Manier & Lamrous 2008] ;

Nous les présenterons plus en détails dans la suite.

*** Les heuristiques**

Comme dit précédemment, des travaux sont disponibles pour le RCSP qui est un problème connexe au HSP, et dont les chercheurs travaillant sur le HSP ont pu s'inspirer par ailleurs. Par exemple, Liu et Jiang [Liu & Jiang 2005] ont considéré un problème cyclique à deux robots et avec des temps de traitement fixes et ont proposé un algorithme polynomial pour sa résolution. Ce dernier permet la résolution du problème en un temps de calcul limité à $O(n^4 \log n)$. Il fournit à chaque robot un ordonnancement en considérant la zone de chevauchement. Les auteurs ont défini des intervalles de temps pour limiter le domaine de recherche des solutions faisables. Ces intervalles ont été déterminés en se basant sur les travaux de Levner et al. [Levner *et al.* 1997], à partir desquels, les auteurs définissent différentes valeurs du temps

de cycle T. Ensuite, pour chacune de ces valeurs de T, les dates de début des mouvements en charge sont déterminées. Ensuite, une procédure d'affectation est employée pour affecter les mouvements à l'un des deux robots. Enfin, si la solution est faisable, alors la solution optimale est trouvée.

Dans le cas du CHSP, Lei et Wang [Lei & Wang 1991] ont proposé un algorithme heuristique qui trouve des séquences de mouvements pour les systèmes à deux robots en utilisant une approche de partitionnement. Cet algorithme a été appelé l'algorithme du plus petit cycle commun (*The Minimum Commun Cycle*), de complexité $O(n(n-1)!)$. Il s'agit de partitionner le système en deux ensembles de postes de travail contigus et y affecter chacun des deux robots. La dernière cuve du premier groupement représente la cuve de déchargement de cette première zone et en même temps, la cuve de chargement pour la deuxième zone. Le système est ainsi divisé en deux sous-systèmes adjacents, chacun à un robot. Chaque partition représente un problème monorobot qui est résolu indépendamment de l'autre par un algorithme d'optimisation. Deux temps de cycle sont calculés, chacun pour une partition. Les deux durées de cycle doivent être égales, sinon une seconde résolution est relancée pour la zone avec la plus petite durée de cycle, et en parallèle, la durée la plus grande sera considérée comme une borne inférieure pour la nouvelle résolution. L'algorithme s'arrête quand les ordonnancements des deux zones ont un temps de cycle commun. A la fin, il reste à synchroniser les mouvements des deux robots au niveau de la cuve commune des deux zones, pour que la durée de trempe dans celle-ci soit respectée. Ces auteurs ont été les premiers à étudier un système à deux robots et ont également résolu l'exemple industriel de [Phillips & Unger 1976] pour deux robots. Ils ont mis en évidence l'avantage de leur approche, à savoir que l'affectation des robots a été simplifiée par l'approche de partitionnement (pas besoin de contraintes d'évitement des collisions ; supposer que les zones séparées sont non chevauchées). Néanmoins, ils ont également affirmé qu'avec l'approche de partitionnement, ils ne pouvaient pas examiner toutes les solutions possibles.

Zhou et Liu [Zhou & Liu 2008] ont aussi étudié le CHSP avec des zones de robots qui se chevauchent, dans une ligne de galvanisation à deux robots. En se référant à l'algorithme présenté dans [Liu & Jiang 2005], ils ont proposé un algorithme heuristique qui génère des séquences de transfert et les affectent aux robots. Les séquences générées sont monorobots et ne sont pas nécessairement réalisables. Pour l'affectation, une procédure est employée, qui attribue chaque opération à l'un des deux robots. Ensuite, un modèle de programmation linéaire (PLM) a été développé avec les contraintes de non-collision pour chercher la séquence optimale pour chaque affectation d'opérations aux robots. Ce modèle teste la faisabilité des deux ordonnancements monorobots générés. Les contraintes de non-collision ont été définies sur tous les cas de figure (vingt différents cas illustrés), et selon les positions des cuves pour les deux mouvements de transport à effectuer.

Chtourou et al. [Chtourou *et al.* 2013] ont traité le même problème que celui

abordé dans [Zhou & Liu 2008], des lignes à deux robots où les zones des robots se chevauchent. Toutefois, les auteurs ne fixent pas a priori une affectation donnée des opérations aux robots. Pour résoudre ce problème, ils ont proposé une approche hybride basée sur deux heuristiques et un programme linéaire mixte. Tout d'abord, ils commencent par générer différentes séquences de mouvements, en appliquant la méthode utilisée par [Zhou & Liu 2008] qui a les mêmes bases que celles employées dans [Kats & Levner 1997b, Kats & Mikhayletskii 1980]. Puis, ils proposent un algorithme pour affecter les mouvements à chaque robot. Ensuite, ils appliquent un modèle de programmation linéaire en nombres entiers mixtes (MILP) pour trouver les dates de début des mouvements en charge et le temps de cycle résultant. Enfin, la meilleure solution (avec le temps de cycle minimal) est choisie parmi les différentes séquences générées.

*** La programmation linéaire**

Leung et Zhang [Leung & Zhang 2003] ont développé un programme linéaire mixte (PLM) pour résoudre le CHSP avec plusieurs robots qui partagent le même rail et avec zone de chevauchement. Pour traiter l'évitement de collisions entre les robots, les auteurs ont défini un ensemble de contraintes en se basant sur les travaux de [Heinrichs & Moll 1997].

Un an plus tard, Leung et al. [Leung *et al.* 2004] ont proposé un nouveau programme linéaire mixte pour traiter le CHSP multirobot, avec zone de chevauchement. Ils ont considéré le cas où la gamme opératoire suit l'implantation des cuves. Ils ont appliqué une relaxation continue pour ce PLM, tout en ajoutant les inégalités triangulaires proposées par [Grötschel *et al.* 1985]. Les chercheurs ont montré que la relaxation du programme linéaire ainsi que l'ajout des inégalités valides permettent de limiter le nombre des nœuds branchés lors de la résolution du programme et de baisser les temps de résolution (CPU).

Che et Chu [Che & Chu 2008] ont investi le problème d'ordonnancement cyclique de plusieurs robots dans des ateliers de traitement de surface, mais à nouveau avec des temps de traitement constants. Ils ont développé un modèle mathématique et un algorithme polynomial pour le résoudre. Ils ont affirmé que l'algorithme proposé peut servir d'heuristique pour résoudre le même problème mais avec des fenêtres de temps dépendantes.

Zhou et Li [Zhou & Li 2009] ont étudié le CHSP à robots multiples avec des fenêtres temporelles. Ils ont suggéré une méthode qui divise le système en plusieurs zones ne se chevauchant pas et qui attribue un robot à chaque zone, comme l'approche proposée par Lei et Wang [Lei & Wang 1991]. Ils ont développé un modèle de programmation linéaire à nombres entiers mixtes pour traiter la dimension d'ordonnancement. Ils ont également étendu leur modèle pour résoudre les cas de goulots d'étranglement nécessitant des postes à plusieurs cuves identiques.

Che et al. [Che *et al.* 2013] ont soumis une nouvelle approche basée sur la programmation linéaire en nombres entiers mixtes. Ils supposent que les mouvements en charge des robots peuvent commencer et finir dans des cycles différents, contrairement à l'idée assumée implicitement ou explicitement dans la littérature, que les mouvements en charge commencent et finissent durant le même cycle.

D'autres recherches comme [Li *et al.* 2015, Mao *et al.* 2018, Li *et al.* 2019] ont étudié le CHSP multirobot et ont considéré des cycles de degré k : c'est le cas d'un cycle où k pièces identiques joignent la chaîne de production et k pièces la quittent. Ces études ont développé des modèles linéaires en nombres mixtes pour résoudre le problème. [Li *et al.* 2019] ont considéré le problème avec des fenêtres temporelles, des pièces à traiter identiques, et des cuves ré-entrantes (multifonctions).

*** Les méthodes *Branch and Bound* (B&B)**

Hanen et Munier [Hanen & Munier 1994] ont étudié le HSP cyclique à robots multiples, et ont supposé que les mouvements de transport sont arbitrairement affectés aux robots. Elles ont proposé un modèle de programmation en nombres entiers mixtes et une procédure par séparation et évaluation pour résoudre le problème.

Che et Chu [Che & Chu 2004] ont suggéré une approche différente pour traiter les contraintes d'évitement de collision, autre que l'approche de partionnement. Ils ont formulé ces contraintes sous forme d'inégalités disjonctives et ont étudié deux propriétés supplémentaires qui contrôlent les collisions. Ces dernières doivent être valables pour toutes les solutions possibles. Pour résoudre le problème, ils ont présenté un algorithme de séparation et évaluation (B&B) où les deux propriétés mentionnées sont relâchées. Cet algorithme est basé sur deux approches de type séparation-évaluation, nommées procédure A et procédure B. Pour la résolution de ces deux programmes, les auteurs ont appliqué l'algorithme polynomial proposé par [Chen *et al.* 1998b].

Jiang et Liu [Jiang & Liu 2014] ont étudié le CHSP multirobot avec des temps de trempe bornés et des sens de traitement multidirectionnels [Sun *et al.* 1994]. Ils ont analysé les positions relatives possibles de toute paire de mouvements et dérivé un ensemble de contraintes linéaires pour exprimer les impératifs de non-collision entre les robots. Sur la base de ces analyses, ils ont développé un nouveau modèle linéaire en nombres entiers mixtes, et ils ont proposé une stratégie efficace de séparation et d'évaluation pour résoudre le problème plus rapidement.

*** La programmation Logique sous contraintes**

Sur la base des travaux de [Manier 1994], Varnier et al. [Varnier *et al.* 1997] ont traité le cas de robots multiples dans des lignes qui peuvent contenir des cuves

dupliquées. Comme ils considéraient des séquences de traitement multidirectionnelles, le cloisonnement de la ligne en zones non chevauchées n'était plus possible. Ils ont donc proposé une approche de fractionnement qui attribue les opérations de transfert aux robots, au lieu de diviser la ligne en zones non chevauchées qui seraient attribuées à chaque robot. Pour traiter la contrainte de non-collision, ils ont imposé une zone de sécurité pour chaque robot qui doit se trouver dans sa zone de circulation. Le problème d'affectation a été résolu grâce à une règle heuristique, tandis que pour le problème d'ordonnancement, les auteurs ont présenté un modèle basé sur la programmation logique par contraintes. Ce modèle calcule le temps cyclique optimal pour un nombre donné de robots et une affectation donnée des opérations.

Manier et al. [Manier *et al.* 2000] ont également considéré le CHSP avec des robots multiples où la ligne de production est plus proche des environnements industriels réels avec des cuves dupliquées et multifonctionnelles. Les auteurs ont proposé une méthode de partitionnement statique pour affecter les opérations de transport aux robots. Les zones affectées à chaque robot peuvent se chevaucher et les cuves appartenant à une zone de robot peuvent ne pas être contigues. L'objectif de ce travail était de proposer une procédure de résolution pour une partition donnée. Les auteurs ont identifié un nombre de disjonctions correspondantes à toutes les collisions potentielles entre deux opérations de transfert effectuées par deux robots adjacents. Ils ont aussi défini des règles qui permettent d'arbitrer ces disjonctions, sur toutes les situations possibles. Pour la résolution, [Manier *et al.* 2000] ont proposé un modèle basé sur la programmation logique sous contraintes et l'ont résolu avec Prolog IV.

*** Les métaheuristiques**

Yang et al. [Yang *et al.* 2001] ont étudié le CHSP multirobot sans zone de chevauchement et ont proposé une méthode de résolution basée sur le recuit simulé. Dans le cas de système considéré, les cuves de chargement et déchargement sont dissociées. Les auteurs ont suggéré une décomposition arbitraire de la ligne en m partitions. Ensuite, ils ont développé un programme linéaire à appliquer sur chaque partition, qui comporte des contraintes sur les temps de traitement, les temps de transfert du robot et l'affectation des opérations au robot. Ils ont appliqué l'algorithme de [Armstrong *et al.* 1994] pour la résolution des sous-problèmes sur les partitions monorobots. Ensuite, les auteurs ont proposé un deuxième algorithme basé sur le recuit simulé pour résoudre le problème multirobot.

Manier et Lamrous [Manier & Lamrous 2008] ont proposé un algorithme évolutionnaire pour résoudre le CHSP multirobot avec zones de chevauchement. Les cuves de chargement et déchargement considérées sont associées. La nouveauté de ce travail réside sur le fait d'utiliser les mouvements à vide dans le codage des solutions, contrairement à l'approche de codage habituellement utilisée dans la littérature, qui, elle se base sur les mouvements en charge des robots. L'algorithme est bâti

comme suit. Un ensemble de chromosomes de plusieurs tailles viennent composer la population initiale, qui est générée d'une manière aléatoire. Ensuite, un programme linéaire mixte est appliqué pour évaluer chaque chromosome (calculer sa *fitness*). Puis, un sous-ensemble de solutions est sélectionné suivant une méthode élitiste et une sélection universelle stochastique. Pour obtenir la descendance, un croisement en 2-points et une mutation qui consiste à échanger deux gènes sélectionnés aléatoirement, sont réalisés. Par la suite, le chromosome fils doit remplacer le chromosome parent dans la prochaine population, si bien évidemment, la solution correspondante est réalisable. Dans le cas inverse, deux procédures de réparation sont employées. Toutefois, l'algorithme proposé ici ne gère pas les situations conflictuelles entre les robots partageant le même rail.

Nous déployons les références citées dans la figure 1.13 associée au *CHSP multi-robot*, ainsi que d'autres références utiles.

Le CHSP a été traité, parfois, en envisageant d'autres objectifs d'optimisation que la minimisation du temps de cycle. Avec un objectif environnemental, Xu et Huang [Xu & Huang 2004] ont associé le CHSP à la minimisation des déchets et ont proposé un procédé de galvanoplastie respectueux de l'environnement. Quant à Liu et al. [Liu *et al.* 2012], ils ont développé un modèle à triple objectifs, qui optimise simultanément la productivité, l'économie d'énergie et la minimisation de l'eau douce. Par ailleurs, d'autres travaux se sont intéressés au CHSP tout en considérant l'optimisation de la conception de la ligne. Nous approfondissons l'étude de cet objectif, dans le paragraphe suivant.

1.4.3 Le CHSP et la conception

L'objectif de conception (*Design en anglais*), associé ou pas à celui de l'ordonnancement, a été relativement peu abordé, et chaque fois avec un nouveau scénario. Certains auteurs le liaient à l'implantation de la chaîne de production, comme dans l'étude de Zhao et al. [Zhao *et al.* 2013], où l'objectif était d'optimiser l'allocation spatiale des ressources de traitement (les cuves de trempe). Qu et al. [Qu *et al.* 2017], cependant, l'ont considéré dans le cas d'un système en 2 dimensions (2-D). Ils ont supposé que la ligne de production n'est pas nécessairement une structure à une seule dimension (1-D) mais plutôt, une ligne de production compacte en 2-D. Ils ont alors étudié le problème joint de conception et de fonctionnement du CHSP mais en 2-D.

Dans cette thèse, nous nous intéressons également au problème combiné de conception et d'ordonnancement des ateliers de traitement de surface. Toutefois, l'aspect de conception que nous considérons ne concerne pas la répartition spatiale des ressources de traitement mais plutôt le dimensionnement des moyens de manutention, c'est à dire la quantification du nombre de robots. Dans cette optique, des travaux antérieurs ont été réalisés, que nous citons ci-dessous.

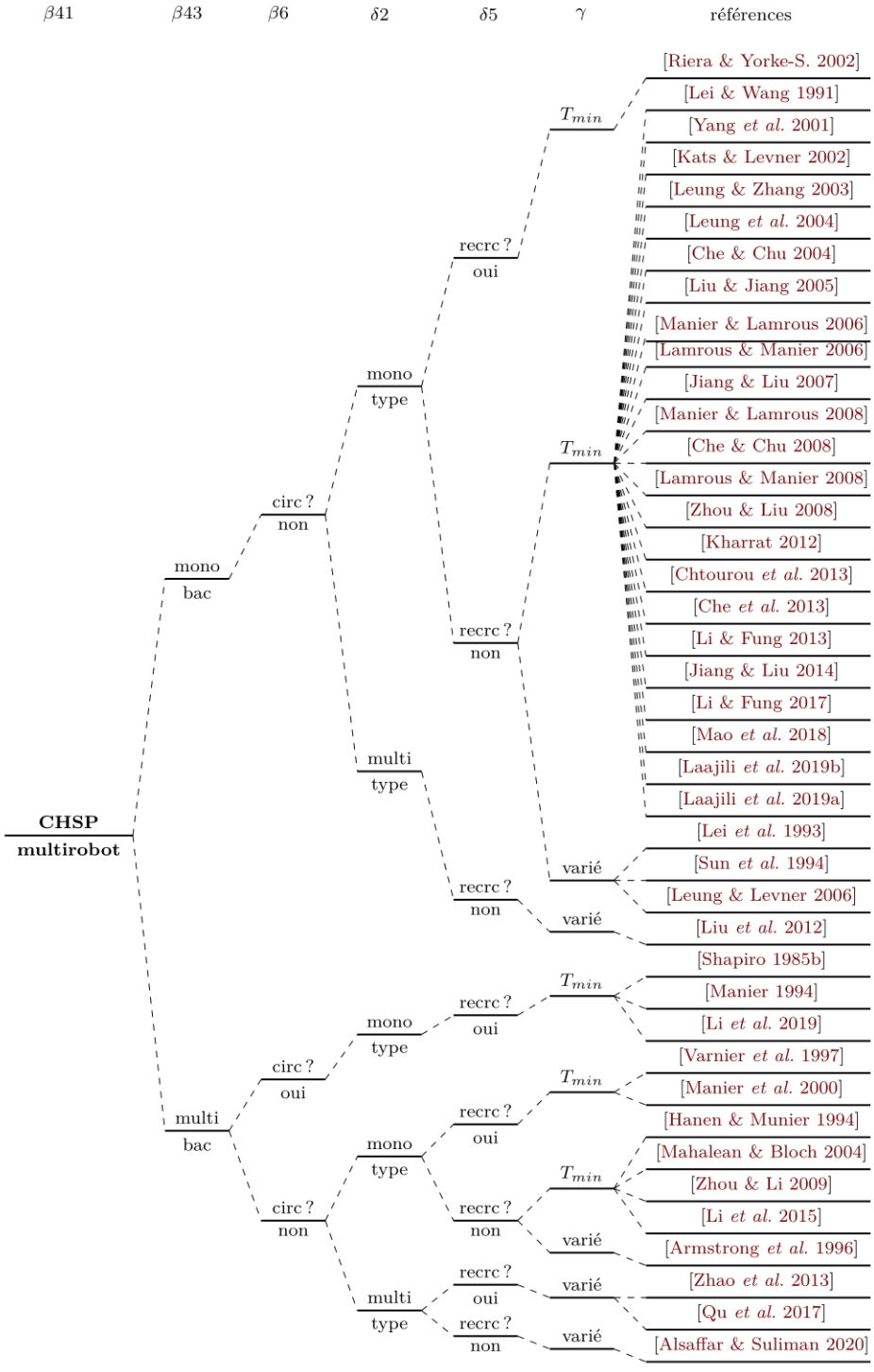

FIGURE 1.13 – Typologie des problèmes HSP cycliques à robots multiples.

Karzanov et Livshits [Karzanov & Livshits 1978] ont considéré les ateliers de galvanoplastie avec des voies de circulation parallèles pour les robots et des durées de traitement constantes dans les cuves. Ils ont proposé un algorithme de complexité $O(N^3)$ pour trouver le nombre minimal de robots pour un temps de cycle donné, où N est le nombre de cuves dans la ligne. Kats et Levner [Kats & Levner 1997a] ont constaté que le problème de la minimisation du nombre de robots pour toutes les durées de cycle possibles peut être résolu dans une complexité temporelle $O(N^5)$. Quant à Leung et Levner [Leung & Levner 2006], ils ont proposé un algorithme qui trouve le nombre minimal de robots pour tous les temps de cycle possibles et détermine ensuite la séquence cyclique la plus courte pour les robots, également avec des temps de traitement fixes. Rappelons ici pour ces trois articles, que le fait de considérer des durées fixes et non bornées rend le problème différent voire plus simple, et ne correspond pas tout à fait à la définition des caractéristiques du HSP donnée ci-dessus et dans les articles d'état de l'art.

Lei et al. [Lei *et al.* 1993] ont cherché à trouver une séquence cyclique pour un CHSP tout en minimisant le nombre de robots nécessaires dans la ligne de production. Les auteurs ont proposé un algorithme glouton qui cherche la partition de la ligne en groupes de cuves contiguës. Chaque groupe est affecté à un robot et cette partition géographique élimine les risques de collision. La construction itérative des groupes cherche à maximiser la taille de chaque groupe, donc de minimiser le nombre de robots, de sorte de respecter les contraintes du système. Pour une affectation donnée l'ordonnancement optimal est obtenu par un modèle exact, néanmoins cet algorithme ne garantit pas que la partition obtenue sera optimale pour le nombre de robots obtenu, en termes de temps de cycle.

Les mêmes auteurs ont amélioré le temps de résolution de leur heuristique gloutonne dans [Armstrong *et al.* 1996]. Comme dans leur premier article, un groupe de cuves donné est construit de manière progressive en insérant à chaque étape de construction une cuve contiguë supplémentaire et en testant la faisabilité de la séquence associée. Dans cette phase d'évaluation d'une séquence, ils ont remplacé la résolution d'un programme linéaire précédemment utilisé par la recherche d'un plus long chemin dans un graphe. Pour un groupe donné, le graphe est obtenu en insérant dans le graphe précédent : un noeud associé à la nouvelle cuve insérée et des arcs correspondant aux contraintes supplémentaires à prendre en compte. Ils ont évalué les performances de leur méthode sur des instances générées de manière aléatoire et des instances de référence.

Manier et Lamrous [Manier & Lamrous 2006, Manier & Lamrous 2008] ont traité le CHSP multirobot également avec les deux objectifs de conception et d'ordonnancement. L'aspect conception a été abordé grâce à une procédure de construction et d'affectation simultanées des séquences de mouvements aux robots. Cette procédure finit ainsi par affecter un nombre de robots à la ligne, aléatoirement. Ensuite, pour

l'aspect ordonnancement, les auteurs ont développé un modèle de programmation linéaire en nombres mixtes qui évalue la faisabilité des séquences de mouvements en tenant compte des contraintes du problème, puis calcule le temps de cycle résultant. Le problème joint a été résolu grâce à un algorithme génétique hybride qui fait appel à la procédure de construction et au modèle linéaire. Par ailleurs, pour la première fois dans ce type de travaux, les auteurs ont utilisé les mouvements à vide des robots dans la procédure de construction (dans le codage et le décodage des solutions) et non pas les mouvements en charge, comme cela a été toujours le cas dans la littérature. Cette représentation est très intéressante car avec le codage des solutions à base des mouvements à vide, les auteurs utilisent un unique codage pour représenter des solutions à un ou plusieurs robots, ce qui leur permet de traiter dans une même résolution le problème avec différents nombres de robots possibles. A l'issue de leur approche hybride, ils obtiennent un temps de cycle minimal pour chaque nombre de robots possible, et donc un binôme de résultats : le nombre de robots considéré dans la ligne et le temps de cycle minimal correspondant.

1.4.4 Synthèse

Le problème de l'ordonnancement des mouvements des robots dans les ateliers de traitement de surface (le HSP) a été beaucoup étudié dans la littérature scientifique depuis 1976. Le cas cyclique, le CHSP, identifié parmi quatre variantes classiques du HSP, a le plus attiré l'attention des chercheurs, que ce soit dans le cas d'un seul ou de plusieurs robots, avec différents cas réels de systèmes de production, tels que les cuves multifonctions, les cuves à capacité multiple ou les systèmes de traitement de pièces multitypes. Le CHSP a majoritairement été traité en considérant l'objectif de minimisation du temps de cycle, pour maximiser le rendement et améliorer la productivité du système. L'état de l'art réalisé sur les problèmes de CHSP nous a permis de :

- Distinguer les variantes *monorobot* et *multirobot* du CHSP,
- Mettre en évidence les différents objectifs d'optimisation considérés, ceux liés à l'ordonnancement des mouvements des ressources de transport tels que la maximisation de la productivité (la minimisation du temps de cycle), et ceux liés à la conception de la ligne tels que la minimisation du nombre de robots,
- Nous rendre compte de la variété des approches investiguées pour résoudre le problème et leur utilité, telles que les méthodes de type séparation-évaluation (*Branch and Bound*), la programmation linéaire, la programmation logique par contraintes, les heuristiques et les métaheuristiques,
- Et surtout de situer nos travaux par rapport à ceux de la littérature, comme le montre le tableau 1.2, organisé par type de CHSP (*mono* ou *multirobot*), type de ligne (simple ou complexe), objectifs à optimiser (relatifs à l'ordonnancement, à la conception ou aux deux), et méthode de codage (à base de mouvements en charge ou à vide).

Le travail présenté dans cette thèse s'intéresse à l'ordonnancement cyclique des mouvements des robots (CHSP) dans les ateliers de traitement de surface *multirobots*, *monoproduits*, avec des cuves *monofonctions* et *monobacs* (sans ré-entrance et de capacité unitaire). Nous intégrons aussi l'aspect conception de la ligne de la même manière que celle abordée dans les travaux de [Manier & Lamrous 2006, Manier & Lamrous 2008]. Nos recherches portent donc sur le problème conjoint de conception et d'ordonnancement cyclique des robots, que nous notons *CHDSP : the Cyclic Hoist Design and Scheduling Problem*. De ce fait, et contrairement à la plupart des études de la littérature, le nombre de robots devient une variable de décision au même titre que le temps de cycle, également à minimiser. Notre point de départ sera la méthode de codage proposée dans [Manier & Lamrous 2008], qui utilise les mouvements à vide des robots dans le codage des solutions, et non pas les mouvements en charge des robots comme dans la majorité des recherches existantes. Au cours de nos travaux, nous serons en particulier amenés à considérer et gérer les différentes situations conflictuelles entre les robots qui se déplacent sur le même rail.

Si notre travail veut s'inspirer des travaux de Manier et Lamrous [Manier & Lamrous 2006, Manier & Lamrous 2008] en matière de codage de solutions, toutefois, nous avons d'ores et déjà relevé deux inconvénients en examinant de près ces contributions :

— Le codage basé sur les mouvements à vide est certes un nouveau concept pour construire des séquences de mouvements affectées chaque fois à un nombre différent de robots. Néanmoins, cette approche de codage, telle qu'elle était avancée, ne permet pas de représenter toutes les solutions possibles de l'espace de recherche du problème.

— Les auteurs ne considéraient pas les contraintes d'évitement de collisions entre les robots dans leur modèle mathématique. Leur modèle prend en compte toutes les contraintes du problème sauf celles spatiales, liées aux mouvements des robots sur la même voie de circulation.

Ainsi, cette thèse a en particulier pour but de remédier à ces manques et de proposer une nouvelle approche hybride pour la résolution du problème traité.

1.5 Conclusion

Dans ce chapitre, nous avons présenté d'une manière générale le problème d'ordonnancement, et en particulier les problèmes d'ordonnancement d'ateliers. Nous avons décrit les ateliers de traitement de surface avec leurs types de lignes et leurs différentes ressources, aussi bien celles de traitement que celles de manutention. Nous nous sommes, ensuite, intéressés au problème d'ordonnancement rencontré dans ce type d'atelier et connu dans la littérature sous le nom de *Hoist Scheduling Problem* (HSP). Nous en avons présenté une notation, ainsi que les quatre différentes

classes : Le HSP cyclique, le HSP dynamique, le HSP prédictif et le HSP réactif. Comme nous étudions un problème qui rentre dans la catégorie cyclique, nous avons essayé de produire un état de l'art approfondi sur cette variante, aussi bien sur les problèmes qui prennent en compte des lignes à robot unique, que sur ceux qui étudient des lignes à plusieurs robots. Nous avons également organisé cet état de l'art en présentant un panorama de méthodes et approches mises en œuvre pour résoudre ces problèmes. Nous avons, en outre, mis en exergue le fait que ce type de problèmes a été abordé en visant divers objectifs d'optimisation, y compris liés à la conception de la ligne, dont le plus considéré est la minimisation du nombre de robots.

L'étude de la littérature du domaine nous a permis de situer nos travaux par rapport à ceux des autres chercheurs. Nous étudions ainsi, le problème de conception et d'ordonnancement cyclique des robots dans les ateliers de traitement de surface à robots multiples, à cuves de traitement simples (monobacs et monofonctions), et monoproduits. Nous avons appelé ce problème le CHDSP pour *Cyclic Hoist Design and Scheduling Problem*. Les chapitres suivants présentent nos travaux et nos contributions dans la résolution du CHDSP.

TABLE 1.2 – Positionnement de nos travaux dans la littérature scientifique du problème CHSP.

<table>
<tr><td colspan="7">The Cyclic Hoist Scheduling Problem (CHSP)</td></tr>
<tr><td colspan="2">Monorobot</td><td colspan="5">Multirobot</td></tr>
<tr><td>Ligne Simple**</td><td>Ligne Complexe***</td><td colspan="4">Ligne Simple</td><td>Ligne Complexe</td></tr>
<tr><td rowspan="2"></td><td rowspan="2"></td><td colspan="4">Objectifs</td><td rowspan="2"></td></tr>
<tr><td>Ordonnancement</td><td>Conception</td><td colspan="2">Ord. et Conc.*</td></tr>
<tr><td rowspan="2">[Lei & Wang 1989a]
[Baptiste et al. 1992]
[Song et al. 1993]
[Lei 1993]
[Baptiste et al. 1993]
[Armstrong et al. 1994]
[Lei & Wang 1994b]
[Song et al. 1995]
[Levner et al. 1997]
[Ng & Leung 1997]
[Lim 1997]
[Chen et al. 1998b]
[Collart D. & Denat 1998]
[Mateo et al. 2000]
[Che et al. 2002]
[Mateo et al. 2002]
[Mangione et al. 2003b]
[Zhou & Li 2003]
[Xu & Huang 2004]
[Kuntay et al. 2006]
[Subai et al. 2006]
[Lei et al. 2017]
[Wallace & Yorke-S. 2020]</td><td rowspan="2">[Phillips & Unger 1976]
[Shapiro & Nuttle 1988]
[Hanen & Munier 1993]
[Manier 1994]
[Lei & Wang 1994a]
[Ng 1995]
[Kats et al. 1999]
[Liu et al. 2002]
[Bonhomme et al. 2002]
[Zhou & Li 2003]
[Subai et al. 2003]
[Che & Chu 2005]
[Mateo & Companys 2006]
[Che & Chu 2007]
[Mateo & Companys 2007]
[Kats et al. 2008]
[El Amraoui et al. 2008]
[El Amraoui et al. 2009]
[Che et al. 2010]
[El Amraoui et al. 2012]
[Yan et al. 2016]
[Feng et al. 2018]</td><td colspan="3">Codage à base de</td><td rowspan="2">[Shapiro 1985b]
[Manier 1994]
[Hanen & Munier 1994]
[Varnier et al. 1997]
[Manier et al. 2000]
[Riera & Yorke-S. 2002]
[Mahalean & Bloch 2004]
[Zhou & Li 2009]
[Liu et al. 2012]
[Zhao et al. 2013]
[Li et al. 2015]
[Qu et al. 2017]
[Li et al. 2019]
[Alsaffar & Suliman 2020]</td></tr>
<tr><td colspan="2">Mouvements en charge</td><td colspan="2">Mouv. à vide</td></tr>
<tr><td>[Lei & Wang 1991]
[Yang et al. 2001]
[Kats & Levner 2002]
[Leung & Zhang 2003]
[Leung et al. 2004]
[Che & Chu 2004]
[Liu & Jiang 2005]
[Jiang & Liu 2007]
[Che & Chu 2008]
[Zhou & Liu 2008]
[Kharrat 2012]
[Chtourou et al. 2013]
[Che et al. 2013]
[Li & Fung 2013]
[Jiang & Liu 2014]
[Li & Fung 2017]
[Mao et al. 2018]</td><td>[Karzanov & Livshits 1978]
[Kats & Levner 1997a]
[Lei et al. 1993]
[Armstrong et al. 1996]
[Leung & Levner 2006]</td><td colspan="2">Traitement des collisions
Non Oui
[Manier & Lamrous 2006] [Manier & Lamrous 2008] Nos travaux</td></tr>
</table>

* : Bi-objectif : Ordonnancement et Conception.

** : Ligne simple avec des cuves monobacs, (et/ou) des cuves monofonctions, et/ou des produits monogammes.

*** : Ligne complexe avec des cuves multibacs, (et/ou) des cuves multifonctions, et/ou des produits multigammes.

Chapitre 2

CHDSP : Problème et approche de codage

Ce chapitre introduit une nouvelle variante liée au Hoist Scheduling Problem, que nous appelons Cyclic Hoist Design and Scheduling Problem (CHDSP). Elle intègre le dimensionnement des ressources de manutention de l'atelier en plus de l'ordonnancement de celles-ci. Après avoir défini les éléments de ce problème bi-objectif complexe, nous proposons un nouveau codage dédié à la représentation des solutions, dans une optique de résolution par méthode approchée. Nous développons également un premier algorithme de type métaheuristique basé sur ce codage.

Sommaire

Publication

Laajili E., Lamrous S., Manier M-A. et Nicod J-M., "Genetic Algorithm Based Approach for the Multi-Hoist Design and Scheduling Problem", *The 8th International Conference on Industrial Engineering and Systems Management (IEEE IESM 2019)*, Shanghaï, China (september 25-27, 2019), pp.1-6.

2.1 Le CHDSP

2.1.1 Introduction

Dans ce chapitre, nous étudions le CHDSP (*Cyclic Hoist Design and Scheduling Problem*) dans le cadre d'un atelier de traitement de surface. Cet atelier est dédié à une production de masse où un nombre important et illimité de pièces doivent être traitées, en suivant la même séquence de traitement. Le système que nous étudions est composé d'un ensemble de cuves disposées sur une seule ligne dans l'ordre de la gamme opératoire des pièces. Les postes de chargement et de déchargement en entrée et sortie de ligne sont ici associés en un seul et même poste. Plusieurs robots identiques assurent le transfert des porteurs de pièces entre les cuves, en partageant un rail unique (Figure 2.1).

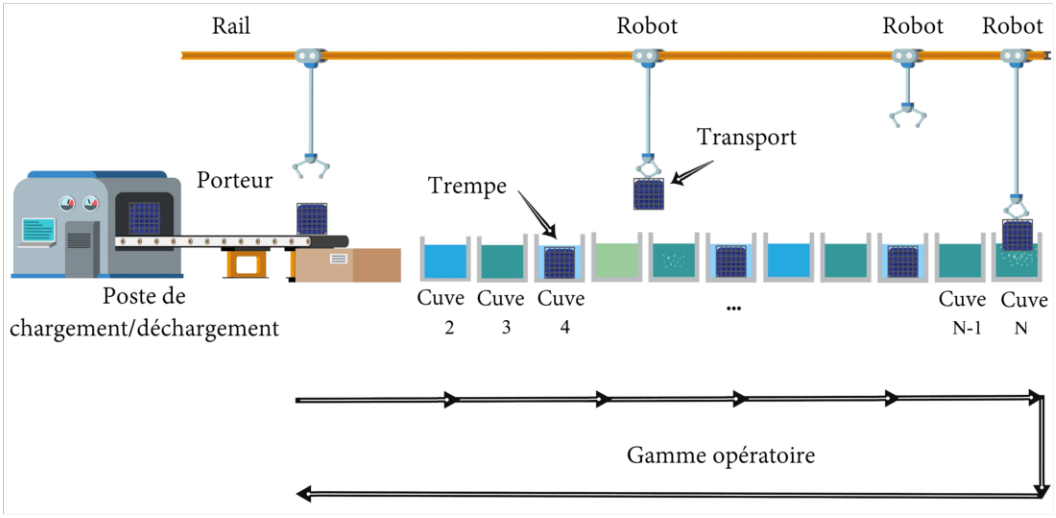

FIGURE 2.1 – La ligne de traitement de surface considérée.

Comme toutes les pièces suivent la même séquence de traitement, la production devient uniforme et une séquence de traitement fixe est répétée. Cela signifie que les robots répètent la même séquence de mouvements alternés. Les durées de déplacement des robots ne peuvent pas être ignorées car elles sont aussi importantes que les temps de traitement. Les robots sont d'ailleurs souvent les ressources critiques de telles lignes.

La répétition de la même séquence constitue *le cycle de production* et la séquence fixe des mouvements des robots est appelée *séquence cyclique*. Nous traitons ainsi le CHDSP avec l'objectif de trouver la séquence cyclique optimale qui donnera le meilleur couple (nombre de robots, temps de cycle). Nous supposons que le nombre de robots est inconnu. Il représente donc une variable de décision du problème. Nous nous attaquons donc à un problème à double-objectifs : minimiser le nombre de robots et minimiser le temps de cycle associé.

Nous décrivons dans la suite, avec plus de détails, la ligne de traitement de surface étudiée avec toutes les hypothèses prises en compte, et donnons les notations qui sont utilisées tout au long de ce manuscrit.

2.1.2 Description du problème

Nous considérons un atelier de traitement de surface avec N cuves. La ligne gère la production cyclique d'un seul type de produits. Les cuves considérées sont des cuves mono-bac et ne permettent pas la ré-entrance des porteurs des pièces (cuves mono-fonction). La ligne possède des stations de chargement-déchargement associées, matérialisées par la cuve numéro 1, comme le montre la figure 2.1. Les cuves de trempe sont alors les cuves dont les indices vont de 2 à N. L'opération de déchargement a pour indice $N+1$, mais se déroule dans la cuve 1. Ainsi la cuve 1 et la cuve $N+1$ sont les mêmes ici. Soit *nop* le nombre des opérations de la séquence de traitements. Les opérations de chargement et de déchargement des pièces sont respectivement la première et la dernière opération de cette séquence. Ainsi, $nop = N+1$.

La ligne emploie H robots identiques. Le robot effectue un mouvement *en charge* lorsqu'il transporte un porteur de pièces entre deux cuves consécutives (i.e., de la cuve i à la cuve $i+1$) de la manière suivante : il soulève le porteur de la première cuve i, s'arrête au-dessus i si nécessaire pour permettre aux pièces de s'égoutter, transporte le porteur vers la cuve suivante $i+1$ et enfin le plonge dans cette dernière. Après chaque mouvement *en charge*, le robot effectue un mouvement *à vide* où il se déplace sans charge vers une autre cuve j de la ligne pour exécuter le prochain mouvement *en charge* programmé (opération de transport). La figure 2.2 schématise la décomposition des ces deux types de mouvements des robots et la présente sur un diagramme spatial temporel.

Nous notons $E(i,j)$ le mouvement à vide d'un robot de la cuve i à la cuve j, et $L(i,i+1)$ le mouvement en charge entre les cuves consécutives i et $i+1$, où $1 \leq i,j \leq N$. Les temps requis pour les mouvements en charge et à vide sont des données constantes du problème et diffèrent d'un atelier à un autre. Soit $d_{i,j}$ le temps de déplacement à vide de la cuve i à la cuve j et r_i le temps de déplacement en charge de la cuve i à la cuve $i+1$. $r_i = d_{i,i+1} + c$ où $d_{i,i+1}$ est le temps de déplacement à vide entre les deux cuves consécutives i et $i+1$, et c une constante qui regroupe les temps nécessaires respectivement pour soulever un porteur, le laisser

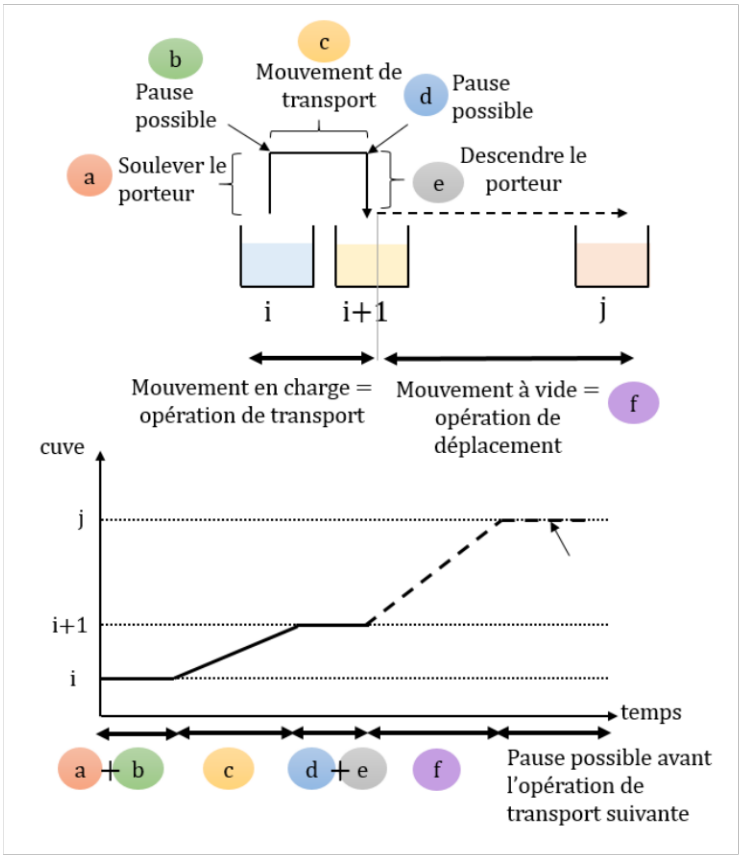

FIGURE 2.2 – Décomposition des mouvements d'un robot.

s'égoutter au-dessus de la cuve i, puis le stabiliser et le descendre dans la cuve $i+1$. Notons que nous considérons c comme une constante, mais un temps tel que la durée d'égouttage peut dépendre de la cuve i et de la nature du bain que celle-ci contient.

Un cycle commence toujours par l'opération de transport $L(1,2)$ car, avant toute chose, il faut déplacer un porteur depuis le poste de chargement (cuve 1) jusqu'à la cuve 2 (première cuve de traitement). Nous supposons que le robot 1 est toujours le responsable de la première opération de transport $L(1,2)$ et donc, du déplacement à vide $E(2,j)$. Nous désignons par T la période du cycle et par H le nombre de robots de la ligne ; $h = 1,\ldots,H$ est le numéro de chaque robot. Que ce soit dans des systèmes à robot unique ou à robots multiples, chaque robot h effectue z_h mouvements et aura une séquence répétitive de mouvements S_h.

2.1.3 Contraintes

Le problème que nous étudions présente de multiples contraintes. Nous les explicitons dans ce qui suit par catégories :

— Les contraintes relatives aux cuves :

- *Contraintes de capacité* : nous considérons que les cuves sont mono-bacs. Ce sont des ressources disjonctives c'est à dire de capacité unitaire, de sorte que chacune d'elles ne peut traiter qu'un seul porteur à la fois ;
- *Contraintes de non re-circulation ou de non ré-entrance* : nous considérons que les cuves sont mono-fonctions, donc chaque porteur ne peut visiter qu'une seule fois chacune des cuves. Ainsi, les cuves sont considérées comme *non ré-entrantes.*

— Les contraintes relatives aux robots :

- *Contraintes de capacité* : chaque robot ne peut transporter qu'un seul porteur à la fois ;
- *Contraintes de disponibilité* : chaque robot doit disposer d'un temps suffisant pour effectuer les mouvements à vide entre les mouvements en charge qui leurs sont assignés ;
- *Contraintes temporelles* : chaque robot doit avoir suffisamment de temps pour effectuer les mouvements de transfert des porteurs entre les cuves ;
- *Contraintes de non-attente* : les robots ne sont pas autorisés de faire des pauses pendant le déplacement d'un porteur, sauf pour l'égouttage, qui est une constante définie par les exigences du processus ;
- *Contraintes de non-collision* : dans le cas de lignes à robots multiples, il faut éviter toute collision éventuelle entre les robots qui partagent le même rail de circulation.

— Les contraintes liées à la gamme de production :

- *Contraintes d'ordre opératoire* : toutes les pièces à traiter sont ici identiques, donc chaque porteur doit suivre la même séquence de traitement ;
- *Contraintes de fenêtres de temps (time window constraints)* : lors des opérations de trempe, chaque porteur doit respecter les limites de temps bornées imposées sur chaque opération de traitement réalisée dans chaque cuve i. En effet, ces temps de trempe doivent se situer entre une limite minimale et une limite maximale de temps (que nous dénotons respectivement par m_i et M_i), qui sont aussi des données du système étudié. Si le temps de trempe se situe en dehors de cet intervalle, les pièces sont alors considérées comme défectueuses.

— Les contraintes liées au processus de production :

- *Contraintes de non-préemption* : aucune opération commencée, qu'elle soit de trempe ou de transport, ne peut être interrompue ;
- *Contraintes de non attente* : les pièces doivent quitter les cuves dès la fin du traitement, sans aucun retard, c'est-à-dire à la limite de temps supérieure imposée par la contrainte de fenêtres de temps ;
- *Contraintes de non stock* : Il n'y a pas de stock tampon entre les opérations de trempe, pour encore empêcher la dégradation des pièces.

En respectant la notation proposée par [Manier & Bloch 2003] et en utilisant les notations avancées, notre problème se note de la manière suivante :

$$CHDSP|H, N-1, 1//ass|/N+1|(T_{min}, H_{min})$$

C'est un problème HSP cyclique, avec les deux objectifs d'optimisation de la conception et de l'ordonnancement, dans une ligne de traitement de surface unique comportant H robots, $N-1$ cuves de traitement (postes de chargement/déchargement non compris dans la notation), et des postes de chargement-déchargement associés ; $N+1$ opérations sont effectuées par chaque porteur. Il s'agit d'un problème bi-objectif, dans lequel on cherche à minimiser à la fois la période T et le nombre de robots H.

2.1.4 Approche générale de résolution

Comme évoqué en synthèse du premier chapitre (paragraphe 1.4.4), nous avons choisi d'utiliser la méthode de codage et de décodage basée sur les mouvements à vide des robots, car elle permet de représenter des solutions qui peuvent correspondre à différents nombres de robots H : une ou plusieurs séquences de mouvements cycliques S_h peuvent être générées à partir de chaque solution. Le nombre de robots H n'est donc pas une donnée du problème mais plutôt une variable de décision. Par conséquent, cette méthode de codage-décodage des solutions nous permet de traiter l'aspect de conception de la ligne qui se déploie en la minimisation du nombre de robots.

Cette méthode a été proposée pour la première fois par Manier et Lamrous en 2006 et 2008 [Manier & Lamrous 2006, Manier & Lamrous 2008]. Elle est intéressante car la quasi-totalité des travaux de la littérature réalisés sur le CHSP à robots multiples a utilisé les mouvements en charge des robots pour encoder-décoder les solutions et déduire les séquences de mouvements [Lei & Wang 1991, Lei *et al.* 1993, Armstrong *et al.* 1996, Yang *et al.* 2001, Leung & Zhang 2003, Leung *et al.* 2004, Che & Chu 2004, Liu & Jiang 2005, Leung & Levner 2006, Jiang & Liu 2007, Che & Chu 2008, Zhou & Liu 2008, Kharrat 2012, Chtourou *et al.* 2013, Che *et al.* 2013, Li & Fung 2013, Jiang & Liu 2014, Li & Fung 2017, Mao *et al.* 2018]. Le décodage suivant cette méthode résulte toujours en un même nombre de séquences de mouvements, qui correspond au nombre de robots déjà défini dans le problème à résoudre. La taille de la flotte des robots est donc une donnée fixe et connue au préalable, à laquelle les opérations de transport sont affectées puis ordonnancées.

Néanmoins, le codage proposé dans [Manier & Lamrous 2006, Manier & Lamrous 2008], tel qu'il a été suggéré, ne permet pas de représenter toutes les solutions possibles de l'espace de recherche du problème. Ainsi, ce chapitre a en particulier pour but de remédier à ce problème en proposant une amélioration de l'approche initiale de codage à base des mouvements à vide. Pour tester l'efficacité de cette approche d'amélioration, nous avons développé un algorithme de résolution général basé sur un *algorithme génétique*. Cet algorithme repose sur la méthode améliorée de

codage et de décodage des solutions à base de mouvements à vide et sur un modèle mathématique de type programme linéaire mixte en nombres entiers (*MILP : Mixed Integer Linear Program*). La figure 2.3 présente les éléments utiles à cette approche générale de résolution du CHDSP, que nous explicitons ici :

— Le codage amélioré des solutions : il est employé pour la génération des solutions qui sont encodées à base des mouvements à vide ;

— La procédure de décodage associée à la méthode de codage adoptée : elle décode les solutions pour former une ou plusieurs séquences de mouvements qui sont affectées aux robots. Le nombre de séquences déduit de chaque solution donne le nombre de robots affectés à la ligne ;

— Le test de faisabilité des séquences trouvées : il est réalisé en résolvant un modèle mathématique de type *MILP* intégrant les contraintes du problème. Pour la(les) séquence(s) faisable(s), le modèle renvoie les dates d'exécution des mouvements des robots et le temps de cycle T résultant ;

— L'algorithme général de résolution : il met en œuvre un *algorithme génétique* qui optimise la recherche des solutions faisables et tend à trouver les meilleures solutions par nombre de robots.

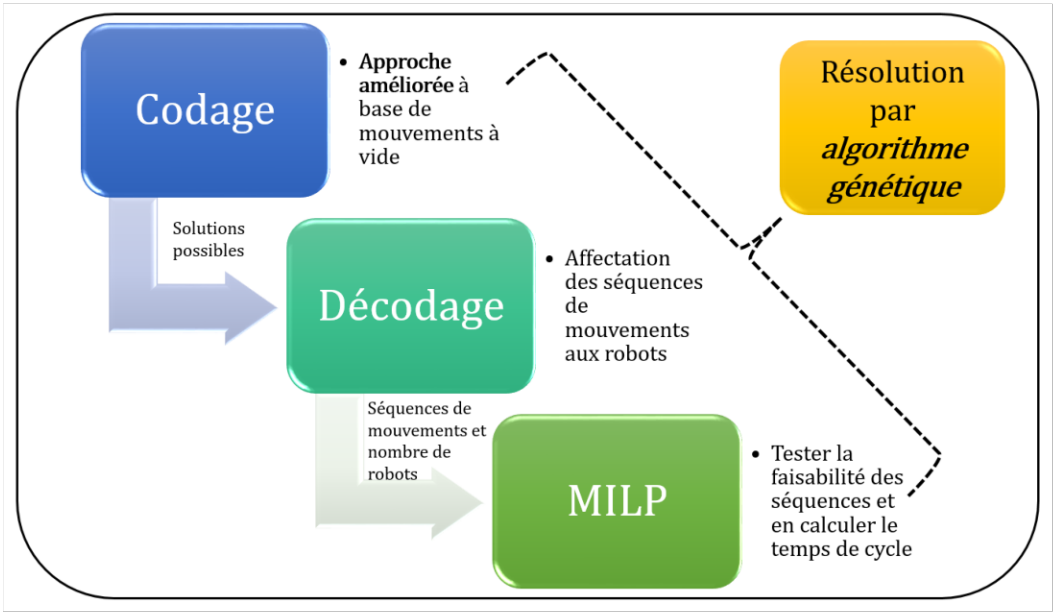

FIGURE 2.3 – Approche de résolution du CHDSP.

Nous consacrons le paragraphe suivant à la présentation de l'approche initiale ainsi que l'approche d'amélioration du codage-décodage des solutions.

2.2 Codage-décodage des solutions

2.2.1 Analyse préliminaire

Le codage à base de mouvements à vide est tout à fait équivalent au codage basé sur les mouvements en charge en termes d'obtention de séquences de mouvements cycliques. En effet, le problème ne permet ni attente ni préemption lors des opérations de traitement. Par conséquent, le début d'une opération de transport correspond exactement à la fin d'une opération de trempe et la fin d'une opération de transport correspond au début d'une opération de trempe. Ainsi, ordonnancer les opérations de traitement équivaut à ordonnancer les mouvements de transport. Et comme la séquence répétitive des mouvements des robots est composée d'une alternance de mouvements en charge et à vide, l'ordonnancement des mouvements des robots est possible soit avec les mouvements en charge, soit avec les mouvements à vide, à partir duquel la séquence cyclique finale peut facilement être déduite.

Par ailleurs, les deux types de codage-décodage diffèrent en termes de nombre de séquences cycliques qu'ils peuvent induire. La figure 2.4 montre cette différence sur l'exemple de la solution encodée $\{1, 3, 2, 4\}$. D'une part, si nous considérons que cette solution est encodée selon les mouvements en charge des robots, alors chaque nombre figurant sur cette solution correspond au numéro de la cuve d'origine d'un mouvement en charge. Ainsi, les mouvements en charge correspondant à cette solution sont $L(1,2), L(3,4), L(2,3)$ et enfin $L(4,1)$. En les liant avec les mouvements à vide qui les séparent, on obtient une seule séquence de mouvements qui est la suivante, et qui est associée à un seul robot ($S_h = 1$ et $H = 1$) : $L(1,2) - E(2,3) - L(3,4) - E(4,2) - L(2,3) - E(3,4) - L(4,1) - E(1,1)$.

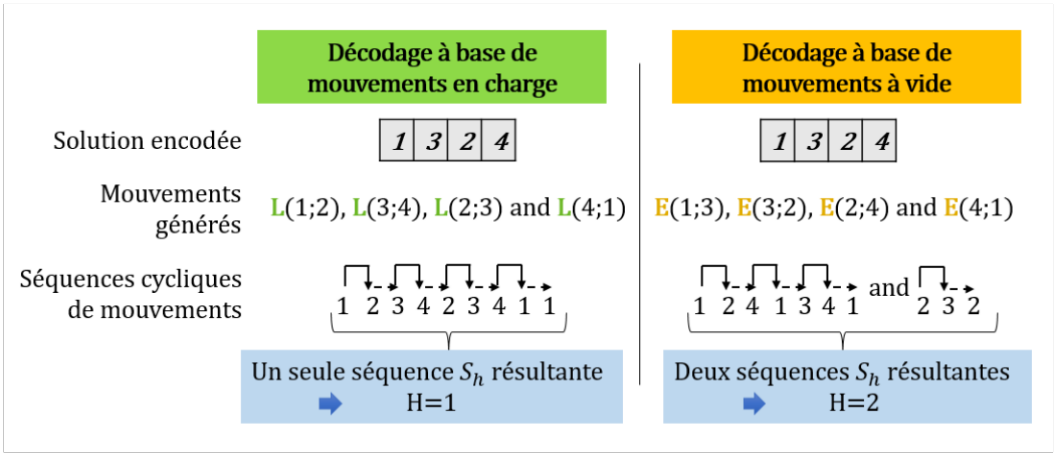

FIGURE 2.4 – Séquences résultant des deux types de codage à partir d'une même solution encodée.

D'autre part, si nous considérons que cette solution représente les mouvements à vide, alors chaque deux nombres successifs figurant sur cette solution constituent un

mouvement à vide : le premier nombre correspond au numéro de la cuve de départ de ce mouvement et le deuxième nombre correspond au numéro de la cuve d'arrivée de ce dernier. Le principe de la génération des mouvements à vide à partir des solutions est expliqué en détail dans la sous-section suivante 2.2.2. Toutefois nous en mentionnons ici l'essentiel pour expliquer cet exemple. Avec cette méthode, ainsi, les mouvements à vide déduits de cette solution sont $E(1,3), E(3,2), E(2,4)$ et $E(4,1)$. En les liant avec les mouvements en charge qui les précèdent, on arrive à obtenir deux séquences cycliques de mouvements donc deux robots ($S_h = 2$ et $H = 2$). Ces deux séquences sont les suivantes : $L(1,2) - E(2,4) - L(4,1) - E(1,3) - L(3,4) - E(4,1)$, et $L(2,3) - E(3,2)$.

Cet exemple souligne en particulier la capacité du codage basé sur les mouvements à vide à fournir une affectation de séquences à plusieurs robots, là où la même solution encodée, en suivant le codage à base des mouvements en charge, ne fournit qu'une affectation mono-robot.

2.2.2 Approche initiale

Dans l'approche initialement proposée par [Manier & Lamrous 2008], une solution est représentée par toute combinaison possible de numéros de cuves. Ces numéros de cuves sont dénotés par i, avec $i = 1, \ldots, N$ et N étant le nombre total de cuves de la ligne (y compris le poste de chargement/déchargement). Ainsi, i fait référence au numéro de la cuve ou encore à l'opération de trempe réalisée dans cette cuve i, sauf pour l'opération de déchargement, qui se déroule dans la cuve 1 alors qu'elle est considérée comme la $N + 1^{\text{ème}}$ opération de traitement.

Nous désignons par O toute solution possible (quelconque). Chaque couple de deux numéros successifs de cette solution forme un mouvement à vide : le premier numéro représente le numéro de la cuve de départ de ce mouvement, et le deuxième en représente le numéro de la cuve d'arrivée. Le codage considéré est cyclique, ce qui signifie que le dernier couple de cuves de chaque solution est composé du dernier numéro de cuve de cette solution puis de son premier numéro de cuve. Par exemple, si nous considérons une solution $\{ijk\}$, nous en retirons trois couples formant trois mouvements à vide : $E(i,j)$, $E(j,k)$ et $E(k,i)$.

Un codage basé sur les mouvements en charge nécessite de connaître et donc de représenter tous les mouvements en charge réalisés dans le cycle. Comme ces mouvements en charge sont issus de chacune des cuves de la ligne, une solution encodée sous cette méthode de codage, doit absolument contenir tous les numéros de cuves de la ligne. Chaque solution aura ainsi une taille égale à N (le nombre des cuves de la ligne). Cependant, avec un codage à base de mouvements à vide, nous avons au plus N mouvements à vide effectifs au cours d'un cycle, donc au plus N numéros de cuves dans les solutions. En effet, si un numéro de cuve n'apparaît pas dans la solution, cela signifie qu'un robot attend au-dessus d'une cuve jusqu'à la

fin de l'opération de trempe d'un porteur qu'il vient de déposer avant d'entamer l'opération suivante de transport.

Dès lors, avec le codage basé sur les mouvements à vide, la taille d'une solution peut varier entre 2 et N. Quand cette taille est inférieure à N, certains numéros de cuves sont absents, qui forment alors des mouvements à vide dits *fictifs*. Par exemple, si nous reprenons la solution $\{ijk\}$ et que l est un numéro de cuve absent de celle-ci, alors nous devons considérer le *mouvement à vide fictif* $E(l, l)$. De cette manière, la solution a bien quatre mouvements à vide au total, au lieu de seulement trois. La figure 2.5 présente deux exemples de solutions dans un problème à six cuves. La première solution $\{1, 5, 2, 6, 3\}$ est de taille égale à 5. Son décodage permet d'identifier six mouvements à vide parmi lesquels un mouvement fictif, qui est le résultat de l'absence de la cuve 4 dans la solution. Les mouvements identifiés sont ainsi $E(1, 5)$, $E(5, 2)$, $E(2, 6)$, $E(6, 3)$, $E(3, 1)$, et le mouvement à vide fictif $E(4, 4)$. La deuxième solution $\{4, 2, 5, 6, 1, 3\}$ est de taille égale à six. Elle contient tous les numéros de cuves de la ligne, et n'induit donc pas de mouvements à vide fictifs. Son décodage permet de donner les mouvements à vide suivants : $E(4, 2)$, $E(2, 5)$, $E(5, 6)$, $E(6, 1)$, $E(1, 3)$ et $E(3, 4)$.

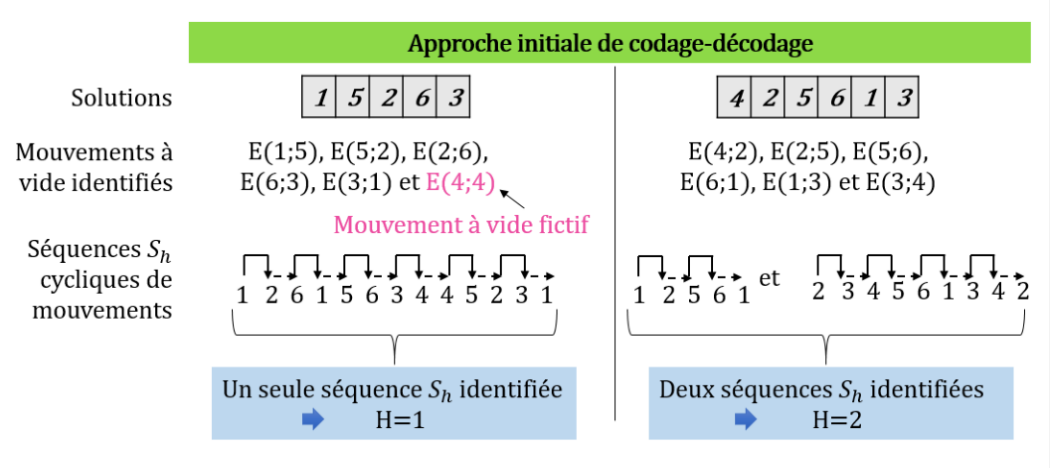

FIGURE 2.5 – Procédure de décodage de solutions en mouvements à vide et d'identification des séquences de mouvements et du nombre de robots associés.

Une fois réalisée l'identification de tous les mouvements à vide d'une solution, un nombre H de séquences de mouvements cycliques S_h peut être déduit. La construction des séquences des mouvements repose sur le principe des mouvements alternés. Comme mentionné au début du paragraphe 2.2, la séquence de mouvements d'un robot est une alternance entre mouvements en charge et mouvements à vide. Nous montrons cette alternance dans la figure 2.6, qui fournit le diagramme de GANTT de la première solution $\{1, 5, 2, 6, 3\}$ de la figure 2.5. Par exemple, si nous considérons le mouvement à vide $E(6, 3)$ sur ce graphe, alors celui-ci est précédé par l'opération de transport (mouvement en charge) $L(5, 6)$ et suivi par $L(3, 4)$.

S'il s'agit d'un mouvement à vide fictif tel que $E(4,4)$ (attente du robot au-dessus de la cuve 4), celui-ci est précédé par l'opération de transport $L(3,4)$ et suivi par l'opération de transport $L(4,5)$.

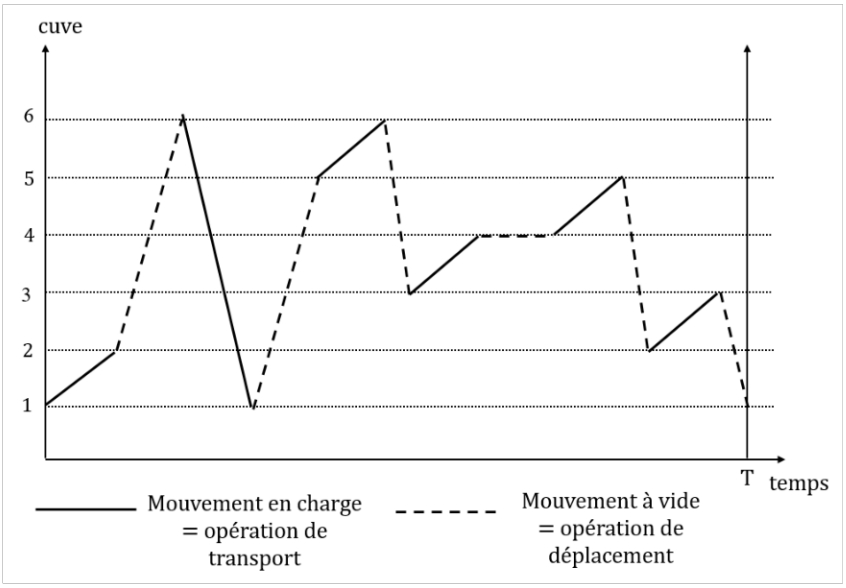

FIGURE 2.6 – Diagramme GANTT associé à la solution $\{1,5,2,6,3\}$.

Lors de la construction des séquences cycliques, et sans perte de généralité, nous commençons toujours par une première séquence affectée au robot numéro 1 ($h = 1$), qui débute par l'opération $L(1,2)$ de transport du porteur, depuis la station de chargement (cuve 1) vers la première cuve de trempe (cuve 2). Nous enchaînons ensuite avec le mouvement à vide qui commence par le numéro de cuve 2. Nous continuons la construction de cette manière en alternant les mouvements en charge et les mouvements à vide jusqu'à ce que la séquence de mouvements boucle : le dernier mouvement à vide de cette séquence finit par le numéro de cuve origine du premier mouvement en charge de ladite séquence. Dans le cas de la première séquence, le dernier mouvement à vide finit par le numéro 1 car le premier mouvement en charge est le $L(1,2)$. A l'issue de la construction de la première séquence, nous vérifions s'il reste encore des mouvements à vide dans la solution et qui n'ont pas été affectés à cette première séquence. Si le test est négatif, alors la procédure de construction des séquences de mouvements est finalisée et le nombre de séquences déduites est égal à 1 correspondant à un nombre de robot égal à 1 ($H = 1$). Si le test est positif, alors la procédure de construction des séquences de mouvements est non achevée et nous commençons la construction d'une seconde séquence : celle-ci commence par le mouvement en charge qui n'a pas encore été attribué dans la première séquence et qui a le plus petit numéro de cuve de départ. Ce mouvement est le $L(i(restants)_{min}, i(restants)_{min} + 1)$. Nous enchaînons de la sorte jusqu'à l'épuisement de tous les mouvements à vide et en charge à réaliser. Le nombre résultant de séquences S_h de mouvements donne le nombre de robots H à

associer à la ligne. Par conséquent, avec le codage basé sur les mouvements à vide, nous sommes en mesure d'identifier à partir de chaque solution O possible, la ou les séquences cycliques de mouvements S_h, correspondant chaque fois à un nombre de robots H différent.

La figure 2.5 illustre bien cette capacité à l'aide des deux exemples de solutions proposées, considérées, dans un problème à six cuves ($N = 6$) dans lequel tous les mouvements en charge ($L(1,2)$, $L(2,3)$, $L(3,4)$, $L(4,5)$, $L(5,6)$ et $L(6,1)$) sont à réaliser exactement une fois au cours du cycle. Comme nous l'avons vu, la première solution $\{1,5,2,6,3\}$ de taille 5, après décodage, fournit les mouvements à vide $E(1,5)$, $E(5,2)$, $E(2,6)$, $E(6,3)$, $E(3,1)$, et $E(4,4)$. Nous pouvons alors construire la première séquence de mouvements associée $S_h = S_1$ en commençant par le mouvement en charge $L(1,2)$, et en suivant le principe d'alternance entre les mouvements et d'arrêt au bouclage de la séquence. De cette manière, la séquence résultante est la suivante : $L(1,2) - E(2,6) - L(6,1) - E(1,5) - L(5,6) - E(6,3) - L(3,4) - E(4,4) - L(4,5) - E(5,2) - L(2,3) - E(3,1)$. Elle inclut tous les mouvements en charge et à vide possibles. Nous obtenons ainsi une seule séquence de mouvements $S_h = S_1$ correspondant à une ligne de traitement mono-robot ($H = 1$). La représentation par un diagramme de GANTT de cette séquence est donnée à la figure 2.6.

Quant à la deuxième solution $\{4,2,5,6,1,3\}$ de taille 6, après décodage, nous en identifions les mouvements à vide $E(4,2)$, $E(2,5)$, $E(5,6)$, $E(6,1)$, $E(1,3)$ et $E(3,4)$. En suivant la procédure de construction décrite ci-dessus, nous obtenons deux séquences cycliques $S_h = \{S_1, S_2\}$, et donc une ligne à deux robots ($H = 2$). La première séquence cyclique de mouvements $S_h = S_1$ associée au robot 1 ($h = 1$) commence par le mouvement en charge $L(1,2)$ et finit par le mouvement à vide $E(6,1)$ qui ferme le cycle : $L(1,2) - E(2,5) - L(5,6) - E(6,1)$. La deuxième séquence cyclique de mouvements $S_h = S_2$ associée au robot 2 ($h = 2$) commence par un mouvement en charge qui n'a pas été attribué à la première séquence et qui a le plus petit numéro de cuve de départ. Les mouvements en charge restants après la construction de la première séquence sont $L(2,3)$, $L(3,4)$, $L(4,5)$ et $L(6,1)$ et celui qui a le plus petit numéro de cuve de départ est $L(2,3)$. Ainsi, la séquence résultante est la suivante : $L(2,3) - E(3,4) - L(4,5) - E(5,6) - L(6,1) - E(1,3) - L(3,4) - E(4,2)$. La séquence est cyclique car le mouvement à vide $E(4,2)$ ferme le cycle pour recommencer à nouveau du mouvement en charge $L(2,3)$. Cette deuxième séquence épuise tous les mouvements en charge restants, ce qui termine la procédure de construction.

2.2.3 Approche d'amélioration

2.2.3.1 Inconvénient du codage existant

Sur la base de l'analyse de leurs résultats, les auteurs de [Manier & Lamrous 2006] ont conclu que le codage basé sur les mouvements à vide qu'ils ont proposé, présente une limite concernant l'exploration de l'espace de recherche. En effet, il ne permet

pas de générer toutes les solutions possibles et admissibles de l'espace de recherche du problème et donc, éventuellement, les solutions optimales. Ce problème est illustré par la figure 2.7 qui montre que la solution optimale à un robot du benchmark de Phillips and Unger à 13 cuves ($N = 13$) ([Phillips & Unger 1976]) ne peut pas être représentée par ce codage.

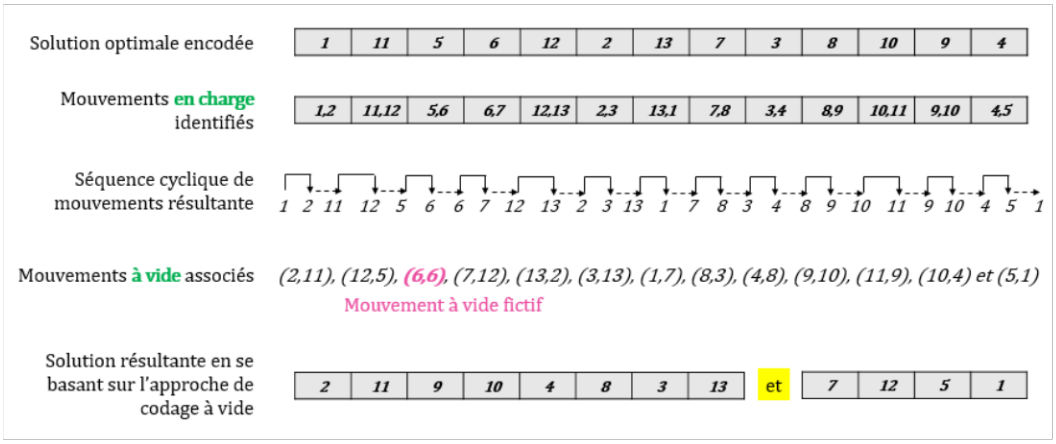

FIGURE 2.7 – Analyse de possibilité de codage à base des mouvements à vide sur une solution optimale à un robot de l'instance de [Phillips & Unger 1976].

La première ligne de la figure 2.7 fournit la solution optimale à un seul robot de l'instance, exprimée par un codage classique des mouvements en charge. La deuxième ligne explicite ces mouvements en charge. La troisième ligne schématise la séquence cyclique de mouvements qui résulte de la liste des mouvements en charge identifiés. La quatrième ligne extrait de cette séquence la liste des mouvements à vide associés, parmi lesquels nous identifions un seul mouvement à vide fictif qui est $E(6, 6)$. Nous déduisons enfin sur la dernière ligne de cette figure, l'éventuelle solution résultante encodée suivant l'approche de codage à base de mouvements à vide. Nous nous rendons compte que cette dernière ne comporte pas une mais deux listes distinctes de numéros de cuves : $\{2, 11, 9, 10, 4, 8, 3, 13\}$ et $\{7, 12, 5, 1\}$, alors que le codage à vide ne devrait comporter qu'une seule liste cyclique, quel que soit le nombre de robots. Ainsi, avec ce codage, la solution optimale du problème de Phillips [Phillips & Unger 1976] ne peut pas être obtenue.

Nous déduisons de manière plus générale, que l'approche de codage basée sur les mouvements à vide telle que proposée par [Manier & Lamrous 2006] est limitée car elle ne peut pas représenter toutes les solutions de l'espace de recherche.

2.2.3.2 Nouveau codage

Pour remédier au problème identifié, tout en conservant les mêmes propriétés de codage-décodage à base de mouvements à vide déjà déployées dans le paragraphe 2.2.2, nous suggérons d'introduire des séparateurs dans le codage des so-

lutions. Ces derniers vont permettre de rassembler plusieurs listes de numéros de cuves dissociées appartenant à une même solution sur une seule liste qui représente la solution. Ils vont permettre aussi de marquer une rupture lors du décodage d'une solution donnée, pour en déduire la ou les séquences de mouvements associées.

Nous encodons un séparateur par le chiffre 0, chiffre qui n'est pas utilisé comme numéro de cuve i ($i \in \{1, \ldots, N\}$ donc $i \neq 0$). En conséquence, la définition d'une solution change pour s'adapter avec cette approche d'amélioration du codage à base de mouvements à vide : *une solution est toute combinaison possible de numéros de cuves i avec $i \in \{1, \ldots, N\}$ avec un nombre limité de chiffres* 0 *qui représentent le ou les séparateurs, ce nombre pouvant être nul.* Nous revenons à l'exemple de solution affichée en figure 2.7, et nous montrons sur la figure 2.8 la nouvelle solution résultante encodée selon notre approche améliorée.

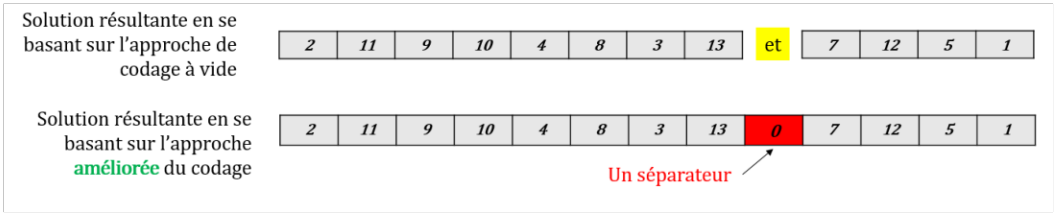

FIGURE 2.8 – Nouvelle solution encodée selon l'approche améliorée par les séparateurs pour la solution du problème [Phillips & Unger 1976].

L'ajout de séparateurs à l'encodage adopté nous permet de générer plus de solutions que la première version du codage. Nous le schématisons sur la figure 2.9 par les graphes des mouvements à vide qui correspondent à deux solutions du benchmark de [Phillips & Unger 1976]. La première solution est la nouvelle solution encodée selon l'approche améliorée avec séparateurs (voir figure 2.8). Le graphe associé à cette solution se compose de trois circuits : le premier lie 8 sommets (cuves 10, 4, 8, 3, 13, 2, 11, et 9) ; le deuxième a un seul sommet (cuve 6) ; et le troisième lie 4 sommets (cuves 7, 12, 5, et 1). La deuxième solution considérée reprend la même suite de numéros de cuves que la première mais sans séparateur. Le graphe associé est composé de deux circuits : le premier lie 12 sommets (cuves 10, 4, 8, 3, 13, 7, 12, 5, 1, 2, 11, et 9) et le deuxième a un seul sommet qui correspond à la cuve 6.

Par conséquent, l'ajout de séparateurs à l'encodage adopté nous permet de générer des solutions qui n'étaient pas atteignables avec l'approche de codage initiale, et donc d'augmenter le nombre de solutions représentables de l'espace de recherche. Cette approche d'amélioration permet de diversifier davantage l'espace de recherche, voire de le compléter.

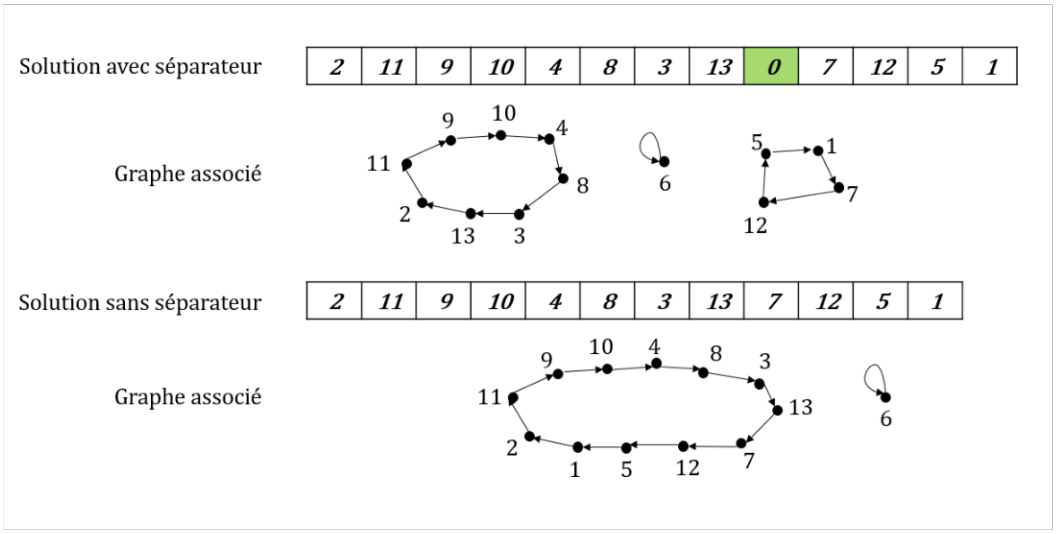

FIGURE 2.9 – Graphes multi-circuits pour des solutions avec et sans séparateur.

2.2.3.3 Décodage

Avec l'ajout des séparateurs dans le codage des solutions, l'opération de décodage initiale doit aussi être adaptée. Le principe d'identification des mouvements à vide est donc modifié avec la présence des séparateurs. En effet, un séparateur prend place entre deux numéros consécutifs de cuves dans une solution. Nous désignons alors cette position par le terme *position inter-cuve*. Ainsi, chaque séparateur existant dans la solution divise cette dernière en deux parties : il succède au dernier numéro de cuve de la première partie et précède le premier numéro de cuve de la deuxième partie. La procédure de décodage reste la même pour chaque partie identifiée.

Le décodage d'une solution reste cyclique. De même, le décodage de chaque partie de la solution doit également être cyclique. Nous le montrons sur l'exemple de solution situé à droite sur la figure 2.10. Dans la solution encodée $\{4, 2, 0, 5, 6, 1, 3\}$, le séparateur occupe la *deuxième position inter-cuve*, entre les cuves 2 et 5. Ce séparateur engendre la séparation de la solution en deux parties : $\{4, 2\}$ et $\{5, 6, 1, 3\}$). Les mouvements à vide identifiés de la première partie sont alors $E(4, 2)$ et $E(2, 4)$; ceux relatifs à la deuxième partie sont $E(5, 6)$, $E(6, 1)$, $E(1, 3)$ et $E(3, 5)$. Nous obtenons une seule séquence cyclique de mouvements $(S_h = S_1) : L(1, 2) - E(2, 4) - L(4, 5) - E(5, 6) - L(6, 1) - E(1, 3) - L(3, 4) - E(4, 2) - L(2, 3) - E(3, 5) - L(5, 6) - E(6, 1)$. Elle correspond à un seul robot à associer à la ligne $(H = 1)$.

La partie gauche de la figure 2.10 présente le résultat du décodage de la même liste ordonnée de cuves, mais sans séparateur : $\{4, 2, 5, 6, 1, 3\}$. Les mouvements à vide identifiés à partir de cette solution sont : $E(4, 2)$, $E(2, 5)$, $E(5, 6)$, $E(6, 1)$, $E(1, 3)$ et $E(3, 4)$. A partir de ces mouvements et en respectant la procédure de construction des séquences de mouvements, détaillée précédemment dans le paragraphe 2.2.2, nous obtenons deux séquences cycliques $S_h = \{S_1, S_2\} : L(1, 2) - E(2, 5) - L(5, 6) - E(6, 1)$

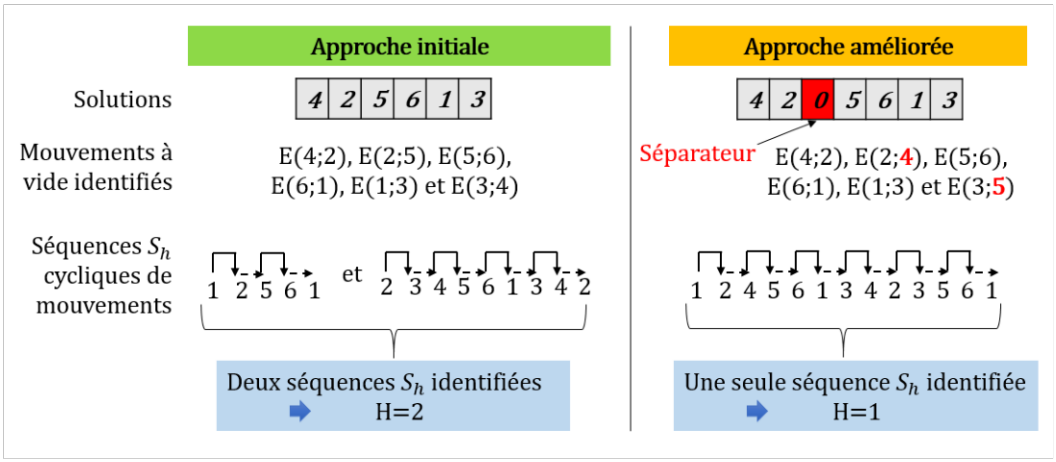

FIGURE 2.10 – Séquences générées à partir d'une solution sans et avec séparateur.

et $L(2,3) - E(3,4) - L(4,5) - E(5,6) - L(6,1) - E(1,3) - L(3,4) - E(4,2)$. La solution correspond donc à un atelier à deux robots ($H = 2$), à la différence du résultat obtenu sur la même solution avec séparateur. Ainsi, cette comparaison entre les deux solutions montre à nouveau que l'ajout d'un séparateur a un impact sur le couple (H, S_h) résultant.

Dans la suite de ce manuscrit, nous appelons approche de codage basée sur les mouvements à vide, l'approche de codage initiale que nous avons améliorée par insertion de séparateurs.

2.2.3.4 Mise en œuvre du nouveau codage-décodage

Nous avons fourni dans le paragraphe 2.2.3.2, la définition d'une solution avec le codage amélioré : Il s'agit de *toute combinaison possible de numéros de cuves i avec $i \in \{1, \ldots, N\}$ avec un nombre limité de chiffres* 0 *qui représentent le ou les séparateurs, ce nombre pouvant être nul.* Ainsi, la taille totale d'une solution, que nous notons $T_{solution}$, est la somme du nombre des entiers i associés aux cuves avec le nombre de séparateurs utilisés (nombre de fois qu'apparaît le chiffre 0). Nous notons aussi par NC le nombre des numéros de cuves dans une solution et par NS le nombre de séparateurs qui y sont intégrés. Nous avons alors :

$$T_{solution} = NC + NS$$

NC correspond à la taille d'une solution avant intégration de séparateurs. Nous supposons que cette taille se situe entre 2 et N :

$$2 \leq NC \leq N$$

Pour déterminer la taille totale d'une solution avec séparateurs, il faut alors

connaître le nombre NS de séparateurs qui lui sont intégrés. Comme nous ciblons la génération de toutes les solutions admissibles dans l'espace de recherche du problème, nous devons générer les solutions avec tous les séparateurs possibles. NS est compris entre 0 et un nombre maximal de séparateurs par solution, noté MNS :

$$0 \leq NS \leq MNS$$

En effet, nous pouvons ne pas appliquer de séparateurs ($NS = 0$), ce qui revient au cas du codage initial de [Manier & Lamrous 2006]. Avec notre codage, nous pouvons aussi insérer un nombre aléatoire de séparateurs, pour participer à la diversification des solutions et à l'objectif d'atteindre toutes les solutions possibles. Par ailleurs, il semble nécessaire de connaître le nombre maximal MNS de séparateurs qui peuvent être appliqués à une solution. Nous savons qu'une solution sans séparateur de taille NC comporte $(NC - 1)$ positions inter-cuves. Pour déterminer MNS, nous avons tout d'abord analysé la structure de notre codage pour identifier les positions possibles d'insertion des séparateurs parmi toutes les positions inter-cuves d'une solution. Une *position possible* est définie par *toute position inter-cuve d'une solution sur laquelle l'ajout d'un séparateur mène à une nouvelle solution qui donnera une séquence de mouvements différente de la première séquence (sans séparateur).*

*** 1ère constatation**

L'insertion d'un séparateur en première (respectivement dernière) position inter-cuves équivaut à une solution sans ce séparateur dans laquelle le premier (respectivement dernier) élément (numéro de cuve) de la solution n'apparaît pas.

La démonstration est la suivante : l'application d'un séparateur en première position inter-cuve (autrement dit, entre le premier et le deuxième numéro de cuves) d'une solution de taille NC divise la solution en deux parties : un premier segment de taille égale à 1 et un deuxième segment de taille égale à $(NC - 1)$. Le premier segment est composé d'un seul numéro de cuve i, qui correspond au premier numéro de cuve de la solution de base. Le décodage de ce premier segment identifie ainsi un seul mouvement à vide fictif $E(i, i)$, et un cycle de taille $NC - 1$ ne contenant pas i. De même, l'application d'un séparateur sur la dernière position inter-cuve (autrement dit, entre le dernier et l'avant dernier numéro de cuves) d'une solution de taille NC terminant par le numéro de cuve j, conduit à l'identification du mouvement à vide fictif $E(j, j)$, et d'un cycle de taille $NC - 1$ ne contenant pas j.

La figure 2.11 illustre cette observation, avec une solution initiale $\{4, 2, 5, 6, 1, 3\}$ de taille $NC = 6$, pour un problème à six cuves ($N = 6$). Sur sa partie gauche, nous appliquons un séparateur en première position inter-cuve de cette solution, entre les cuves 4 et 2. La solution devient : $\{4, 0, 2, 5, 6, 1, 3\}$, à laquelle sont associés les mouvements à vide $E(4, 4)$, $E(2, 5)$, $E(5, 6)$, $E(6, 1)$, $E(1, 3)$ et $E(3, 2)$. Par ailleurs, nous affichons la solution $\{2, 5, 6, 1, 3\}$ de taille $NC = 5$ qui correspond au deuxième

segment de taille $NC-1$ de la solution initiale après l'application du séparateur. A partir de cette solution, nous identifions les mouvements à vide $E(2,5)$, $E(5,6)$, $E(6,1)$, $E(1,3)$, $E(3,2)$, auxquels nous ajoutons le mouvement fictif $E(4,4)$ résultant de l'absence de la valeur 4 dans la solution. Le décodage des deux solutions donne le même résultat en termes de mouvements à vide, elles sont donc équivalentes. Le même raisonnement conduit sur l'exemple en partie droite de la figure conduit à déduire que les solutions $\{4,2,5,6,1,0,3\}$ et $\{4,2,5,6,1\}$ sont identiques, avec les mêmes mouvements à vide, y compris le mouvement fictif $E(3,3)$.

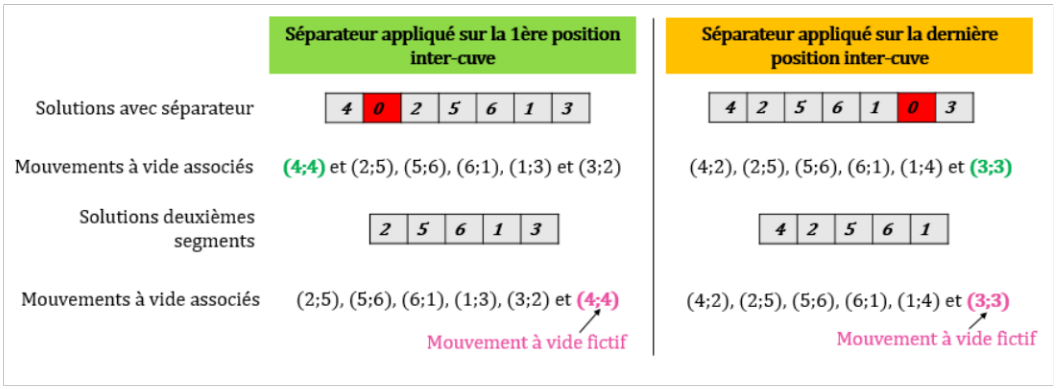

FIGURE 2.11 – Deux positions inter-cuves à éviter pour les séparateurs.

Par conséquent, pour notre codage, nous inhibons toute application de séparateurs sur la première ou la dernière position inter-cuve d'une solution.

*** 2ème constatation**

Une autre configuration de positionnement des séparateurs doit être exclue : celle de deux séparateurs placés sur deux positions inter-cuves successives d'une solution.

La démonstration est la suivante : l'application de deux séparateurs sur deux positions inter-cuves successives d'une solution de taille NC divise la solution en trois parties : un premier segment délimité par les deux séparateurs de taille égale à 1 et deux autres segments qui se retrouvent de part et d'autre de chacun des deux séparateurs et dont la taille totale est égale à $(NC-1)$. Le premier segment, délimité par les deux séparateurs, est composé d'un seul numéro de cuve i, qui correspond au mouvement à vide fictif $E(i,i)$. La figure 2.12 illustre ce résultat pour un problème à 6 cuves ($N=6$), et une solution $\{4,2,0,5,0,6,1,3\}$, dans laquelle deux séparateurs se trouvent sur deux positions inter-cuves successives (positions 2 et 3), donc séparés par un seul numéro de cuve (5). Les mouvements à vide associés à cette solution sont $E(4,2)$, $E(2,4)$, $E(5,5)$, $E(6,1)$, $E(1,3)$ et $E(3,6)$. Or ces mêmes mouvements sont également associés à la solution $\{4,2,0,6,1,3\}$ où le numéro de cuve 5 est absent et où un seul séparateur au lieu de deux est placé entre les cuves 2 et 6. Les deux solutions sont donc équivalentes.

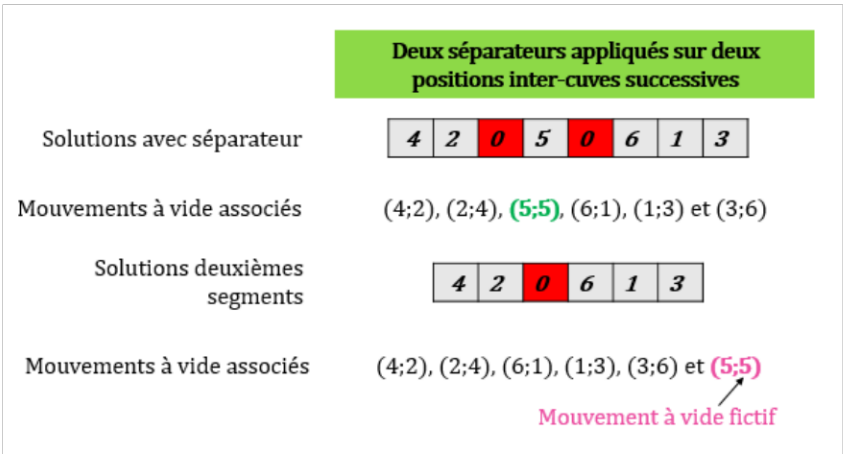

FIGURE 2.12 – Autre configuration à éviter lors de l'application des séparateurs.

Nous concluons ainsi que cette configuration d'application de séparateurs sur deux positions inter-cuves successives a les mêmes conséquences que l'application de séparateurs sur la première ou la dernière position inter-cuve.

*** Synthèse**

Il devient évident que l'application de séparateurs sur les premières ou les dernières positions inter-cuves ou sur deux positions inter-cuves successives d'une solution, entraîne à obtenir les mêmes mouvements à vide associés que d'autres solutions représentées plus simplement. Ces positions ne présentent donc pas un avantage dans l'exploration de nouvelles solutions. Au contraire, elles mènent à la répétition d'un certain nombre de séquences de mouvements générées. Nous choisissons alors d'interdire l'application de séparateurs sur de telles positions lors de la génération des solutions, pour éviter une redondance des solutions et une perte du temps d'exécution.

La règle qui en découle doit être prise en compte lors de la mise en oeuvre de la procédure de codage des solutions : *Le nombre minimal de cuves permis entre deux séparateurs consécutifs est égal à 2 cuves, en évitant toujours la première et la dernière positions inter-cuves.*

Nous précisons alors la notion de position inter-cuve possible : *Il s'agit de la position d'insertion d'un séparateur qui permet de générer des séquences qui ne peuvent pas être obtenues autrement.*

*** Nombre maximal de séparateurs MNS**

Les analyses effectuées nous permettent de calculer le nombre maximal de séparateurs MNS qui peuvent être insérés à une solution de taille NC où $NC \in [\![2, N]\!]$. En effet, une solution de taille NC possède $(NC - 3)$ positions inter-cuves

possibles pour l'insertion de séparateurs au lieu de $(NC - 1)$ à la base, après l'élimination de la première et de la dernière positions de la solution. En outre, en évitant l'insertion de séparateurs sur des positions inter-cuves successives, il reste $(NC - 3)/2$ positions possibles. $(NC - 3)/2$ prendra une valeur entière si NC est un nombre impair, mais réelle si NC est un nombre pair. Le nombre maximal de séparateurs MNS est donc obtenu par la formule :

$$MNS(NC) = \lceil (NC - 3)/2 \rceil$$

Le symbole « $\lceil \rceil$ » traduit la fonction d'arrondi supérieur. Nous en déduisons qu'une taille NC paire et la taille impaire juste subséquente $(NC + 1)$ ont par conséquent le même nombre maximal de séparateurs MNS. Cela signifie également que l'application de séparateurs n'aura pas de bénéfice sur les solutions de taille deux ou trois cuves ($NC = 2$ ou $NC = 3$), puisque MNS a une valeur nulle. De même, une solution de taille $NC = 4$ a 1 seule position possible pour un séparateur $(MNS = 1)$, tout comme une solution de taille égale à 5 $(NC = 5)$. Par conséquent, les séparateurs ne sont intégrés que sur les solutions dont la taille varie entre 4 et N cuves $(4 \leq NC \leq N)$. De plus, pour un problème à N cuves, le nombre maximal de séparateurs $MNS(N)$ présents dans les solutions, correspond au nombre maximal de séparateurs MNS qui peuvent être insérés sur une solution de taille maximale $NC = N$:

$$MNS(N) = \lceil (N - 3)/2 \rceil$$

Par conséquent, le nombre maximal de séparateurs pour un problème de taille N dépend du nombre N de ressources de traitement de la ligne $(MNS(N))$.

L'algorithme 1 détaille la procédure de décodage d'une solution O. Les données d'entrée sont : NC, NS, O' qui est la solution sans séparateurs équivalente à la solution O, et P qui est un vecteur de taille NS. Il est composé des positions P_s prises par les séparateurs, $s \in 1, \ldots, NS$. En particulier, P_1 est la position occupée par le premier séparateur appliqué à la solution $O, \ldots, P_{NS}$ est la position prise par le dernier séparateur. En sortie, la procédure renvoie l'ensemble Seq des séquences de mouvements cycliques S_h et le nombre de robots H associés à la solution O.

La section suivante détaille le modèle mathématique que nous utilisons pour tester la faisabilité des séquences de mouvements générées vis-à-vis de toutes les contraintes du problème.

2.3 Le modèle mathématique

Les séquences de mouvements S_h générées doivent être évaluées pour vérifier si elles sont faisables ou pas à l'égard de toutes les contraintes du problème. Nous réalisons cette évaluation en utilisant le modèle mathématique proposé par Manier et

Algorithme 1 Procédure de décodage

Entrées : O', NC, NS et P
Sorties : Seq et H
début
  $Seq \leftarrow \emptyset$; $h \leftarrow 1$ // 1er robot à associer à la ligne
  $S_h(1) \leftarrow 1$; $S_h(2) \leftarrow 2$ // 1ère séquence commence par le mouvement en charge $L(1,2)$
  $k \leftarrow 2$ // index de la séquence de mouvements sous construction
  $d \leftarrow 0$ // compteur des mouvements à vide identifiés
  $LMO \leftarrow \{1, 2, 3, \ldots, N\}$ // vecteur contenant tous les i, $i \in [\![1, N]\!]$
  $LMO \leftarrow \{2, 3, \ldots, N\}$ // éliminer la valeur 1 (mouvement $L(1,2)$ déjà identifié)
  tant que $d \leq N$ **faire**
    // chercher la dest. $S_h(k+1)$ du mvt à vide suivant $(S_h(k), S_h(k+1))$:
    $POS \leftarrow$ la position de $S_h(k)$ sur la solution O'
    si *POS est vide* **alors**
      $S_h(k+1) \leftarrow S_h(k)$ // Mouvement à vide fictif
    sinon
      si *POS existe dans P* **alors**
        si $POS = P_1$ **alors**
          $S_h(k+1) \leftarrow O'(1)$ // Après un séparateur, le décodage est cyclique
        sinon
          Récupérer s où $P_s = POS$ // Qd ce n'est pas le 1er séparateur de P_s
          $S_h(k+1) \leftarrow O'(P_{s-1}+1)$ // le décodage est cyclique, nous revenons au premier numéro de cuve après le séparateur précédent P_{s-1}
      sinon
        si $POS = NC$ **alors**
          $S_h(k+1) \leftarrow O'(P_{NS}+1)$ // nous revenons au premier numéro de cuve après le dernier séparateur P_{NS}
        sinon
          $S_h(k+1) \leftarrow O'(POS+1)$
    Eliminer de LMO la valeur de $S_h(k+1)$; $d \leftarrow d+1$
    si $S_h(k+1) = S_h(1)$ **alors**
      $Seq \leftarrow Seq \cup S_h$ // la séquence de mouvements est bouclée
      si $d \leq N$ **alors**
        $h \leftarrow h+1$ // nous passons à un second robot h
        $S_h(1) \leftarrow \min(LMO)$ // cuve de départ du 1er mouvement en charge de la nouvelle séquence du robot h
        $S_h(2) \leftarrow S_h(1)+1$ // cuve de destination de ce 1er mouvement en charge
        $k \leftarrow 2$
    sinon
      // chercher la dest. $S_h(k+2)$ du mvt en charge suivant $(S_h(k+1), S_h(k+2))$:
      si $S_h(k+1) = N$ **alors**
        // la cuve suivante à la cuve N est la cuve $N+1=1$
        $S_h(k+2) \leftarrow 1$ // mouvement en charge résultant $L(N,1)$
      sinon
        $S_h(k+2) \leftarrow S_h(k+1)+1$ // tout mouvement en charge va de la cuve i à la cuve $i+1$
      $k \leftarrow k+2$ // 2 pour le mouvement à vide et le mouvement en charge
  $H \leftarrow h$
  retourner Seq, H

Lamrous [Manier & Lamrous 2008], de type programme linéaire en nombres entiers mixtes (*Mixed Integer Linear Programming* ou *MILP*). Grâce à ce modèle, nous obtenons le triplet de résultats (H, S_h, T), donnant respectivement le nombre de robots de la ligne, les séquences de mouvements associées à ces robots, et enfin le temps de cycle commun correspondant à la réalisation de ces séquences.

Si les séquences de mouvements S_h déduites de la solution O sont faisables, la solution O est aussi dite faisable et l'exécution du modèle mathématique aboutit au calcul du temps de cycle T ($T \neq 0$) qui correspond à la solution O (cette période T étant commune à toutes les séquences de mouvements S_h dans le cycle). Si toutefois les séquences de mouvements S_h ne sont pas faisables, le modèle renvoie une valeur nulle au temps de cycle T ($T = 0$).

Dans la suite, nous définissons d'abord les paramètres du problème.

2.3.1 Notations

2.3.1.1 Les données

— N : nombre de cuves de la ligne ; la cuve 1 est la station de chargement-déchargement et les cuves de 2 à N sont les cuves de trempe ;

Pour $i, j \in \{1, 2, ..., N\}$:

— $E(i, j)$: mouvement à vide de la cuve i à la cuve j ;

— $d_{i,j}$: durée de $E(i, j)$;

— r_i : durée du mouvement en charge $L(i, i+1)$ de la cuve i à la cuve $i+1$. $r_i = d_{i,i+1} + c$, où c est une constante de temps requise par le robot pour soulever un porteur de la cuve i, faire une pause si nécessaire au-dessus de la cuve i et laisser le porteur s'égoutter, se stabiliser à l'arrivée de la cuve $i+1$ et descendre le porteur dans la cuve $i+1$. Les temps $d_{i,j}$ et r_i sont des données constantes ;

— m_i : temps minimal de trempe dans la cuve i ;

— M_i : temps maximal de trempe dans la cuve i ;

— M : un nombre très grand qui représente la valeur de l'infini ($+\infty$) ;

— H : nombre de robots associés à la ligne, déduit à l'issue de la procédure de décodage ;

— h : numéro des robots où $h \in \{1, 2, ..., H\}$. Nous supposons que c'est le robot 1 qui est toujours le responsable du premier mouvement en charge $L(1, 2)$, et donc du mouvement à vide $E(2, j)$;

— z_h : nombre de mouvements à vide réalisés par le robot h, avec :

$$\sum_{h=1}^{H} z_h = N,$$

— $v_{h,u}$: le $u^{\text{ème}}$ mouvement à vide identifié pour le robot h, $u \in \{1, 2, ..., z_h\}$.

2.3.1.2 Les variables

Les variables de décision sont les suivantes. Pour $i, j \in \{1, 2, ..., N\}$, $h \in \{1, 2, ..., H\}$ et $u \in \{1, 2, ..., z_h\}$:

— T : temps de cycle (période) ;

— $t_{i,j}$: date de début du mouvement à vide $E(i, j)$. Pour faciliter la lecture des notations, nous l'écrivons t_i à la place $t_{i,j}$, en faisant référence au mouvement à vide commençant à la cuve i ;

— b_i : variable booléenne, telle que :

$$b_i = \begin{cases} 0 \text{ si un porteur est introduit dans la cuve } i \text{ et en est retiré durant le} \\ \quad \text{même cycle,} \\ 1 \text{ sinon.} \end{cases}$$

— $a_{h,u}$: variable booléenne, telle que :

$$a_{h,u} = \begin{cases} 0 \text{ si les mouvements à vide consécutifs } v_{h,u} \text{ et } v_{h,u+1} \text{ sont effectués} \\ \quad \text{par le robot } h \text{ au cours d'un même cycle,} \\ 1 \text{ sinon.} \end{cases}$$

Notons que $v_{h,z_h+1} = v_{h,1}$.

2.3.2 Formulation mathématique

Le modèle qui évalue la faisabilité des séquences de mouvements S_h des robots et donc les ordonnance pour trouver le temps de cycle T minimal dans le cas de robots multiples, est formulé par Manier et Lamrous [Manier & Lamrous 2008] comme un programme linéaire en nombres entiers mixtes (*MILP*), comme suit :

$$\text{Minimiser} \quad T \tag{2.1}$$

Sous les contraintes :

$$0 \leq t_i \leq T, \quad \forall i \in \{1, 2, \ldots, N\} \tag{2.2}$$

$$t_2 = r_1 \tag{2.3}$$

$$t_1 + m_{N+1} \leq T \tag{2.4}$$

$$m_i \leq T, \qquad \forall i \in \{1, 2, \ldots, N\} \tag{2.5}$$

$$(b_i - 1)M \leq t_i - (t_{i+1} - r_i) \leq b_i M, \qquad \forall i \in \{1, 2, \ldots, N-1\} \tag{2.6}$$

$$(b_N - 1)M \leq t_N - (t_1 - r_N) \leq b_N M \tag{2.7}$$

$$m_i - b_i M \leq t_{i+1} - r_i - t_i \leq M_i + b_i M \qquad \forall i \in \{1, 2, \ldots, N-1\} \tag{2.8}$$

$$m_i + (b_i - 1)M \leq T + t_{i+1} - r_i - t_{i+1} \leq M_i + (1 - b_i)M \qquad \forall i \in \{1, 2, \ldots, N-1\} \tag{2.9}$$

$$m_N - b_N M \leq t_1 - r_N - t_N \leq M_N + b_N M \tag{2.10}$$

$$m_N + (b_N - 1)M \leq T + t_1 - r_N - t_N \leq M_N + (1 - b_N)M \tag{2.11}$$

$$\sum_{u=1}^{z_h} a_{h,u} = 1 \qquad \forall h \in \{1, 2, \ldots, H\} \tag{2.12}$$

$$\text{Si } h = 1, \quad a_{1,u} = 0 \text{ et } a_{1,z_1} = 1 \qquad \forall u \in \{1, 2, \ldots, z_1 - 1\}, \tag{2.13}$$

$$\forall h \in \{1, 2, \ldots, H\}; \quad \forall u \in \{1, 2, \ldots, z_h\} :$$

$$t_j + d_{j,i} \leq t_{i+1} - r_i + a_{h,u} M \tag{2.14}$$

$$t_j + d_{j,i} \leq T + t_{i+1} - r_i + (1 - a_{h,u})M \tag{2.15}$$

$$\text{avec } v_{h,u} = E(j, i), \; v_{h,u+1} = E(i + 1, l) \text{ et } v_{h,z_h+1} = v_{h,1}$$

$$b_i \in \{0, 1\} \qquad \forall i \in \{1, 2, \ldots, N\} \tag{2.16}$$

$$a_{h,u} \in \{0, 1\} \qquad \forall h \in \{1, 2, \ldots, H\}; \forall u \in \{1, 2, \ldots, z_h\} \tag{2.17}$$

La fonction objectif représentée par l'équation (2.1) vise à minimiser le temps de cycle T commun à la réalisation de toutes les séquences de mouvements S_h des H robots identifiés de la ligne.

Les contraintes (2.2) à (2.5) définissent et bornent les variables de décision T et t_i : L'inégalité (2.2) garantit que les variables de décision t_i soient des entiers positifs. En outre, elle assure que chaque mouvement à vide, même fictif, doit être effectué pendant le cycle T. Sans perte de généralité, l'équation (2.3) suppose que chaque cycle 1-périodique commence à la fin de l'opération de chargement d'un porteur dans la cuve 1, donc au début du mouvement en charge $L(1, 2)$. On en déduit que le mouvement à vide issu de la cuve 2 commence à la date t_2 qui est égale au temps de transport du porteur de la cuve 1 à la cuve 2, donc r_1. L'équation (2.4) permet de vérifier que chaque porteur est déchargé avant la fin du cycle T. Notons que nous pouvons avoir $m_{N+1} = 0$, où m_{N+1} est le temps minimal de déchargement effectué dans la cuve 1. L'inégalité (2.5) donne une borne inférieure du temps de cycle T. Elle garantit également que le nombre de porteurs traités en même temps dans une cuve ne dépasse jamais la capacité de celle-ci ; plus généralement, si chaque cuve représente une ressource disjonctive (ce qui est le cas dans notre étude), cette contrainte devra être exprimée sous la forme $\max(m_i) \leq T, \forall i \in \{1, 2, ..., N\}$.

Les contraintes (2.6) et (2.7) définissent les variables booléennes b_i. En effet, selon la définition donnée dans la notation (paragraphe 2.3.1.2), $b_i = 0$ si $t_i \leq t_{i+1} - r_i$. Ainsi, dans la cuve i, le même porteur est d'abord chargé puis déchargé pendant une même période T. Sinon, $b_i = 1$ si $t_i \geq t_{i+1} - r_i$: au cours d'une période T, un porteur est déchargé de la cuve i avant qu'un autre porteur n'y soit déposé.

Les équations (2.8) à (2.11) traduisent les contraintes de fenêtres de temps (time window constraints, voir paragraphe 2.1.3). Elles vérifient donc le respect des limites

des temps de trempe m_i et M_i, le temps réel de traitement étant initialement défini par l'expression non linéaire : $b_i.T + t_{i+1} - r_i - t_i$.

Les contraintes (2.12) à (2.15) sont associées aux ressources de transport (les robots) traduisant les contraintes de capacité et les contraintes de disponibilité mentionnées dans le paragraphe 2.1.3. Comme les mouvements des robots ne sont encore pas synchronisés, nous ne connaissons pas a priori leurs premiers mouvements respectifs (au début d'un cycle), hormis pour le robot numéro 1 qui commence toujours par le mouvement en charge $L(1, 2)$. Ceci explique l'introduction des variables booléennes $a_{h,u}$. L'affectation de toutes les variables booléennes dépend de la synchronisation des mouvements des robots durant le cycle. Ensuite, elles doivent être déterminées dans le programme linéaire comme les autres variables. Par exemple, dans la liste $\{v_{h,u}\}_{u=1 \text{ à } z_h}$ associée au robot h, tout couple de deux mouvements à vide consécutifs se présente sous la forme $(E(j, i), E(i + 1, k))$. Si $E(j, i)$ est effectué avant $E(i + 1, k)$ dans la même période du cycle, alors $a_{h,u} = 0$ et le temps entre le début de ces deux mouvements doit être au maximum égal à la durée $d_{j,i} + r_i$ (contrainte (2.14)). Sinon, $a_{h,u} = 1 : E(j, i)$ et $E(i + 1, k)$ sont exécutés successivement par le robot h, mais dans deux périodes consécutives (contrainte (2.15)). Notons que pour le robot 1 dont on connaît le mouvement de départ au cours d'un cycle, les variables booléennes $a_{1,u}$ sont a priori affectées (contrainte (2.13)), et nous avons $v_{1,z_1} = E(j, 1)$ et $v_{1,1} = E(2, k)$.

Enfin, les équations (2.16) et (2.17) vérifient les intervalles de définition des variables binaires b_i et $a_{h,u}$.

Par ailleurs, comme évoqué dans le paragraphe 2.1.3 via les contraintes de non-collision dans le cas des lignes à plusieurs robots, il est de toute évidence nécessaire d'étudier les situations conflictuelles qui peuvent éventuellement se produire entre les robots qui partagent la même voie de déplacement. Néanmoins, dans la littérature des problèmes d'ordonnancement des ateliers avec ressources de transport (les systèmes de fabrication flexibles avec AGVs, les cellules robotisées avec robots et les installations de traitement de surface avec robots), nous pouvons constater que de nombreux chercheurs se sont concentrées principalement sur les problèmes d'affectation et d'ordonnancement des machines et des ressources de transport, sans tenir compte des risques de collision, à l'instar des travaux [Bilge & Ulusoy 1995], [Riera & Yorke-S. 2002], [Liu & Jiang 2005], [Manier & Lamrous 2008], [Deroussi & Norre 2010], [Zhang *et al.* 2012], [Lacomme *et al.* 2013], [Zhang *et al.* 2014].

Notre objectif étant dans un premier temps de valider la nouvelle approche du codage-décodage avec séparateurs, nous employons ce modèle mathématique proposé par Manier et Lamrous dans [Manier & Lamrous 2008] dans lequel ces contraintes d'évitement de collision sont relaxées, ce qui nous permet de comparer nos résultats avec ceux de ces auteurs. Dans un second temps, avec l'optique d'étudier ces situations conflictuelles entre les robots, nous les intégrons dans la procédure d'évaluation

pour prendre en compte toutes les contraintes possibles du problème multi-robots. Cette partie de notre travail fait l'objet du chapitre 4 de ce manuscrit.

Nous consacrons la section suivante à la présentation de l'algorithme général de résolution du CHDSP basé sur l'algorithme génétique et nous en développons les procédures adoptées.

2.4 Algorithme génétique bi-objectif pour le CHDSP

2.4.1 Concepts et structure générale

Pour résoudre le CHDSP, et compte tenu de la complexité de ce problème, nous avons choisi d'utiliser une méthode approchée, de type métaheuristique. Avec les métaheuristiques, on ne peut pas garantir l'optimalité des solutions trouvées. Cependant, elles demeurent capables de donner des solutions de bonne qualité (proches de la solution optimale et parfois même les solutions optimales si nous les connaissons a priori). Les temps de calcul sont raisonnables comparés aux méthodes exactes pour les problèmes combinatoires. Parmi cette classe de méthode, nous avons choisi d'adopter un algorithme génétique afin de comparer nos résultats à ceux de Manier et Lamrous [Manier & Lamrous 2008]. Ces chercheurs ont en effet développé un algorithme génétique pour la résolution du CHDSP, en adoptant l'approche de codage initialement décrite à base des mouvements à vide et sans séparateur. Dans un premier travail d'élaboration d'une méthode de résolution du CHDSP, et pour évaluer au mieux l'efficacité de notre approche d'amélioration du codage à l'aide des séparateurs, nous avons décidé de conserver la même base de travaux que ces auteurs, afin d'estimer l'apport de l'intégration des séparateurs sur la représentation de l'espace de recherche des solutions.

L'algorithme génétique appartient à la catégorie des méthodes approchées de la famille des métaheuristiques dites évolutionnaires [Coello 2005]. Cette dernière regroupe les méthodes à base de population de solutions [Coello 2005], telles que l'algorithme génétique [Holland 1975, Holland *et al.* 1975, Goldberg & Holland 1988, Goldberg 1989, Davis 1991, Holland *et al.* 1992, Holland 1992, Mitchell 1998, Sastry *et al.* 2005, Sivanandam & Deepa 2008], l'optimisation par colonies de fourmis [Colorni *et al.* 1991, Dorigo 1992, Dorigo & Blum 2005, Dorigo *et al.* 2006, Dorigo & Stützle 2019], l'optimisation par essaims particulaires [Kennedy & Eberhart 1995, Eberhart & Kennedy 1995, Poli *et al.* 2007, Clerc 2010], etc. L'algorithme génétique est une méthode inspirée du principe darwiniste de la théorie de l'évolution. Il s'agit d'un algorithme stochastique basé sur l'évolution d'une population de solutions appelées individus. La méthode effectue une recherche aléatoire par croisements et par mutations entre les individus de la population. Nous décrivons ici l'algorithme que nous avons élaboré pour le CHDSP, et que nous notons SGA. Il s'agit d'une adaptation de celui développé par [Manier & Lamrous 2008] que nous appelons GA.

Un algorithme génétique réalise une optimisation dans un espace de données tributaire de la population initiale générée. Dans l'algorithme GA, la fonction d'adaptation ou fitness permettant d'aller vers des solutions de meilleure qualité a été définie par une fonction d'agrégation tenant compte des deux critères à minimiser : le nombre de robots et le temps de cycle. Toutefois, cette approche, quoi qu'apparemment simple, présente l'inconvénient du choix de la pondération des critères, rendu d'autant plus difficile que les ordres de grandeur de ces critères sont complètement différents. De plus, cette méthode biaise l'exploration de sous espaces de recherche. Ceci a été remarqué en observant le paysage des solutions, malgré les résultats appréciables obtenus selon les instances.

Contrairement à GA, l'approche adoptée dans SGA est multi-objectifs, et cherche, non pas à faire converger une population vers un point optimum, mais vers un ensemble de solutions de compromis qui ne sont dominées par aucune autre solution. C'est ce qu'on appelle un front de Pareto ([Deb *et al.* 2002]), que nous cherchons à déterminer pour notre problème. Schématiquement, le front de Pareto est l'ensemble des solutions de compromis. Sur la figure 2.13, A et B sont deux points de ce front : A ne domine pas B, B ne domine pas A, mais tous les deux dominent le point C. La résolution de notre problème bi-objectif permet deux phases. La première est de produire des solutions de meilleur compromis, c'est la phase d'optimisation. La deuxième concerne le choix de la solution. Ce choix relève d'un décideur qui, parmi l'ensemble des solutions de compromis, doit extraire celle qu'il utilisera. On parle de système d'aide à la décision. Nos travaux s'intéressent à la première phase.

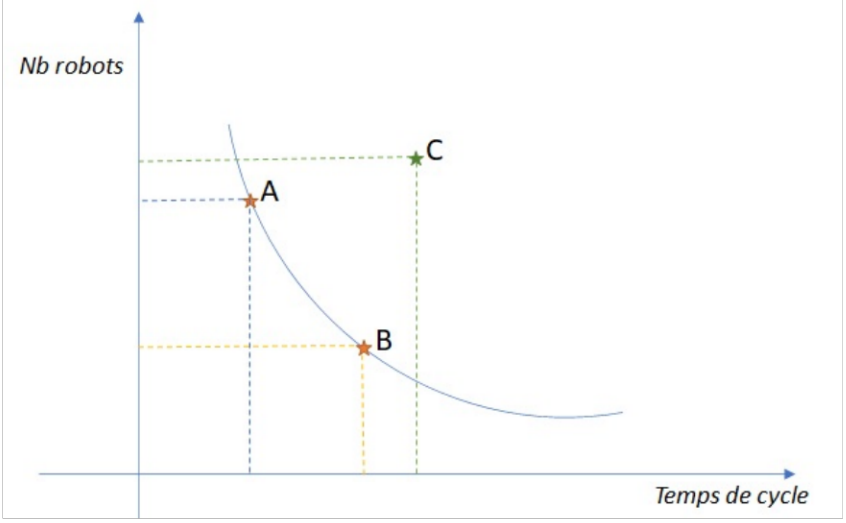

FIGURE 2.13 – Exemple de front de Pareto.

La figure 2.14 illustre l'approche générale de résolution et les principales étapes de l'algorithme SGA, qui sont détaillées dans la suite. Les données relatives au déroulé de l'algorithme suit le bouclage classique connu des algorithmes génétiques

avec les paramètres suivants :

- la taille de la population initiale : 10 fois le nombre de cuves ;
- Sous-population élitiste : 5% des meilleures solutions du front de Pareto courant sont conservées dans la population suivante ;
- le type de sélection : sélection par tournoi ;
- le taux de croisement : 0.8 ;
- le taux de mutation : 0.2 ;
- le critère d'arrêt : le nombre de générations égal à 20 fois le nombre de cuves OU nombre de générations successives sans évolution du front de Pareto est égal à 50.

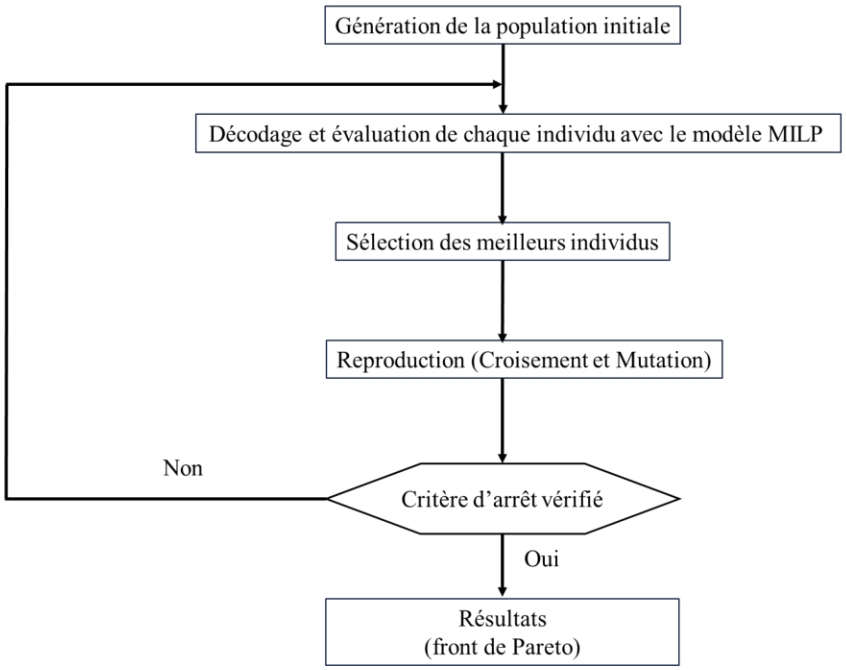

FIGURE 2.14 – Fonctionnement général de l'algorithme génétique SGA.

2.4.2 Génération de la population initiale

La population initiale est générée aléatoirement avec un nombre de séparateurs possible selon la taille de l'individu. Dans cette population, nous cherchons en particulier, pendant un nombre raisonnable d'itérations, à intégrer au moins une solution faisable à un robot. En effet, ce type de configurations intéressantes peut s'avérer rare selon les instances, et l'amélioration du front de Pareto doit fonctionner globalement sur toutes les configurations, mono et multi-robots. Notons que si une solution est non faisable, l'évaluation par le MILP renvoie une valeur à 0 pour le temps de cycle. Comme nous cherchons à minimiser la période, nous remplaçons ce 0 par une grande valeur, pour simuler l'infini.

2.4.2.1 Génération des solutions

Chaque solution est générée en utilisant le codage basé sur les mouvements à vide avec séparateurs que nous avons proposé. Comme défini auparavant, le nombre de cuves considérées NC apparaissant dans la solution codée, de même que le nombre de séparateurs appliqués NS, sont tous deux variables. Les solutions ont alors des tailles variables. La génération d'une solution O se fait de manière stochastique, selon la démarche suivante :

(1) choix aléatoire de la taille NC dans l'intervalle $[\![2, N]\!]$, où N est le nombre de cuves ;

(2) construction progressive en choisissant aléatoirement NC numéros de cuves i dans l'intervalle $[\![1, N]\!]$, chaque numéro ne pouvant être sélectionné qu'au plus une fois ;

(3) choix aléatoire d'un nombre NS de séparateurs dans l'intervalle $[\![0, MNS(N)]\!]$;

(4) intégration à la solution O de ces NS séparateurs (NS chiffres 0) sur des positions inter-cuves choisies aléatoirement.

Dans la solution O, de taille $T_{solution} = NC + NS$, les positions inter-cuves sur lesquelles les séparateurs sont placés ne sont pas toujours des positions possibles au sens des règles énoncées précédemment. Ainsi, nous développons une procédure de réparation des séparateurs qui prend en compte les positions interdites.

2.4.2.2 Procédure de réparation des séparateurs

Avant de passer au décodage de la solution O, il est nécessaire de la réparer en respectant les positions inter-cuves autorisées. La procédure de réparation des séparateurs est donc appliquée chaque fois à l'issue des procédures de génération ou de modification de la solution O et avant les procédures de décodage et d'évaluation. Elle fonctionne comme suit :

(1) Elle extrait d'abord de chaque solution O les positions P_s occupées par des zéros (les séparateurs) qu'elle stocke dans le vecteur P ;

(2) Elle traduit ensuite ces positions en positions inter-cuves de la solution O sans séparateur, avec la formule suivante : $P_s = P_s - 1 - (s-1)$ avec $s = 1, \ldots, NS$;

(3) Elle répare le vecteur P : les positions jugées interdites sont annulées selon les conditions de permission ou d'évitement définies :

- **(Condition 1)** Rejeter les première et dernière positions inter-cuves ($P_s = 1$ et $P_s = NC - 1$) ainsi que les positions extrêmes ($P_s = 0$ et $P_s = NC$). Les positions finales doivent respecter la condition $2 \leq P_s \leq NC - 2$;
- **(Condition 2)** Éliminer les répétitions ($P_s = P_s + 1$). Si elles existaient, ces répétitions traduiraient que plus d'un séparateur occupent la même position inter-cuves ;
- **(Condition 3)** Vérifier si les écarts entre deux positions successives de séparateurs sont supérieures ou égales à 2 ($P_s + 1 - P_s \geq 2$). Il faut éviter que

deux séparateurs n'occupent deux positions inter-cuves successives. Si c'est le cas ($P_s + 1 - P_s = 1$), nous gardons la première position et nous supprimons la seconde.

La figure 2.15 donne un exemple d'application de cette procédure, pour un problème de taille égale à 13 ($N = 13$), donc avec un nombre maximal de séparateurs égal à $MNS(13) = \lceil (13-3)/2 \rceil = 5$. Nous considérons une solution initiale (solution O) générée aléatoirement : $\{11, 5, 2, 10, 6, 0, 0, 13, 7, 0, 8, 0, 3, 12, 1, 0, 4\}$. Elle est de taille $T_{solution} = 17$ et contient 5 séparateurs. La solution équivalente sans séparateurs est donnée sur la deuxième ligne de la figure : $\{11, 5, 2, 10, 6, 13, 7, 8, 3, 12, 1, 4\}$. Nous donnons ensuite le vecteur P_s des positions extraites des séparateurs de la solution O ; $P_s = \{6, 7, 10, 12, 16\}$ de taille $NS = 5$. Nous traduisons ces positions en positions inter-cuves où P_s devient $\{5, 5, 7, 8, 11\}$. Enfin nous réparons ce vecteur : la position 5 étant répétée deux fois, nous n'en gardons qu'une seule ; les positions 7 et 8 étant deux positions successives, nous ne conservons que la première ; enfin la position 11 étant la dernière position inter-cuve de la solution O, elle est interdite et donc nous la supprimons. Le vecteur actualisé des positions exactes des séparateurs après réparation est alors $P_s = \{5, 7\}$, dont la taille est $NS = 2$.

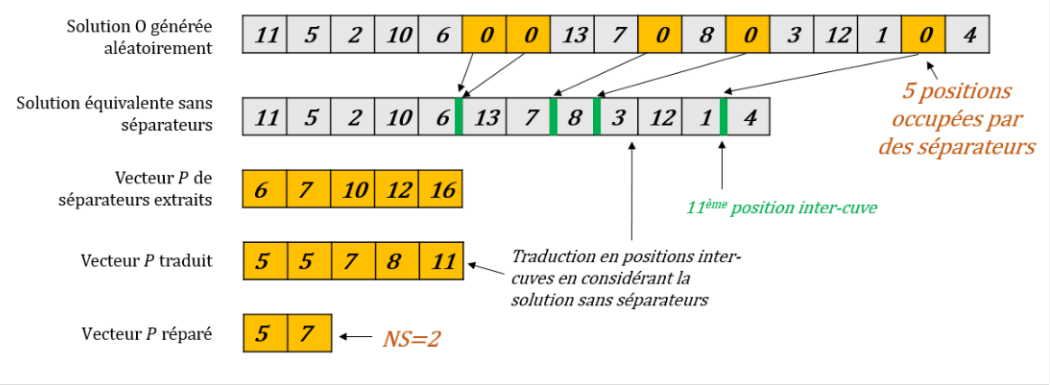

FIGURE 2.15 – Application de la procédure d'identification de séparateurs sur une solution générée aléatoirement dans un problème à 13 cuves.

2.4.3 Procédures de décodage et d'évaluation

La procédure de décodage traite l'aspect conception du CHDSP. Elle vient juste après la procédure de réparation des séparateurs. Elle fonctionne selon l'algorithme de décodage (Algorithme 1) déployé dans la section 2.2.3.4. La procédure fournit les séquences de mouvements cycliques S_h et le nombre de robots H correspondant à chaque individu réparé.

La procédure d'évaluation, quant à elle, traite le sous-problème d'ordonnancement du problème CHDSP. Elle emploie le modèle mathématique MILP qui a été décrit au

paragraphe 2.3. Ce modèle évalue les séquences cycliques S_h déduites de la procédure de décodage, pour tester si elles respectent toutes les contraintes considérées. Il calcule ainsi le temps de cycle minimal résultant T, en synchronisant en parallèle les H séquences de mouvements des H robots. Cette procédure fait appel aux données du problème $m_i, M_i, d_{i,j}$ et r_i, et fournit en sortie les variables de décision $t_{i,j}, T, b_i$ et $a_{h,u}$.

2.4.4 Croisements et mutations

Pour former la population à une itération donnée, nous appliquons une stratégie élitiste qui consiste à ne conserver que 5% des meilleures solutions du front de Pareto courant. Pour compléter les 95% restants de la nouvelle population, nous appliquons des opérateurs de croisement et de mutation aux individus de la population courante, préalablement sélectionnés par tournoi. Nous appliquons un tournoi binaire qui consiste à choisir aléatoirement deux individus distincts et de conserver le mieux adapté, c'est à dire celui ayant le rang le plus faible au sens de NSGA-II (Non-dominated Sorting Genetic Algorithm) [1]. La phase de sélection interdit de choisir un individu plusieurs fois.

Les opérateurs de croisement classiques prennent en entrée deux parents et renvoient deux enfants combinés en choisissant aléatoirement un ou plusieurs points de croisement. Nous opérons des croisements en un point aléatoire comme le montre l'exemple de la figure 2.16. Lors de la transcription d'un descendant tout gène déjà copié est ignoré pour éviter les doublons.

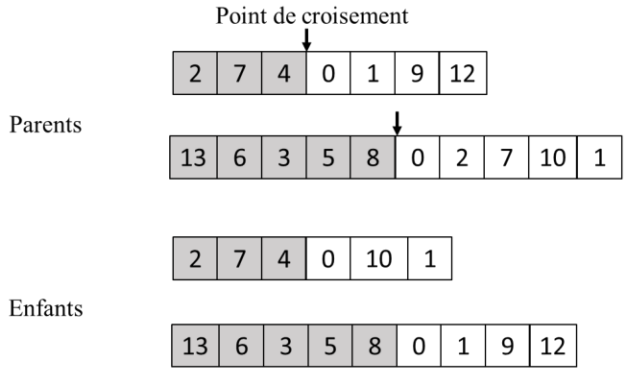

FIGURE 2.16 – Opérateur de croisement.

Les opérateurs de mutation permettent une modification aléatoire d'individus choisis aléatoirement dans la population. Plusieurs manières de muter un chromosome sont possibles [Deb *et al.* 2002] : par modification d'un ou plusieurs gènes, par permutation, etc. La mutation que nous appliquons est adaptée de celle utilisée par [Manier & Lamrous 2008], et permet :

1. *Ortiz, Gilberto A. "Multi-objective optimization using ES as Evolutionary Algorithm". Mathworks. Retrieved 1 November 2012*

— de supprimer un gène correspondant à un numéro de cuve ou un séparateur choisi aléatoirement ;

— ou d'ajouter aléatoirement un numéro de cuve qui est absent ou un séparateur ;

— ou de permuter deux gènes choisis aléatoirement, quelle que soit leur nature (séparateur ou numéro de cuve).

Ces trois opérateurs de mutation sont appliqués avec une probabilité égale. Le fait d'agir sur les numéros de cuve et les séparateurs permet une bonne diversification de la population.

A chaque génération d'un enfant par croisement ou mutation, la procédure de réparation est appliquée si nécessaire.

2.5 Expérimentations et résultats

2.5.1 Instances

Nous avons testé l'algorithme SGA sur trois instances industrielles de la littérature. La première est celle proposée par Phillips et Unger dans [Phillips & Unger 1976] que nous notons *P&U*. La deuxième et la troisième sont deux instances proposées par Manier dans sa thèse [Manier 1994], nommés *Ligne1* et *Ligne2*. Les instances *P&U* et *Ligne1* comprennent 13 cuves alors que *Ligne2* en comprend 15. Ces instances rassemblent diverses caractéristiques spécifiques qui les rendent assez représentatives des installations réelles. Une description plus complète se trouve dans les références mentionnées ci-dessus et les données qui leurs sont associées sont fournies en Annexe (sections A.1, A.2 et A.3). Tous les temps définis sont exprimés en secondes.

L'exemple du problème *Ligne1* inclut une contrainte de circulation qui a été expliquée par Shapiro [Shapiro & Nuttle 1988] : Le même porteur de pièces, déchargé dans un cycle, est chargé à nouveau pour entrer dans la ligne au cycle suivant. Comme la formulation MILP utilisée ici pour évaluer les solutions ne prend pas en compte cette contrainte, nous avons été obligés de modifier légèrement les données, fournies également dans Manier et Lamrous [Manier & Lamrous 2008] : au lieu de m_i(cuve 1, opération de chargement)= 180s et m_i(cuve 1, opération de déchargement)= 180s, nous avons mis m_i(cuve 1, opération de chargement)= $180 + 180 = 360s$ comme somme des deux temps minimaux et m_i (cuve 1, opération de déchargement)= 0s, pour obliger de considérer que les deux temps minimaux correspondent au même porteur de pièces.

2.5.2 Environnement de test

Les expérimentations ont été réalisées sur le logiciel *Matlab*, sur un processeur Intel®Core™i7-3770 à 3,40 GHz, 8 Go de RAM, 64 bits et sous un système d'exploi-

TABLE 2.1 – Résultats pour l'instance *P&U*.

Benchmark	*P&U*				
Nombre de robots H	T_{opt}	GA		SGA	
		T_{best}	T_{moy}	T_{best}	T_{moy}
1	521	665	673.8	521	1024
2	251	332	346	264	360
3	_	196	205.8	192.33	231.5
4	_	210	226.4	163	207
5	_	151	159.4	151.33	154.14
6	_	_	_	151	_
7	_	151	151	151	151

tation Windows. Le MILP a été résolu grâce à *Hybrid Toolbox*[2] qui est une boîte à outils de MATLAB/Simulink, utilisée pour la modélisation, la simulation et la vérification de systèmes dynamiques hybrides.

Le temps d'exécution moyen de SGA pour les trois instances testées est de 145 min. Comparé à l'algorithme GA [Manier & Lamrous 2008] avec les mêmes conditions matérielles, les tailles de population et les conditions d'arrêt, le temps d'exécution est presque divisé par deux. Cette réduction s'explique par la résolution du problème avec un algorithme bi-objectif plutôt qu'en considérant une fonction d'agrégation. Un front de Pareto contenant des solutions est atteint plus rapidement avec SGA qu'avec GA. Le temps de calcul de la phase d'initialisation représente 70 % de ce temps. Cela s'explique par la difficulté de trouver une solution faisable à un robot ($H = 1$).

2.5.3 Résultats

Dans les tableaux 2.1 et 2.2, nous fournissons les résultats que nous avons obtenus en testant SGA sur ces instances, tout en affichant aussi les résultats de l'algorithme GA. Nous déployons ces résultats en termes des meilleurs temps de cycle (notés T_{best}) et des temps de cycle moyens sur 20 exécutions (notés T_{moy}), pour chaque nombre de robots $H = 1 \ldots 7$. Nous donnons également les temps de cycle optimaux trouvés dans la littérature via des méthodes exactes, pour un seul robot ou deux robots, que nous notons T_{opt}.

Pour les 3 instances, et sauf pour l'instance *Ligne1* avec un robot, SGA retrouve ou améliore les meilleurs temps de cycle de GA (T_{best}). Deux explications à cela : la première est que notre codage implémenté dans SGA permet d'obtenir les solutions

2. *A. Bemporad. "Hybrid Toolbox" - User's Guide, 2004. http ://cse.lab.imtlucca.it/ bemporad/hybrid/toolbox.*

TABLE 2.2 – Résultats pour les instances *Ligne1* et *Ligne2*.

Benchmark	*Ligne1*					*Ligne2*				
Nombre de robots H	T_{opt}	GA		SGA		T_{opt}	GA		SGA	
		T_{best}	T_{moy}	T_{best}	T_{moy}		T_{best}	T_{moy}	T_{best}	T_{moy}
1	425	452	677	492	1032	712	858	1294	712	1332
2	_	361	402	361	404	_	661	662	661	662
3	_	361	361	361	361	_	661	661	661	661
4	_	361	361	361	361	_	661	661	661	661
5	_	361	361	361	361	_	661	661	661	661
6	_	361	361	361	361	_	661	661	661	661
7	_	361	361	361	361	_	678	678	661	661

optimales qui ne peuvent être atteintes avec le codage initial implémenté dans GA ; la seconde explication réside dans la fonction objectif des deux algorithmes qui diffèrent, car la méthode multi-objectifs de SGA surpasse la fonction d'agrégation de GA ramenant le problème à un cas mono-objectif. Pour la solution à un robot de *P&U* ($T = 521$), nous sommes certains que la première explication prévaut puisque nous avons déjà montré que la solution optimale de cette instance ne peut pas être représentée par le codage initial sans séparateurs. De ce fait, elle ne peut pas être trouvée par GA, alors qu'elle le peut avec SGA.

L'algorithme SGA permet d'atteindre la solution optimale à un robot pour deux instances sur les trois, et il approche de la solution optimale pour la troisième instance *Ligne1* avec un écart relatif de 8.85 % par rapport à GA et de 15.67% par rapport à l'optimal. Ce résultat est notable car les solutions faisables à un robot (et donc a fortiori les solutions optimales à un robot) sont les plus difficiles à trouver.

Nous observons par ailleurs que, pour ces exemples à 13 ou 15 cuves, il n'est plus possible d'améliorer le temps de cycle au-delà de 5 robots pour *P&U* et 2 robots pour *Ligne1* et *Ligne2*. En effet, à partir d'un certain nombre de robots, la productivité de la ligne va stagner, c'est-à-dire, elle ne va plus progresser même si le nombre de ces ressources de transport augmente. Dans ce cas, ce sont les cuves qui deviennent les ressources critiques de la ligne et non plus les robots. De plus, comme ces dernières influencent le temps de cycle final avec leurs limites temporelles des opérations de trempe, la cuve critique menante sera celle qui a le temps minimal de trempe le plus grand (le plus grand m_i), en d'autres termes, la plus longue durée minimale de trempe. Cette cuve est considérée ainsi comme un *goulot d'étranglement*. Les temps de cycle ainsi obtenus (151, 361 et 661) correspondent respectivement, à une unité près, à la plus grande durée minimale de trempe de chacune des trois instances ($T = \max(m_i)$, par exemple $\max(m_i) = 150$ pour *P&U*).

En synthèse, notre SGA améliore globalement l'approche GA précédente, car elle obtient le plus souvent de meilleures valeurs de temps de cycle. De plus, SGA permet d'obtenir des solutions optimales dans certains cas. Cependant, un inconvénient de notre algorithme SGA est la robustesse de la méthode, car nous obtenons un temps de cycle moyen plus important qu'avec GA (plus de variations entre les réplications). Ce point devrait être amélioré.

2.6 Conclusion

Dans ce chapitre, nous avons défini la variante de problème d'atelier que nous étudions. Elle intègre conjointement le dimensionnement des ressources de manutention et l'ordonnancement des opérations de transport. Cette variante bi-objectif du CHSP est appelée : *Cyclic Hoist Design and Scheduling Problem* (CHDSP). Nous avons amélioré un codage existant basé sur les mouvements à vide des robots, dont l'originalité et l'avantage est de permettre de représenter dans un même formalisme des configurations de lignes à un ou plusieurs robots, et donc de résoudre des problèmes de type CHSP pour différents nombres de robots. Après avoir montré l'intérêt des améliorations apportées dans l'exploration de l'espace de recherche, nous avons exploité ce codage dans l'élaboration d'un algorithme génétique bi-objectif, hybridé avec un programme linéaire pour la phase d'évaluation des solutions générées. Les résultats obtenus avec ce premier algorithme sont encourageants. Toutefois, nous avons également développé une seconde méthode approchée, que nous présentons dans le chapitre suivant.

Chapitre 3

CHDSP : Résolution par recherche à voisinage variable

Ce chapitre présente la seconde méthode de résolution que nous avons développée pour le Cyclic Hoist Design and Scheduling Problem. Il s'agit à nouveau d'une méthaheuristique, mais cette fois basée sur une Recherche à Voisinage Variable pour l'exploration de l'espace des solutions, et combinée à un programme linéaire en nombres entiers mixtes pour la phase d'évaluation. Nous lui intégrons par ailleurs une procédure de retour en arrière pour en améliorer les performances. Toute l'approche est validée par des tests effectués sur les instances les plus connues. Son efficacité et sa robustesse sont confrontées à celles de notre algorithme génétique proposé dans le chapitre précédent, aussi bien qu'aux résultats de la littérature.

Sommaire

Publication

Laajili E., Lamrous S., Manier M.-A. and Nicod J.-M., "An Adapted Variable Neighborhood Search based algorithm for the cyclic multi-hoist design and scheduling problem", accepté dans *Computers and Industrial Engineering.*

3.1 Motivation

Dans une première approche, nous avons développé un algorithme génétique pour résoudre le CHDSP. Cela nous a permis de confirmer la difficulté de trouver des solutions faisables dans l'espace des solutions. Dans ce chapitre, nous avons un nouvel objectif qui est d'offrir une nouvelle approche pour le CHDSP, qui sera plus performante et robuste que la *SGA* dans la résolution de ce problème bi-objectif complexe. Cette méthode reste dans la catégorie des méthodes approchées de type *métaheuristique*, qui représentent des algorithmes de résolution efficaces ayant un potentiel réel dans la résolution rapide des problèmes complexes comme le nôtre, comparés aux méthodes exactes. Parmi ces algorithmes, nous avons choisi la *Recherche à Voisinage Variable* ou *RVV* (ou *Variable Neighborhood Search* ou *VNS*).

3.2 Méthode de Recherche à Voisinage Variable

La *RVV* ou *VNS* appartient à la catégorie des méthodes dites *de recherche locale* [Aarts *et al.* 2003], à la différence de l'algorithme génétique qui est une méthode *évolutionnaire*. Cette catégorie regroupe les méthodes à solution unique, telles que la méthode de descente (ou *Hill Climbing*) [Johnson *et al.* 1988, Bräysy & Gendreau 2005, Selman & Gomes 2006, Tsamardinos *et al.* 2006], la recherche tabou [Glover 1989, Glover 1990b, Glover 1990a, Glover & Laguna 1998, Gendreau & Potvin 2005], le recuit simulé [Kirkpatrick *et al.* 1983, Van Laarhoven & Aarts 1987, Bertsimas *et al.* 1993, Ingber 1993, Aarts *et al.* 2005, Dowsland & Thompson 2012].

La méthode d'optimisation *VNS* a été introduite en 1997 par Mladenovic et Hansen [Mladenović & Hansen 1997] comme une métaheuristique qui procède à un changement systématique de voisinage dans le cadre d'une recherche locale. Cela fait d'elle une métaheuristique simple et efficace pour les problèmes d'optimisation combinatoire. Selon ses auteurs, la VNS, contrairement à la plupart des autres méthodes de recherche locale, ne suit pas une trajectoire, mais explore de plus en plus des voisinages éloignés de la solution courante, et en effectuant un saut vers une nouvelle solution si et seulement si une amélioration de l'objectif est apportée.

Nous avons choisi la méthode *VNS* pour ses caractéristiques structurelles d'exploration de l'espace de recherche qui influencent de manière cruciale ses performances, telles que les structures de voisinage, la méthode d'exploration, les stratégies d'intensification et/ou de diversification, la condition d'acceptation d'une nouvelle solution voisine. Une meilleure compréhension et manipulation de ces aspects conduit à une meilleure conception de l'algorithme et à son adaptation pour résoudre efficacement les problèmes les plus difficiles. Ainsi, en raison de la complexité du problème CHDSP que nous résolvons et de l'importance de l'espace de recherche admis, nous n'avons pas appliqué la méthode *VNS* dans sa version standard décrite par [Mladenović & Hansen 1997], mais nous l'avons adaptée à notre problème. Nous appelons notre

approche générale de résolution basée sur cette méthode, la *Recherche à Voisinage Variable adaptée* ou *adapted Variable Neighborhood Search* (noté *aVNS*).

Dans la suite, nous présentons l'approche de résolution *aVNS*. Nous donnons ses étapes et son principe de fonctionnement et nous développons les procédures adoptées dans le cadre de son utilisation pour notre problème.

3.3 Approche de résolution aVNS

Cette deuxième approche de résolution que nous proposons reste très similaire à l'approche *SGA* dans le sens où les deux font appel aux mêmes opérations de codage et de décodage à base de mouvements à vide et les deux utilisent le même modèle mathématique pour l'évaluation des solutions (figure 3.1). Néanmoins, leur différence réside dans le principe de chaque méthode à optimiser la recherche des solutions faisables par nombre de robots.

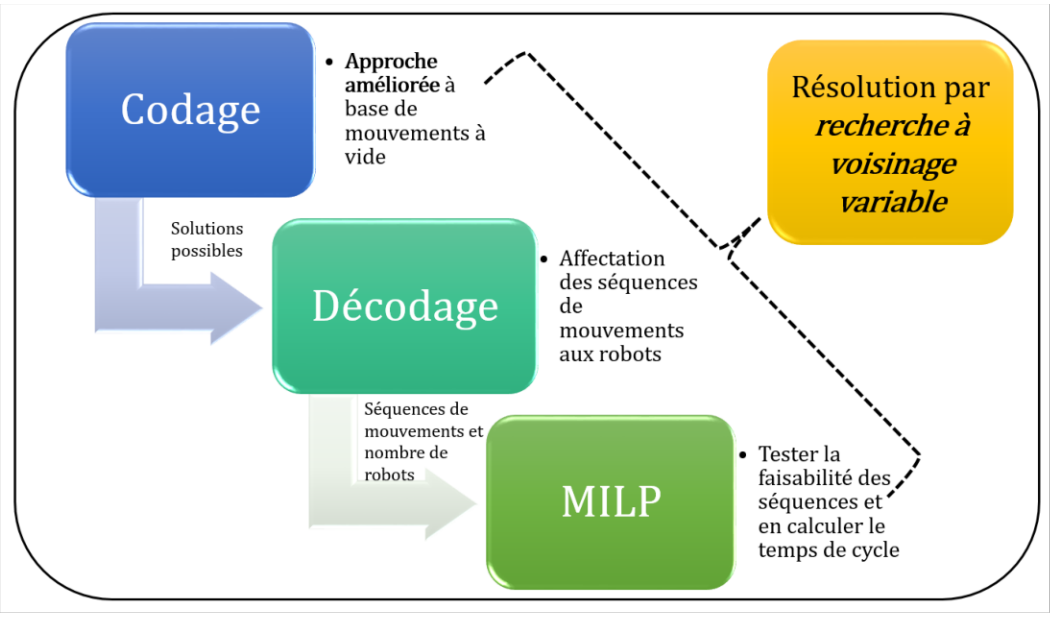

FIGURE 3.1 – Approche de résolution avec la Recherche à Voisinage Variable.

L'aVNS que nous développons est appliqué pour résoudre le CHDSP bi-objectif, c'est à dire pour minimiser à la fois le temps de cycle T et le nombre de robots H. Comme de chaque solution encodée O, nous identifions un nombre de séquences de mouvements cycliques S_h correspondant à un nombre de robots H et à un temps de cycle minimal associé T. Ainsi nous désignons une solution au CHDSP à travers le triplet (H, S_h, T), (H, O, T) ou plus simplement par le couple (H, T).

Nous concevons l'aVNS pour générer des voisinages riches en solutions et pour y rechercher efficacement les meilleures solutions. Ainsi, nous proposons d'essayer

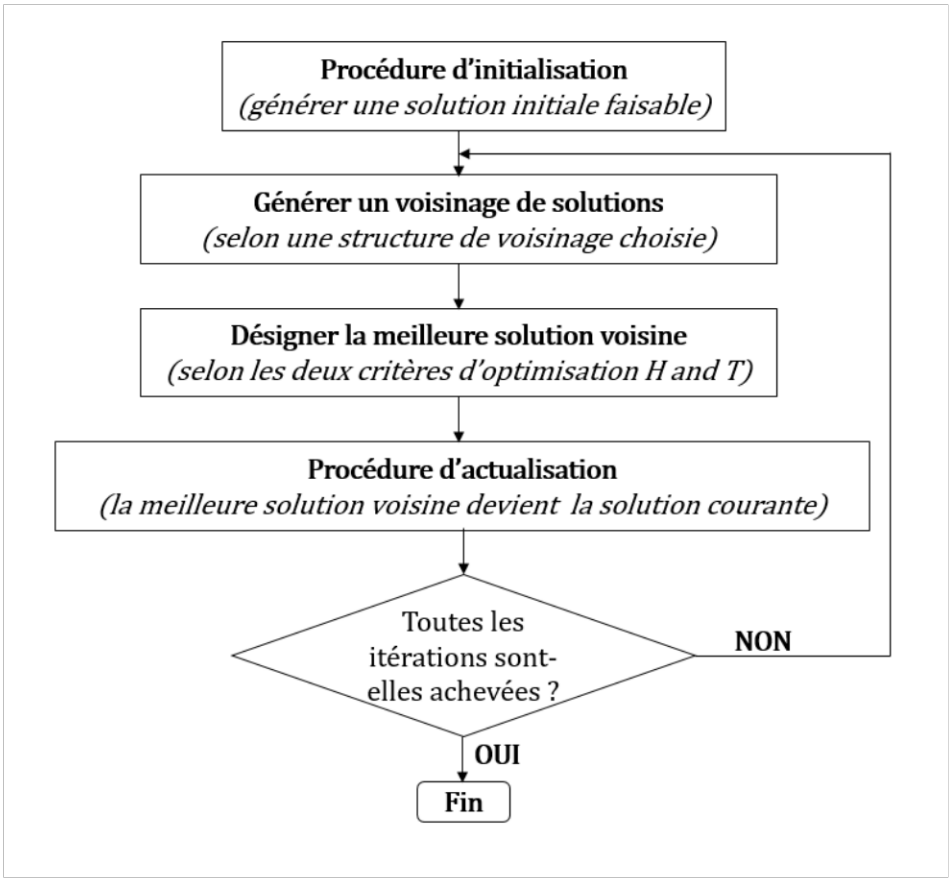

FIGURE 3.2 – Schéma général de l'aVNS.

plusieurs structures de voisinage différentes pour accentuer la diversification des solutions visitées et l'exploration de différents voisinages. L'algorithme général de résolution aVNS est schématisé par la figure 3.2. Nous décrivons dans la suite ses différentes étapes. Les notations, les hypothèses et les contraintes préalablement établies dans le chapitre 2 restent valables dans ce chapitre. Nous ajoutons à cela des nouvelles notations qui sont utiles pour la description des étapes de l'algorithme *aVNS* et qui sont utilisées dans la suite de ce chapitre.

3.3.1 Notations

— $ITER$: nombre maximal d'itérations de l'algorithme aVNS ;

— $iter$: indice d'une itération, $iter = 1, 2, \ldots, ITER$;

— O_1 : solution initiale ;

— $(S_h)_1$: séquences de mouvements identifiées à partir de O_1 ;

— H_1 : nombre de robots identifié à partir de O_1 ;

— T_1 : temps de cycle correspondant à O_1 ;

— O_{cour} : solution courante de l'algorithme aVNS. A chaque itération $iter$, elle représente la solution origine des solutions voisines générées.

— $O_{best}(H)$: meilleure solution pour le nombre de robots H ;

— $T_{best}(H)$: meilleur temps de cycle pour le nombre de robots H ;

— $Size$: taille d'un voisinage. Il s'agit du nombre de solutions voisines à générer à chaque itération ;

— S : indice des solutions voisines, $S = 1, \ldots, Size$;

— N_{type} : structures de voisinage présélectionnées, utilisées pour générer les solutions voisines, $type$ est le nom donné à la méthode de transformation ou de perturbation adoptée pour créer la solution voisine ;

— $N_{type}(O_{cour})$: ensemble des solutions voisines obtenues avec la structure de voisinage N_{type}, à partir de la solution courante O_{cour} ;

— O_{vois} : une solution voisine ;

— $(S_h)_{vois}$: séquences de mouvements identifiées à partir de O_{vois} ;

— H_{vois} : nombre de robots pour O_{vois} ;

— T_{vois} : temps de cycle correspondant à O_{vois} ;

— $(O_{vois})_{best}$: meilleure solution du voisinage ;

— $(H_{vois})_{best}$: nombre de robots de la meilleure solution voisine ;

— $(T_{vois})_{best}$: temps de cycle de la meilleure solution voisine ;

3.3.2 Procédure d'initialisation

L'algorithme aVNS commence par initialiser les variables $iter$, $O_{best}(H)$ et $T_{best}(H)$. Comme nous résolvons le CHDSP pour un nombre de robots H variable et nous ne connaissons pas au préalable le nombre H maximal que l'aVNS atteint, nous initialisons $O_{best}(H)$ et $T_{best}(H)$ pour $H = i = 1, \ldots, N$. Précisément, $O_{best}(H)$ est définie comme étant une matrice de taille $N \times N$ et $T_{best}(H)$ comme un vecteur de taille N. Toutes les solutions $O_{best}(i)$ sont initialisées comme des vecteurs vides de taille N et tous les $T_{best}(H)$ ont une valeur initiale égale à 2000 qui représente une borne supérieure de l'objectif T. Cette limite supérieure a été choisie au hasard, mais assez grande pour dépasser les optima en temps de cycle T des solutions à un robot ($H = 1$) trouvées dans la littérature, pour toutes les instances testées.

Nous commençons à $iter = 1$. Nous générons de manière aléatoire la solution initiale O_1, en suivant la démarche de génération décrite au paragraphe 2.4.2.1 du chapitre 2. Cette solution O_1 doit être faisable ($T_1 \neq 0$), et associée à un ou à deux robots ($H_1 = 1$ ou $H_1 = 2$). Puis nous réitérons la procédure jusqu'à ce qu'une solution avec ces caractéristiques soit trouvée. Nous jugeons ce type de solution, faisable et avec un petit nombre de robots, comme étant une bonne solution initiale pour commencer la recherche. Ce choix a été également validé par les tests.

Une fois la solution O_1 générée et validée, elle est réparée à l'aide de la procédure de réparation des séparateurs détaillée au paragraphe 2.4.2.2. Ensuite, elle est décodée (voir Algorithme 1) pour en déduire les séquences de mouvement $(S_h)_1$ et le nombre de robots H_1. Ensuite les séquences $(S_h)_1$ obtenues sont évaluées grâce au MILP de [Manier & Lamrous 2008], pour en déterminer le temps de cycle T_1. Nous obtenons ainsi les objectifs de la solution initiale résultante (H_1, T_1). Nous affectons la solution O_1 à la solution courante O_{cour} et à la meilleure solution globale $O_{best}(H_1)$ du nombre de robot H_1. Le temps de cycle T_1 est attribué au meilleur temps de cycle global $T_{best}(H_1)$.

Une fois cette étape d'initialisation réalisée, nous itérons $ITER$ fois les autres procédures de l'aVNS jusqu'à la satisfaction de la condition d'arrêt qui marque la fin de l'algorithme. Avant de détailler les autres procédures de l'aVNS, nous fournissons le panorama des structures de voisinages présélectionnées pour être employées dans la génération des voisinages de l'algorithme aVNS.

3.3.3 Structures de voisinage

Nous avons choisi plusieurs structures de voisinage à appliquer dans la génération des voisinages. Une structure de voisinage est une méthode de perturbation d'une solution. Elle agit sur un ou plusieurs attributs de la solution pour modifier leur valeur ou leur emplacement. L'idée principale est d'améliorer, si possible, la solution courante en y effectuant des séries de transformations ou de déplacements. La nouvelle solution transformée obtenue est la solution voisine. Les transformations possibles sont alors nommées les structures de voisinage. Elles prédéfinissent ainsi un type de mouvement ou de perturbation avec lequel les solutions voisines sont générées.

Compte tenu de notre méthode de codage et du problème traité, toute perturbation d'une solution peut modifier la valeur d'un ou des deux objectifs (H, T) et conduire à une solution totalement différente. Ainsi, tout au long de la procédure d'optimisation, chaque nombre possible de robots aura un chemin d'amélioration qui lui est propre. Cela signifie que nous améliorons le temps de cycle T pour chaque nombre possible de robots H : chaque fois qu'une nouvelle solution (H, T) est acceptée, nous comparons sa période T au dernier meilleur temps de cycle $T_{best}(H)$ enregistré pour le nombre de robot H.

Une structure de voisinage qui limite les changements à k composants de la solution est souvent appelée *k-optimale (k-opt)*. Pour notre algorithme aVNS, nous avons adopté six structures de voisinage de différents types, auxquelles nous donnons l'appellation générale N_{type}. Il s'agit de structures d'insertion, de suppression, de remplacement, d'échange, de décalage et d'inversion. Nous les notons respectivement par $N_{insertion}, N_{suppression}, N_{remplacement}, N_{échange}, N_{décalage}$ et $N_{inversion}$. Soit $\phi = (\phi(1), \phi(2), \ldots, \phi(T_{solution}))$ un encodage donné d'une solution courante O_{cour}, où $\phi(p)$ est un numéro de cuve ou un séparateur occupant la position p

($p = 1, \ldots, T_{solution}$). Nous décrivons ci-après les structures de voisinage que nous adoptons. Nous donnons plus de détails sur les choix assumés pour leur application.

3.3.3.1 Structure d'insertion

Nous identifions tout d'abord les numéros de cuves absents de la solution courante O_{cour}. Ensuite, nous choisissons au hasard certains d'entre eux que nous insérons à des positions aléatoires de la solution courante. Par conséquent, la structure d'insertion n'est appliquée que si la taille NC (nombre des numéros de cuves dans la solution) de la solution O_{cour} est différente de N. Si T_{insert} est le nombre des numéros de cuves insérés dans la solution courante, alors la taille totale $T_{solution}$ de la solution voisine créée est $T_{solution} = NC + NS + T_{insert}$.

Cette structure de voisinage est $k-opt$ où $k = T_{insert}$. Comme le nombre de cuves ajoutées T_{insert} est choisi au hasard et qu'il dépend du nombre de cuves absentes, k peut prendre les valeurs $1, 2, 3$, ou même plus. Dans le cas de $1-opt$, un seul numéro de cuve $\phi(t)$ absent de la solution O_{cour} est choisi au hasard et est inséré à la position inter-cuves aléatoire t de cette solution avec $0 \leq t \leq T_{solution}$, en comptant les deux positions extrêmes ($t = 0$ et $t = T_{solution}$), de sorte que l'encodage associé devient $\phi = (\phi(1), \phi(2), \ldots, \phi(t), \ldots, \phi(T_{solution}))$. Dans ce cas, $T_{solution} = NC + NS + 1$. Chaque numéro de cuve $\phi(t)$ inséré entraîne la suppression d'un mouvement à vide de la liste des mouvements à vide associés à la solution courante et son remplacement par deux nouveaux mouvements à vide dans la nouvelle liste de mouvements associée à la solution voisine : le premier se termine par le numéro de cuve $\phi(t)$ et le second commence par celui-ci.

La figure 3.3 illustre la structure d'insertion $N_{insertion}$ pour un problème à 13 cuves ($N = 13$). Cet exemple est celui choisi pour illustrer dans la suite les autres structures de voisinage employées. La solution courante O_{cour} mise en œuvre est $\{0, 7, 2, 5, 13, 8, 4, 9, 3, 10, 6, 1, 0\}$. Elle contient $NC = 11$ numéros de cuves et $NS = 2$ séparateurs. La taille totale est ainsi $T_{solution} = 13$. Les deux numéros de cuves 11 et 12 sont absents de la solution.

Un seul numéro de cuve parmi les deux absents de la solution O_{cour} est ici choisi au hasard : $\phi(t) = 12$. Il est inséré aléatoirement à la position inter-cuves $t = 6$ qui se trouve entre les numéros de cuves 8 et 4. La solution voisine obtenue est $O_{vois} = \{0, 7, 2, 5, 13, \mathbf{8, 12, 4}, 9, 3, 10, 6, 1, 0\}$. Dans ce cas, $T_{insert} = 1$ et la taille totale de la solution voisine est $T_{solution} = 14$. Ce mouvement d'insertion entraîne la suppression du mouvement à vide $E(8, 4)$ et le remplace par les deux mouvements à vide $E(8, 12)$ et $E(12, 4)$.

3.3.3.2 Structure de suppression

Pour cette structure, nous choisissons tout d'abord une position aléatoire t dans la solution courante O_{cour} ($1 \leq t \leq T_{solution}$). t ne doit pas être une position d'un

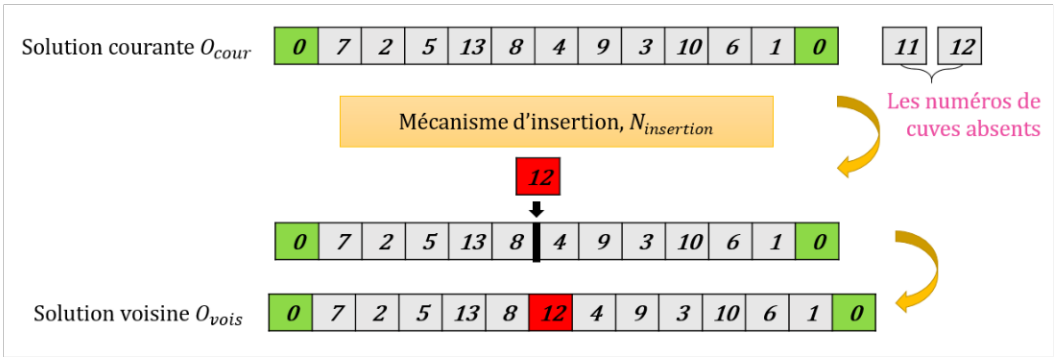

FIGURE 3.3 – Illustration de la structure d'insertion.

séparateur. Ensuite, nous supprimons le numéro de cuve $\phi(t)$ occupant la position t. Nous supprimons toujours un seul numéro de cuve. Ainsi, le nombre de numéros de cuves supprimés, noté T_{delet}, est toujours égal à 1. La taille totale de la nouvelle solution voisine est alors : $T_{solution} = NC + NS - T_{delet} = NC + NS - 1$. Cette structure de voisinage est, par conséquent, $1 - opt$. Dans ce cas, à partir d'une solution courante codée sous forme $\phi = (\phi(1), \phi(2), \ldots, \phi(t), \ldots, \phi(T_{solution}))$, nous obtenons la solution voisine $\phi = (\phi(1), \phi(2), \ldots, \phi(T_{solution}))$, où le numéro de cuve $\phi(t)$ est supprimé. Cela provoque la suppression de deux mouvements à vide de la liste des mouvements à vide associés à la solution courante et les remplace par un nouveau mouvement à vide dans la nouvelle liste de mouvements associée à la solution voisine : $E(\phi(t-1), \phi(t+1))$.

La figure 3.4 illustre la structure de suppression $N_{suppression}$ pour l'exemple précédent de solution courante. La position aléatoire choisie est $t = 7$, qui se trouve entre les numéros de cuves 8 et 9. Le numéro de cuve qui occupe cette position, $\phi(t) = 4$, est supprimé. La solution voisine résultante est alors $O_{vois} = \{0, 7, 2, 5, 13, \mathbf{8}, \mathbf{9}, 3, 10, 6, 1, 0\}$. La taille $T_{delet} = 1$ et la taille totale de la solution voisine est $T_{solution} = 12$. Ce mouvement de suppression entraîne l'élimination des deux mouvements à vide $E(8, 4)$ et $E(4, 9)$ et les remplace par $E(8, 9)$.

3.3.3.3 Structure de remplacement

La structure de remplacement, tout comme la structure d'insertion, ne peut pas être appliquée si la solution courante a une taille $NC = N$. Son mécanisme consiste à identifier d'abord les numéros de cuves absents de la solution O_{cour}, puis à en choisir un au hasard que nous remplaçons dans la solution courante par un de ses numéros de cuves, choisi aléatoirement. Nous ne procédons jamais à un remplacement par un séparateur de la solution courante, de sorte que le nombre de séparateurs NS par solution, reste intact. Cette structure de voisinage ne modifie pas la taille $T_{solution}$ de la solution courante. Ainsi, le nombre de numéros de cuves remplacés dans la solution O_{cour}, noté $T_{replace}$, est toujours égal à un, et

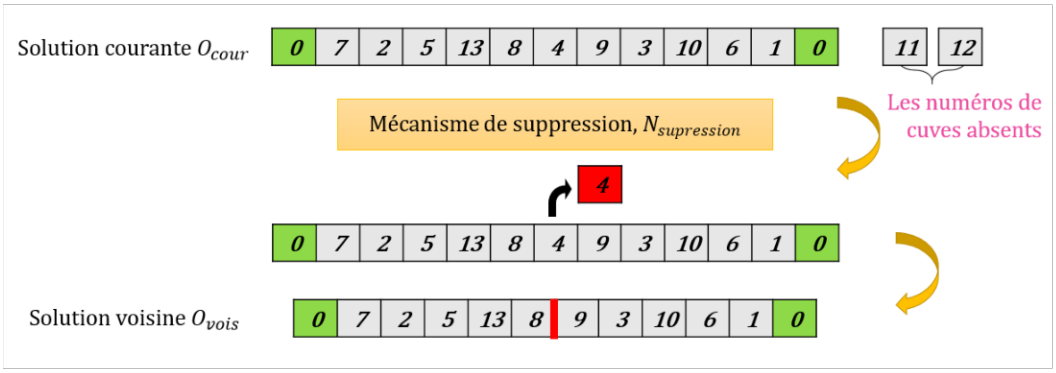

FIGURE 3.4 – Illustration de la structure de suppression.

la taille totale de la nouvelle solution voisine est la même que sa génératrice avec $T_{solution} = NC + NS - T_{insert} + T_{insert} = NC + NS$.

Cette structure de voisinage est aussi $1 - opt$. La solution courante et la solution voisine ont toutes deux l'encodage $\phi = (\phi(1), \phi(2), \ldots, \phi(t), \ldots, \phi(T_{solution}))$ où $\phi(t)$ est le numéro de cuve remplacé et t est sa position aléatoire $(1 \leq t \leq T_{solution})$. Le numéro de cuve remplacé entraîne la modification de deux mouvements à vide sur la liste des mouvements à vide de la solution courante, d'où la solution voisine, identifie deux nouveaux mouvements à vide $E(\phi(t-1), \phi(t))$ et $E(\phi(t), \phi(t+1))$.

Nous explicitons la structure de voisinage $N_{replacement}$ sur la figure 3.5. Le numéro de cuve absent de la solution courante et choisi aléatoirement est $\phi(t) = 12$. Il prend au hasard la position $t = 8$ occupée par le numéro de cuve 9 sur la solution O_{cour}. La solution voisine obtenue est ainsi : $O_{vois} = \{0, 7, 2, 5, 13, 8, \mathbf{4,12,3}, 10, 6, 1, 0\}$ de taille $T_{solution} = 13$. Ce mouvement de remplacement entraîne la modification des deux mouvements à vide $E(4, 9)$ et $E(9, 3)$ de la solution courante par les deux nouveaux mouvements à vide $E(4, 12)$ et $E(12, 3)$.

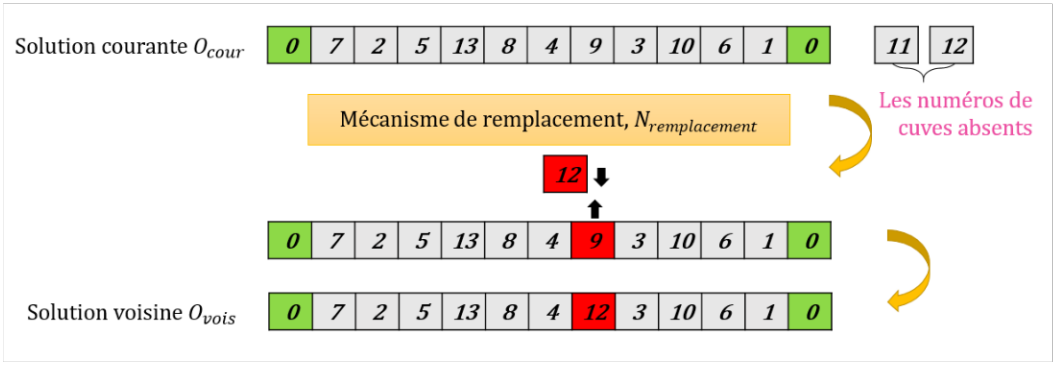

FIGURE 3.5 – Illustration de la structure de remplacement.

3.3.3.4 Structure d'échange

Nous commençons par choisir au hasard deux positions t_1 et t_2 de la solution courante ($1 \leq t_1 < t_2 \leq T_{solution}$). Nous interdisons le cas où les deux positions sont les mêmes ($t_1 = t_2$) car, par définition, la structure d'échange n'a d'intérêt que pour deux positions différentes. De même, nous ne permettons pas le choix de deux positions de séparateurs, car l'échange dans cette situation ne produit aucune différence sur la solution voisine. Sous ces conditions, nous échangeons les deux numéros de cuves $\phi(t_1)$ et $\phi(t_2)$ occupant les positions t_1 et t_2. Ainsi, si la solution courante est encodée $\phi = (\phi(1), \ldots, \phi(t_1), \ldots, \phi(t_2), \ldots, \phi(T_{solution}))$, la solution voisine est alors encodée par $\phi = (\phi(1), \ldots, \phi(t_2), \ldots, \phi(t_1), \ldots, \phi(T_{solution}))$. Par conséquent, le nombre de numéros de cuves échangés, noté T_{swap}, est toujours égal à deux et la taille de la solution voisine est la même que celle de la solution courante, $T_{solution} = NC + NS$.

La structure d'échange est $2 - opt$, car elle perturbe deux composantes de la solution courante. L'échange des deux numéros de cuves, $\phi(t_1)$ et $\phi(t_2)$, provoque la modification de quatre mouvements à vide de la liste des mouvements à vide de la solution courante. La solution voisine, ainsi, aura quatre nouveaux mouvements à vide : $E(\phi(t_1-1), \phi(t_2))$, $E(\phi(t_2), \phi(t_1+1))$, $E(\phi(t_2-1), \phi(t_1))$, et $E(\phi(t_1), \phi(t_2+1))$.

La figure 3.6 illustre le mécanisme de la structure d'échange $N_{échange}$ sur le même exemple de solution O_{cour}. Les deux positions choisies aléatoirement sur cette solution sont $t_1 = 4$ et $t_2 = 10$. La première se trouve entre les numéros de cuves 2 et 13 et est occupée par $\phi(t_1) = 5$, et la seconde est entre les numéros de cuves 3 et 6 et est occupée par $\phi(t_2) = 10$. Les numéros 5 et 10 sont échangés de sorte que $\phi(t_1) = 10$ et $\phi(t_2) = 5$. La solution voisine obtenue est $O_{vois} = \{0, 7, 2, \mathbf{10}, 13, 8, 4, 9, 3, \mathbf{5}, 6, 1, 0\}$ de taille $T_{solution} = 13$. Ce mouvement d'échange entraîne la modification des quatre mouvements à vide de la solution courante, $E(2, 5)$, $E(5, 13)$, $E(3, 10)$ et $E(10, 6)$ qui sont remplacés dans la solution voisine par $E(2, 10)$, $E(10, 13)$, $E(3, 5)$ et $E(5, 6)$.

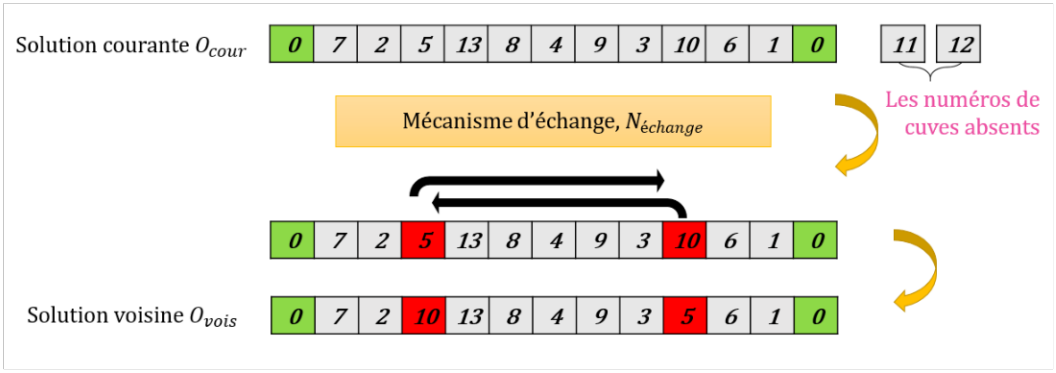

FIGURE 3.6 – Illustration de la structure d'échange.

3.3.3.5 Structure de décalage

La structure de décalage a d'autant plus un effet que le nombre de numéros de cuves NC de la solution O_{cour} est important. La procédure commence par le choix aléatoire de deux positions t_1 et t_2 de O_{cour}, de sorte que $1 \leq t_1 < t_2 \leq T_{solution}$. Si nous indiquons par w le nombre de positions intermédiaires entre t_1 et t_2, alors la position t_2 correspond à la position $t_1 + w + 1$. La solution courante est ainsi encodée $\phi = (\phi(1), \ldots, \phi(t_1), \phi(t_1 + 1), \ldots, \phi(t_1 + w), \phi(t_2), \ldots, \phi(T_{solution}))$. Par la suite, nous déplaçons par un mouvement circulaire la partie de la solution délimitée par les positions t_1 et t_2. La solution voisine résultante est ainsi $\phi = (\phi(1), \ldots, \phi(t_2), \phi(t_1), \phi(t_1 + 1), \ldots, \phi(t_1 + w), \ldots, \phi(T_{solution}))$. Il est clair que cette structure de voisinage n'altère pas la taille de la solution courante. La solution voisine a donc toujours la même taille $T_{solution} = NC + NS$. Le nombre de numéros de cuves décalés de manière circulaire, noté $T_{décalage}$, est égal à $w + 2$.

Ce type de voisinage est $k - opt$ car il perturbe k composantes de la solution courante, où $k = w + 2$. Plus précisément, lorsque le décodage des solutions est également circulaire, comme c'est le cas dans notre étude, le voisinage de décalage n'est que $2 - opt$. En effet le numéro de cuve $\phi(t_2)$ est retiré de sa position t_2 et est inséré en position t_1. Les numéros de cuves allant de $\phi(t_1)$ à $\phi(t_1 + w)$ ne sont pas modifiés mais juste décalés chacun d'une position vers la droite. Ainsi, les mouvements à vide identifiés à partir de ces numéros de cuves (de $\phi(t_1)$ à $\phi(t_1 + w)$) restent les mêmes. Par conséquent, le voisinage de décalage est similaire à un voisinage de suppression-insertion qui effectue une suppression du numéro de cuve $\phi(t_2)$ et une insertion de celui-ci sur la position t_1. Comme ces deux voisinages sont chacun de $1 - opt$, le voisinage de décalage est de $2 - opt$. Le numéro de cuve $\phi(t_2)$ supprimé de la solution courante entraîne le changement de deux mouvements à vide et son insertion sur la position t_1 entraîne la modification d'un autre mouvement. La solution voisine a par conséquent trois nouveaux mouvements à vide, $E(\phi(t_1 - 1), \phi(t_2))$, $E(\phi(t_2), \phi(t_1))$ et $E(\phi(t_2 - 1), \phi(t_2 + 1))$. Notons que $\phi(t_2 - 1)$ correspond à $\phi(t_1 + w)$ et que $\phi(t_2 + 1)$ correspond à $\phi(t_1 + w + 2)$. Nous illustrons sur la figure 3.7 la structure de décalage $N_{décalage}$, toujours pour le même exemple. Les deux positions choisies au hasard sur la solution O_{cour} et échangées sont $t_1 = 5$ et $t_2 = 9$ qui sont occupées respectivement par les numéros de cuves $\phi(t_1) = 13$ et $\phi(t_2) = 3$. Le nombre de positions intermédiaires entre 13 et 3 est $w = 3$, et $T_{décalage} = w + 2 = 5$. La solution voisine obtenue est $O_{vois} = \{0, 7, 2, 5, \mathbf{3}, 13, 8, 4, \mathbf{9,\ 10}, 6, 1, 0\}$ de taille $T_{solution} = 13$. Ce mouvement de décalage remplace les trois mouvements à vide de la solution courante $E(5, 13), E(9, 3)$ et $E(3, 10)$ par les trois nouveaux mouvements à vide $E(5, 3), E(3, 13)$ et $E(9, 10)$ sur la solution voisine.

3.3.3.6 Structure d'inversion

La structure d'inversion, tout comme la structure de décalage, a un effet intéressant lorsque le nombre des numéros de cuves NC de la solution O_{cour} est important. Nous choisissons tout d'abord deux positions aléatoires différentes t_1 et t_2 sur la

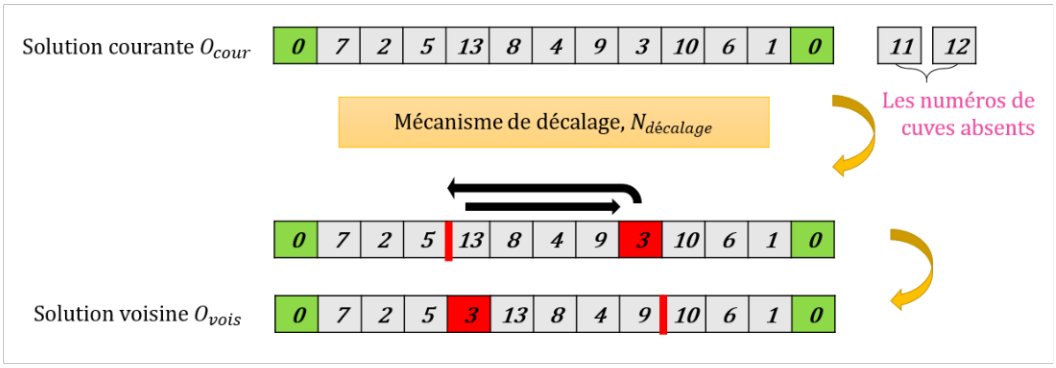

FIGURE 3.7 – Illustration de la structure de décalage.

solution courante ($1 \leq t_1 < t_2 \leq T_{solution}$). Ensuite, nous inversons la chaîne des numéros de cuves située entre t_1 et t_2, de sorte que le dernier numéro de cuve $\phi(t_2)$ devient le premier et occupera la position t_1, l'avant dernier numéro de cuve $\phi(t_2 - 1)$ devient le second et occupera la position $t_1 + 1$, et ainsi de suite jusqu'à ce que le premier numéro de cuve $\phi(t_1)$ devienne le dernier et prenne la position t_2. Si nous considérons la solution courante $\phi = (\phi(1), \ldots, \phi(t_1), \phi(t_1 + 1), \ldots, \phi(t_2 - 1), \phi(t_2), \ldots, \phi(T_{solution}))$, alors la solution voisine résultant de cette inversion est $\phi = (\phi(1), \ldots, \phi(t_2), \phi(t_2 - 1), \ldots, \phi(t_1 + 1), \phi(t_1), \ldots, \phi(T_{solution}))$. La taille $T_{solution}$ de la solution courante n'est pas modifiée avec cette structure de voisinage, et la solution voisine est donc de taille $T_{solution} = NC + NS$. Le nombre de numéros de cuves inversés, noté $T_{inversion}$, est égal à $t_2 - t_1 + 1$.

Cette structure de voisinage est $k - opt$, car elle perturbe k composantes de la solution courante, où $k = T_{inversion}$. Dans le cas où le décodage des solutions aboutit à des couples ou des arcs sur lesquels une orientation différente n'affecte pas le coût du mouvement (le coût du mouvement ou la distance sur les arcs ne change pas si l'orientation du mouvement change), cette structure de voisinage n'est que de $2 - opt$. En fait, la partie de la solution entre les positions t_1 à t_2 est inversée de telle sorte qu'elle ne modifie que l'orientation des mouvements à vide. Les couples de numéros de cuves qui composent ces mouvements à vide restent les mêmes. Ainsi, seuls les numéros de cuves $\phi(t_1)$ et $\phi(t_2)$ après inversion, entraînent chacun la modification d'un mouvement à vide, car ils sont collés chacun à un nouveau numéro de cuve, $E(\phi(t_1 - 1), \phi(t_2))$, $E\phi(t_1), \phi(t_2 + 1)$.

Nous explicitons la structure d'inversion $N_{inversion}$ sur la figure 3.8. Les deux positions aléatoires choisies sont $t_1 = 5$ et $t_2 = 9$ occupées respectivement par les numéros de cuves $\phi(t_1) = 13$ et $\phi(t_2)$=3. La suite des numéros de cuves qui se trouve entre ces deux positions est inversée et la solution voisine résultante est $O_{vois} = \{0, 7, 2, 5, \mathbf{3, 9, 4, 8, 13}, 10, 6, 1, 0\}$ de taille $T_{solution} = 13$. Nous avons $T_{inversion} = t_2 - t_1 + 1 = 5$. Ce mouvement d'inversion entraîne la modification des mouvements à vide de la solution courante $(5, 13)$, $(13, 8)$, $(8, 4)$, $E(4, 9)$, $E(9, 3)$ et

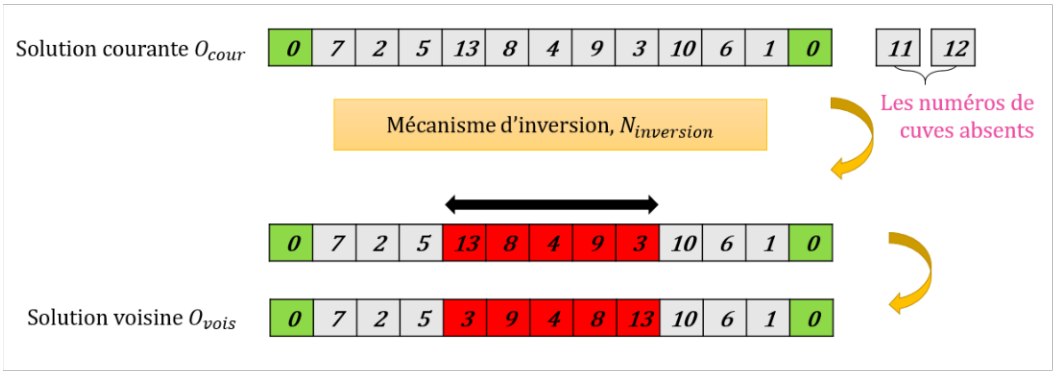

FIGURE 3.8 – Illustration de la structure d'inversion.

$E(3, 10)$ par les nouveaux mouvements à vide de la solution voisine $(5, 3)$, $(3, 9)$, $(9, 4)$, $E(4, 8)$, $E(8, 13)$ et $E(13, 10)$.

3.3.4 Choix d'une structure de voisinage

À chaque itération *iter* de l'algorithme aVNS, une seule structure de voisinage est sélectionnée et appliquée parmi les six structures prédéfinies N_{type}. Elle permet la génération d'un voisinage $N_{type}(O_{cour})$ de taille $Size$, à partir de la solution courante O_{cour}. Notre *procédure de sélection* effectue un choix aléatoire d'une structure de voisinage en fonction de la taille $T_{solution}$ de la solution courante O_{cour}, comme suit :

- Si $2 + NS \leq T_{solution} \leq 4 + NS$ alors le nombre de numéros de cuves dans la solution est compris entre 2 et 4. Cette procédure choisit systématiquement la structure d'insertion $N_{insertion}$ pour enrichir la solution ;
- Si $T_{solution} = N + NS$ (valeur maximale), nous ne pouvons appliquer que l'une des quatre structures suivantes, $N_{suppression}$, $N_{échange}$, $N_{décalage}$ ou $N_{inversion}$. Ainsi, le choix est effectué de manière aléatoire où les structures ont une équi-probabilité de $1/4$ d'être sélectionnée ;
- Autrement, si $5 + NS \leq T_{solution} \leq N - 1 + NS$, nous n'avons aucune contrainte de choix, et nous choisissons donc aléatoirement l'une des six structures prédéfinies. De même ici, chaque structure a la probabilité de $1/6$ d'être sélectionnée.

La procédure de sélection est explicitée ci-dessous via l'algorithme 2. Elle fonctionne en fonction de la taille $T_{solution}$ comme donnée d'entrée et renvoie la variable de sortie CNS qui contient la structure de voisinage choisie.

3.3.5 Génération de voisinage

À l'itération *iter*, après avoir sélectionné la structure de voisinage CNS à appliquer, la procédure de génération de voisinage crée $Size$ solutions voisines différentes O_{vois} à partir de la solution courante O_{cour} en utilisant le mécanisme de voisinage

Algorithme 2 Procédure de sélection de la structure de voisinage

Entrées : $T_{solution}$
Sorties : CNS
début
  si $2 + NS \leq T_{solution} \leq 4 + MNS(N)$ **alors**
    $CNS \leftarrow N_{insertion}$
  sinon
    si $T_{solution} = N + NS$ **alors**
      générer un nombre entier aléatoire $prob \in [1, 4]$
      switch *la valeur de prob* **do**
        case *1* **do**
          $CNS \leftarrow N_{suppression}$
        case *2* **do**
          $CNS \leftarrow N_{échange}$
        case *3* **do**
          $CNS \leftarrow N_{décalage}$
        otherwise do
          $CNS \leftarrow N_{inversion}$
    sinon
      générer un nombre entier aléatoire $prob \in [1, 6]$
      switch *la valeur de prob* **do**
        case *1* **do**
          $CNS \leftarrow N_{insertion}$
        case *2* **do**
          $CNS \leftarrow N_{suppression}$
        case *3* **do**
          $CNS \leftarrow N_{remplacement}$
        case *4* **do**
          $CNS \leftarrow N_{échange}$
        case *5* **do**
          $CNS \leftarrow N_{décalage}$
        otherwise do
          $CNS \leftarrow N_{inversion}$
  retourner CNS

CNS. Chaque solution voisine O_{vois} est décodée selon la procédure de décodage décrite au chapitre 2 (voir Algorithme 1), pour en déduire les séquences de mouvements associées $(S_h)_{vois}$ et le nombre de robots H_{vois}. Suite au décodage, les séquences de mouvements identifiées sont évaluées avec le modèle MILP suivant la procédure d'évaluation (section 2.4.3) pour vérifier leur faisabilité et calculer leur temps de cycle T_{vois} résultant et les dates d'exécution des opérations. Finalement, nous obtenons $Size$ solutions voisines O_{vois} avec leurs triplets de résultats $(H_{vois}, (S_h)_{vois}, T_{vois})$.

3.3.6 Procédure d'élection

Cette procédure compare les valeurs H_{vois}, T_{vois} des solutions voisines générées O_{vois} pour choisir la meilleure solution voisine $(O_{vois})_{best}$ de l'itération $iter$. Nous excluons d'abord toutes les solutions voisines non faisables et nous ne gardons que celles faisables ($T_{vois} \neq 0$). Ensuite, nous désignons la meilleure solution voisine en considérant les deux objectifs H et T. Pour cela, nous privilégions un objectif par rapport à l'autre pendant l'opération d'élection, en donnant un ordre de priorité entre les deux objectifs. Ainsi, deux actions sont possibles :

— *Priorité-T* : Ne garder en priorité que les solutions voisines ayant le temps de cycle minimal $(T_{vois})_{best}$. Nous pouvons trouver plus d'une solution avec le même temps de cycle. S'il y en a plusieurs, nous sélectionnons la solution ayant le nombre minimal de robots $(H_{vois})_{best}$;

— *Priorité-H* : Identifier en priorité les solutions voisines ayant le nombre de robots minimal $(H_{vois})_{best}$, et s'il y en a plusieurs, nous choisissons la solution ayant le temps de cycle minimal $(T_{vois})_{best}$.

Il paraît évident que l'action *Priorité-T* n'est pas la meilleure, ce qui a été confirmé par les premières simulations. En effet, lorsque nous priorisons la minimisation du temps de cycle T, nous ne gardons, dans la plupart des cas, que les solutions voisines avec un nombre de robots élevé car plus le nombre de robots augmente, plus la période du cycle diminue. En outre, comme cette *Priorité-T* est répétée à chaque itération, nous intensifions la recherche vers les voisinages des solutions à meilleurs temps de cycle, ce qui nous éloigne progressivement des voisinages des solutions à nombre minimal de robots. En conséquence, nous avons choisi d'appliquer l'action *Priorité-H*, qui privilégie la minimisation du nombre de robots H, pour déterminer la meilleure solution voisine de chaque voisinage créé sur chaque itération $iter$ de l'aVNS.

Par ailleurs, dans quelques cas rares, il arrive qu'il n'y ait aucune solution faisable dans le voisinage généré. Dans ce cas, nous sélectionnons l'une des solutions voisines qui a le nombre minimal de robots $(H_{vois})_{best}$ pour être la meilleure solution voisine $(O_{vois})_{best}$, même si $(T_{vois})_{best} = 0$. L'algorithme 3 explicite cette procédure d'élection, avec comme paramètres en entrée H_{vois}, T_{vois} et O_{vois} qui sont de taille $Size$.

Algorithme 3 Procédure d'élection du meilleur voisin

Entrées : O_{vois}, H_{vois}, T_{vois}
Sorties : $(O_{vois})_{best}$, $(H_{vois})_{best}$, $(T_{vois})_{best}$
début
  Garder seulement les solutions voisines faisables avec leurs résultats respectifs : $(O_{vois})_{faisable}$, $(H_{vois})_{faisable}$, $(T_{vois})_{faisable}$
  si $(T_{vois})_{feasible}$ *n'est pas vide* **alors**
    $(H_{vois})_{best} \leftarrow \min((H_{vois})_{faisable})$ $(T_{vois})_{best} \leftarrow \min((T_{vois})_{faisable}$ de $(H_{vois})_{best})$
    $(O_{vois})_{best} \leftarrow (O_{vois})_{faisable}$ de $(H_{vois})_{best}$ et $(T_{vois})_{best}$
  sinon
    $(H_{vois})_{best} \leftarrow \min(H_{vois})$ $(T_{vois})_{best} \leftarrow 0$ $(O_{vois})_{best} \leftarrow$ one O_{vois} of $(H_{vois})_{best}$ and $(T_{vois})_{best}$
  retourner $(O_{vois})_{best}$, $(H_{vois})_{best}$, $(T_{vois})_{best}$

3.3.7 Procédure d'actualisation

Cette procédure commence par effectuer deux tests sur la meilleure solution voisine élue $(O_{vois})_{best}$. Nous vérifions si elle est faisable et si son temps de cycle $(T_{vois})_{best}$ est inférieur au meilleur temps de cycle global $T_{best}(H)$ étant enregistré pour le nombre de robots $H = (H_{vois})_{best}$. Si les tests sont validés, alors la valeur enregistrée pour le meilleur temps de cycle $T_{best}(H)$ est mise à jour : elle prend la valeur du temps de cycle du meilleur voisin $(T_{vois})_{best}$. Parallèlement, $O_{best}(H)$ reçoit la meilleure solution voisine élue $(O_{vois})_{best}$. Sinon, dans le cas où le résultat de ces tests est négatif, ces affectations ne sont pas réalisées. Ensuite, la solution courante O_{cour} reçoit la meilleure solution voisine élue $(O_{vois})_{best}$ et l'algorithme boucle avec la procédure 3.3.5 si la condition d'arrêt n'est toujours pas vérifiée.

La dernière étape signifie ici que l'algorithme passe toujours à la nouvelle meilleure solution voisine $(O_{vois})_{best}$, même s'il n'enregistre pas d'amélioration supplémentaire du temps de cycle, car les solutions sont générées avec des nombres différents de robots et cela permet de valoriser le deuxième objectif H. En effet, si une amélioration de la meilleure durée de cycle T_{best} est produite pour le nombre de robots $H = (H_{vois})_{best}$ $((T_{vois})_{best} < T_{best}(H_{vois})_{best})$, alors cette dernière reçoit la valeur améliorée $((T_{vois})_{best})$. De même, la dernière meilleure solution $O_{best}((H_{vois})_{best})$ enregistrée pour le nombre de robots $H = (H_{vois})_{best}$ reçoit la nouvelle solution voisine ayant apporté cette amélioration, $(O_{vois})_{best}$. Par exemple, si nous avons $T_{best} = 280$ pour le nombre de robots $H = 1$ et la meilleure solution voisine à l'itération considérée a un temps de cycle $(T_{vois})_{best} = 263$ avec un nombre de robots $(H_{vois})_{best} = 1$, donc ici il y a une amélioration du meilleur temps de cycle enregistré pour le nombre de robots $H = 1$, alors $T_{best}(1) = 263$, $O_{best}(1) = (O_{vois})_{best}$ et $O_{cour} = (O_{vois})_{best}$.

Dans le cas où le nombre de robots H_{cour} de la solution courante O_{cour} et le nombre de robots $(H_{vois})_{best}$ de la meilleure solution voisine $(O_{vois})_{best}$ sont différents, la comparaison entre les durées de cycle respectifs, T_{cour} et $(T_{vois})_{best}$ n'est

pas pertinente. C'est pourquoi il n'existe pas un schéma d'amélioration strict et régulier, sur chaque itération, de la solution courante O_{cour}. Et nous sommes obligés de changer la solution courante, donc de passer à la nouvelle meilleure solution voisine $(O_{vois})_{best}$, qu'elle améliore ou pas le temps de cycle T_{cour} de la solution courante.

Enfin, si la solution courante ne porte aucun indice d'amélioration alors il convient de revenir aux meilleures solutions trouvées précédemment pour chaque nombre de robots grâce à la traçabilité à intégrer au processus.

Ainsi, dans le cas général, lorsqu'une métaheuristique basée sur la recherche locale est développée, deux pistes possibles peuvent être mises en œuvre à chaque itération :

— Autoriser un saut vers la meilleure solution du voisinage nouvellement élue uniquement si une amélioration de la solution courante est produite. Cette stratégie de la recherche est connue sous le nom de la *méthode de descente* (ou *Descent method*) ;

— Autoriser toujours le saut vers la nouvelle meilleure solution voisine élue même si elle n'améliore pas la solution courante. Cette deuxième stratégie est connue comme la *méthode de descente-montée* (*Descent-ascent method*).

En outre, si à chaque itération de l'algorithme :

— Une seule solution voisine est générée (taille du voisinage $Size = 1$), la décision de procéder au saut ou pas est toujours liée à cette seule solution, qui est la première explorée dans le voisinage. Cette méthode de découverte du voisinage est connue sous le nom de la *méthode de première amélioration* (*First improvement method*) ;

— Toutes les solutions voisines sont générées ($Size$ devient grand), la décision de procéder au saut ou pas est faite en considérant toutes ces solutions et en jugeant la meilleure parmi celles-ci en fonction des objectifs à optimiser. Cette seconde méthode d'exploration du voisinage est connue comme la *méthode de la meilleure amélioration* (ou *Best improvement method*).

Or, lorsque le problème est NP-difficile, sa complexité croît de manière exponentielle avec l'augmentation de la taille du problème. Il devient alors plus difficile et plus coûteux en temps de calcul de générer toutes les solutions possibles dans chaque voisinage exploré. Dans ce cas, il reste possible et plus logique de générer un nombre limité de solutions voisines à chaque itération de l'algorithme, parmi lesquelles nous sélectionnons la meilleure. Le voisinage ainsi généré aura une taille précise et stable tout au long des itérations de l'algorithme.

En conséquence, selon cette synthèse élaborée sur les éventuelles stratégies adoptées pour la recherche des solutions et l'exploration de l'espace de recherche, nous déduisons que l'algorithme aVNS que nous développons, applique une méthode

de recherche basée à la fois sur les stratégies de *descente-montée* et de *meilleure amélioration*.

3.3.8 Condition d'arrêt

Généralement, un critère d'arrêt dans les algorithmes métaheuristiques peut s'exprimer par un nombre maximal d'itérations de l'algorithme, un nombre maximal d'itérations après la dernière amélioration, ou un temps de calcul maximal autorisé. Dans notre cas, nous avons choisi borner le nombre d'itération de l'algorithme aVNS avec $ITER$.

Dans le paragraphe suivant, nous proposons une approche encore améliorée de l'algorithme aVNS.

3.4 Approche aVNS améliorée

Nous avons testé notre algorithme aVNS sur les jeux de tests de la littérature décrit au chapitre précédent, comme détaillé dans la section suivante. Cela nous a incités à enrichir l'approche précédente avec une procédure supplémentaire intégrée de *retour en arrière* (*backtrack procedure*) dans le but d'améliorer encore ses performances. La nouvelle version de l'algorithme est appelée *aVNSBT* pour *aVNS with Bactrack*. Nous illustrons sur la figure 3.9 le changement de structure de l'algorithme initial. Cela se situe au niveau de la procédure d'actualisation. Toutes les autres procédures du premier algorithme aVNS restent identiques.

Le principe de cette procédure de retour en arrière est de revenir sur certaines solutions visitées précédemment et que nous jugeons prometteuses pour une ou plusieurs raisons. Pour expliquer le mécanisme de cette procédure, nous introduisons deux nouvelles variables :

— *Seuil* : nombre d'itérations après lequel le retour en arrière est effectué. Ce nombre doit être un diviseur du nombre global d'itérations $ITER$ de l'aVNS ;

— $O_{seuil}(H_{seuil})$: solution ayant un nombre de robots H_{seuil}, à laquelle l'algorithme retourne après l'exécution de *Seuil* itérations.

La solution seuil O_{seuil} est la meilleure solution enregistrée pour un nombre donné de robots, H_{seuil}. En effet, nous enregistrons les meilleures solutions trouvées pour chaque nombre de robots si nous considérons plus qu'un seul nombre de robots, la solution seuil est alors la meilleure solution enregistrée pour le nombre de robots qui présente l'amélioration la plus ancienne au fil des itérations. Par exemple, si nous considérons les nombres de robots de 1 à 4 dans la procédure de retour en arrière, et que nous sommes sur une itération seuil (*iter* est un multiple du nombre *Seuil*), nous cherchons le nombre de robots parmi ceux considérés sur lequel nous avons enregistré l'amélioration la plus ancienne du temps de cycle. Si, à titre d'exemple,

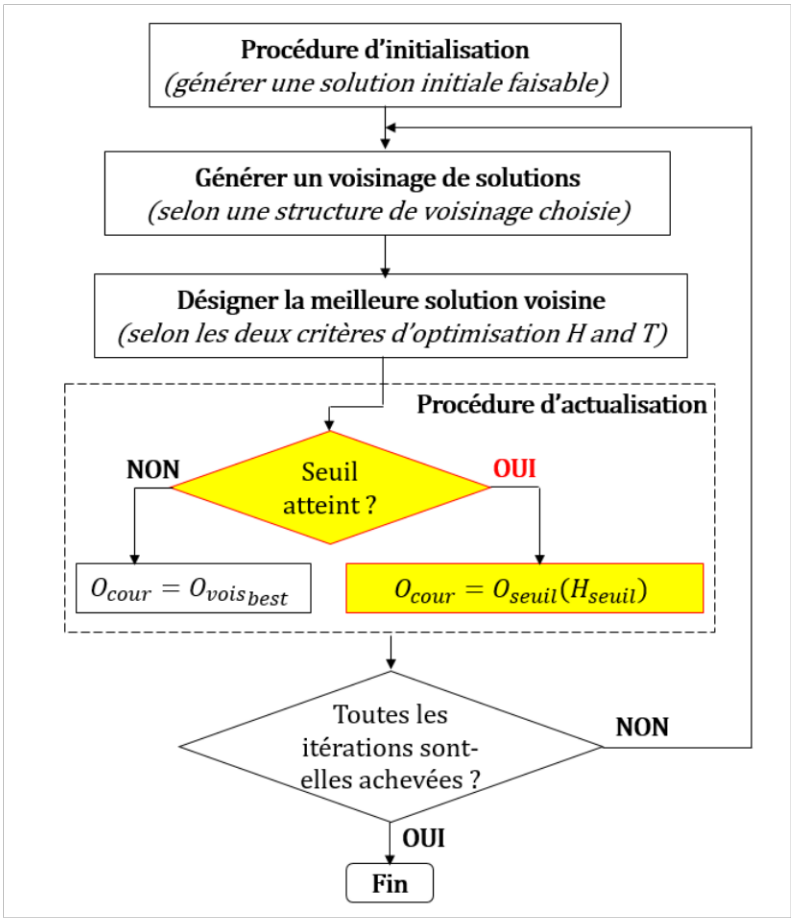

FIGURE 3.9 – Intégration de la procédure de retour en arrière dans l'aVNS.

c'est $H = 2$ alors la solution seuil $O_{seuil}(H_{seuil}) = O_{best}(2)$.

Les nombres de robots testés seuls ou choisis aléatoirement sont dans l'ensemble $H = \{1, 2, 3, 4\}$. Nous avons peu d'intérêt de revenir sur des solutions avec un nombre de robots supérieur à 4, car premièrement cela biaise la recherche des solutions à nombre de robots minimal, et deuxièmement, les solutions avec plus de 4 robots ($H > 4$) sont abondantes et donc faciles à trouver.

La procédure de retour en arrière se déroule comme suit :

(1) Nous initialisons la variable *Seuil* à l'étape d'initialisation de l'aVNS (procédure 3.3.2).

(2) A l'étape d'actualisation, avant d'affecter la meilleure solution voisine $(O_{vois})_{best}$ à la solution courante O_{cour}, nous effectuons le test suivant. Si la recherche est passée par *Seuil* itérations (le nombre incrémenté d'itérations *iter* est divisible par le nombre *Seuil*), alors la solution courante O_{cour} devient la solution seuil O_{seuil} que l'algorithme désigne comme expliqué ci-dessus. Sinon, la solution

courante devient normalement la meilleure solution voisine $(O_{vois})_{best}$.

3.5 Expérimentations et résultats

3.5.1 Environnement de tests

L'algorithme général de résolution, aVNS, est implémenté sur le logiciel *Matlab* version R2016a et exécuté sur un processeur Intel®Core™i5-6500 à 3,20 GHz. Le MILP est toujours résolu grâce à *Hybrid Toolbox*[1].

3.5.2 Instances testées

Il n'est jamais évident de définir les paramètres appropriés d'un algorithme métaheuristique pour mieux l'orienter vers les meilleurs résultats. Dans cette optique, l'aVNS a été premièrement testé sur l'instance *P&U* [Phillips & Unger 1976] pour ajuster au mieux ses différents paramètres. Dans un deuxième temps, pour valider l'ensemble des paramètres choisis lors de la première étape, l'aVNS a été appliqué sur les deux autres instances, *Ligne1* et *Ligne2*.

Enfin, et pour mettre davantage en lumière les performances de notre méthode de résolution, nous avons étendu les exécutions à cinq autres instances de référence, dont les données se trouvent dans l'article de Leung et al. [Leung *et al.* 2004]. Il s'agit de *Mini-Phil* (instance *P&U* tronquée aux huit premières cuves), et de quatre autres instances tirées de Shapiro et al. [Shapiro & Nuttle 1988] (*Black Oxide1*, *Black Oxide2*, *Cuivre* et *Zinc*). Ces quatre instances ont été modifiées par Leung et al. [Leung *et al.* 2004], car ces auteurs ne considéraient pas les cuves dupliquées comme dans les instances originales. Nous notons ces adaptations *BO1-v2*, *BO2-v2*, *Cuivre-v2* et *Zinc-v2*. Les données temporelles associées sont fournies en Annexe (sections A.4, A.5, A.6 et A.7).

3.5.3 Premiers résultats et ajustement des paramètres

Les résultats présentés dans cette partie proviennent des tests effectués sur l'instance *P&U*. Nous nous sommes référés aux meilleurs temps de cycle trouvés dans la littérature sur cette instance, T_{opt}, qui ne sont associés qu'à un ou deux robots ($H = 1$ ou $H = 2$). Le premier a été prouvé par Shapiro et Nuttle [Shapiro & Nuttle 1988] comme étant le temps de cycle optimal pour une résolution d'un CHSP mono-robot, avec $T_{opt}(1) = 521\,\text{s}$. Le second, $T_{opt}(2) = 251\,\text{s}$, a été fourni par Lei et Wang [Lei & Wang 1991]. Pour les autres nombres de robots où $H > 2$, il n'existe dans la littérature aucune résolution optimale pour le CHSP.

1. *A. Bemporad. "Hybrid Toolbox" - User's Guide, 2004. http ://cse.lab.imtlucca.it/ bemporad/hybrid/toolbox.*

Ci-après, nous décrivons les résultats préliminaires qui nous ont permis de déterminer les paramètres de l'algorithme. Chaque fois, un seul des paramètres est modifié sur un ensemble de choix ou de valeurs, tandis que les autres restent fixes.

3.5.3.1 Priorité-T ou Priorité-H

Dans la procédure d'élection mise en œuvre dans notre aVNS, nous avons énuméré deux stratégies d'élection de la meilleure solution voisine, en privilégiant la stratégie *Priorité-H* qui priorise l'objectif lié au nombre de robots par rapport à *Priorité-T* centrée sur le temps de cycle. Le tableau 3.1 évalue l'impact du choix de chacune des deux stratégies sur le temps de cycle pour 1 à 7 robots de l'instance P&U. Il fournit le meilleur temps de cycle T_{best} et le meilleur temps de cycle moyen T_{moy} obtenus sur cinq exécutions. Lors de la réalisation de ces expériences, nous avons fixé le nombre maximum d'itérations à $ITER = 10\,000$ itérations et la taille du voisinage à $Size = 20$ solutions.

TABLE 3.1 – Influence des deux stratégies d'élection sur les résultats de l'instance *P&U*.

Nombre de robots H	*Priorité-T*		*Priorité-H*	
	T_{best}	T_{moy}	T_{best}	T_{moy}
1	1202	1361.6	733	775.6
2	434	498.2	324	359.1
3	214	223.7	204	209.9
4	151	158.7	151	165
5	151	151	151	151
6	151	151	151	151.8
7	151	151	201	290.8

Ces résultats confortent notre choix d'application de la stratégie d'élection *Priorité-H* qui surpasse *Priorité-T*, principalement pour les plus petits nombres de robots, non seulement sur le meilleur temps de cycle T_{best} mais aussi sur le temps de cycle moyen T_{moy}. L'option *Priorité-H* est seulement moins performante lorsque $H = 7$ et sur T_{moy} pour $H \in \{4, 6\}$. Ce résultat est attendu ici car avec l'option *Priorité-H*, la recherche est davantage orientée vers la minimisation du nombre de robots. Cependant, nous pouvons facilement affiner ces résultats en agissant sur d'autres paramètres, par exemple en augmentant le nombre d'itérations $ITER$.

3.5.3.2 Méthode de descente ou méthode de descente-montée

Nos tests portent ici sur la méthode de saut vers la meilleure solution voisine décrite au paragraphe 3.3.7. Parmi les deux stratégies possibles, nous avons choisi d'appliquer la *méthode de descente-montée*. Elle permet de changer à chaque itération la solution courante O_{cour} par la meilleure solution voisine élue $(O_{vois})_{best}$,

même s'il n'y a pas amélioration du temps de cycle T_{cour}. Les résultats numériques, présentés dans le tableau 3.2, justifient notre choix. Ils se déclinent en résultats de l'aVNS d'abord exécuté avec la *méthode de descente*, puis exécuté avec la *méthode de descente-montée*. Dans le tableau, nous reportons le meilleur temps de cycle T_{best} et le meilleur temps de cycle moyen T_{moy} sur cinq exécutions. Durant ces simulations, nous avons gardé les mêmes paramètres de l'aVNS qu'au paragraphe précédent ($ITER = 10000$ et $S = 20$).

TABLE 3.2 – Influence des deux méthodes de saut sur les résultats de l'instance *P&U*.

H	*Méthode descente*		*Méthode descente-montée*	
	T_{best}	T_{moy}	T_{best}	T_{moy}
1	776	1014	733	775.6
2	338	453.4	324	359.1
3	210	236.4	204	209.9
4	163	171.5	151	165
5	151	156.9	151	151
6	151	153.6	151	151.8
7	151	899.6	201	290.8

Nous remarquons qu'avec la *méthode de descente-montée*, les résultats sont en général meilleurs aussi bien sur T_{best} que sur T_{moy}. Pour le nombre de robots le plus grand ($H = 7$), T_{best} est un peu plus grand que celui de la *méthode de descente*, car nous appliquons également ici l'option *Priorité-H*. Néanmoins, nous montrons par la suite, qu'avec un nombre plus important d'itérations ($ITER > 10\,000$), toutes les valeurs optimales sont atteintes même pour les plus grands nombres de robots.

Dans les simulations qui vont suivre, les deux meilleurs choix argumentés, l'option *Priorité-H* et la *méthode de descente-montée*, sont conservés dans l'aVNS.

3.5.3.3 Influence du nombre d'itérations

Pour étudier l'influence du nombre d'itérations $ITER$ sur les résultats, nous avons fixé une taille du voisinage $Size = 20$ solutions et fait varier le nombre d'itérations $ITER$ dans l'ensemble $\{100, 1\,000, 10\,000, 100\,000, 1\,000\,000\}$. Nous résumons les résultats des exécutions effectués dans le tableau 3.3, en termes de temps T_{best} et T_{moy}, sur cinq exécutions, sauf pour $ITER = 1\,000\,000$, où seulement deux exécutions ont été réalisés en raison du temps de calcul très important. La dernière ligne du tableau fournit le temps CPU moyen (en secondes) sur les exécutions, pour chaque valeur de $ITER$. Rappelons que tous les $T_{best}(H)$ sont initialisés à la valeur 2000 comme borne supérieure de l'objectif T.

TABLE 3.3 – Influence du nombre d'itérations $ITER$ sur les résultats de l'instance $P\&U$.

H	$ITER$									
	100		1000		10000		100000		1000000	
	T_{best}	T_{moy}	T_{best}	T_{moy}	T_{best}	T_{moy}	T_{best}	T_{moy}	T_{best}	T_{moy}
1	1147	1191	918	981.8	733	775.6	563	715.6	697	715
2	479	534.2	358	409.6	323.5	359.1	253	282.2	264	266.8
3	223	305.9	215	249.5	204	209.9	183	197.5	192	194.5
4	178	236.9	164	191	151	165	151	157.9	151	151
5	160	917.3	152.5	174.8	151	151	151	151	151	151
6	170	1634	162	255	151	151.8	151	151	151	151
7	2000	2000	215	1643	201	290.8	151	151	151	151
Temps CPU (s)	9.336		93.176		1038.55		10237.5		123560	

Il ressort de ces résultats que plus le nombre d'itérations est élevé, meilleur est le temps de cycle, quel que soit le nombre de robots H, non seulement sur le T_{best} mais aussi sur le T_{moy}. La seule exception est enregistrée pour $ITER = 1\,000\,000$, où T_{best} est médiocre par rapport à celui de $ITER = 100\,000$. Cela peut s'expliquer par le nombre réduit des exécutions réalisés pour $ITER = 1\,000\,000$. Néanmoins, $ITER = 1\,000\,000$ présente de petites améliorations du T_{moy} par rapport à celui de $ITER = 100\,000$, qui peuvent être la conséquence de l'augmentation du nombre d'itérations.

Quant au temps de calcul, il augmente sans surprise avec le nombre d'itérations $ITER$. L'amélioration des temps de cycle n'est pas si importante en comparant les résultats de $ITER = 100\,000$ et $ITER = 1\,000\,000$, alors que le second consomme beaucoup plus de temps de calcul. Dans la limite du nombre d'exécutions effectuées et de l'instance considérée, la valeur $ITER = 100\,000$ paraît un bon compromis entre qualité de la solution et temps de résolution, comme nous pouvons le visualiser sur les graphes de la figure 3.10.

En conséquence, nous avons choisi 100 000 comme meilleure valeur du paramètre $ITER$. Son temps de calcul moyen est équivalent à 2,8 heures par exécution. Jusqu'à présent, c'est un temps de calcul acceptable dans la mesure où la taille de l'espace de recherche est importante. De plus, le CHDSP que nous résolvons dans la présente étude n'est pas considéré dans un contexte dynamique. Il s'inscrit dans le cadre d'une phase de conception, ce qui signifie que le problème n'est résolu qu'une fois lors de l'implantation ou la reconfiguration d'une installation de galvanoplastie.

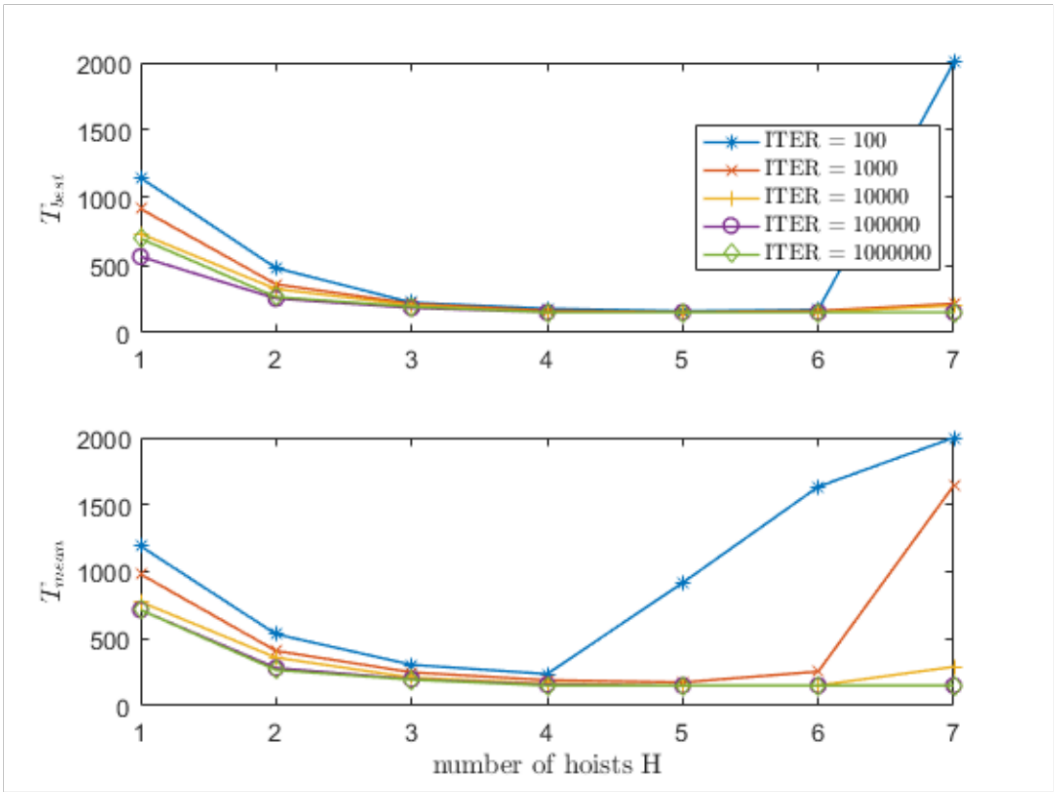

FIGURE 3.10 – T_{best} et T_{moy} sous les cinq valeurs testées de $ITER$ pour l'instance $P\&U$.

3.5.3.4 Influence de la taille du voisinage

Le paramètre suivant à étudier est la taille de voisinage $Size$. Un meilleur ajustement de ce paramètre est un facteur de performance important de notre algorithme. Nous avons exécuté l'aVNS avec différentes valeurs de $Size$ prises dans l'ensemble $\{10, 15, 20, 25, 30, 40\}$. Les paramètres déjà ajustés sont conservés pour ces simulations. Nous fournissons dans le tableau 3.4 T_{best} et T_{moy} pour chaque valeur de $Size$, cette fois pour dix exécutions. De même, la dernière ligne du tableau affiche le temps CPU moyen (en secondes) pour toutes les exécutions. Pour la taille de voisinage $Size = 40$ et le nombre de robots $H = 7$, nous retrouvons $T_{best} = T_{moy} = 2\,000$, ce qui signifie qu'il n'y a pas eu de solutions explorées correspondant à cette configuration.

La meilleure valeur obtenue est $Size = 20$ pour les deux temps de cycle T_{best} et T_{moy}. La figure 3.11 représente graphiquement les mêmes résultats et confirme qu'avec $S = 20$, nous obtenons le meilleur temps de cycle. Nous retenons donc cette valeur de $Size$ pour le reste des expérimentations.

TABLE 3.4 – Influence de la taille du voisinage $Size$ sur les résultats de l'instance *P&U*.

H	$Size$					
	10		15		20	
	T_{best}	T_{moy}	T_{best}	T_{moy}	T_{best}	T_{moy}
1	718	750	689	754.8	563	723.2
2	310	352.2	253	330.3	253	299.7
3	205	213.4	207.5	214	196	202.4
4	162.5	171.7	151	170.3	151	163.9
5	151	152.1	151	152.9	151	151.6
6	151	151	151	153.3	151	153.2
7	151	157.3	151	159.5	151	156
Temps CPU (s)	4289.02		6197.56		8497.57	
H	$Size$					
	25		30		40	
	T_{best}	T_{moy}	T_{best}	T_{moy}	T_{best}	T_{moy}
1	718	755.1	718	736.5	718	736.3
2	254	312.2	270	320.7	282	324.6
3	200	208.6	200	210.7	205	209.4
4	165	175.6	164	173.5	165	175.2
5	151	161	151	168.6	151	165.6
6	151	168.7	151	176.7	164	375.7
7	151	565.5	165.33	1278	2000	2000
Temps CPU (s)	9691.96		11791.78		15461.14	

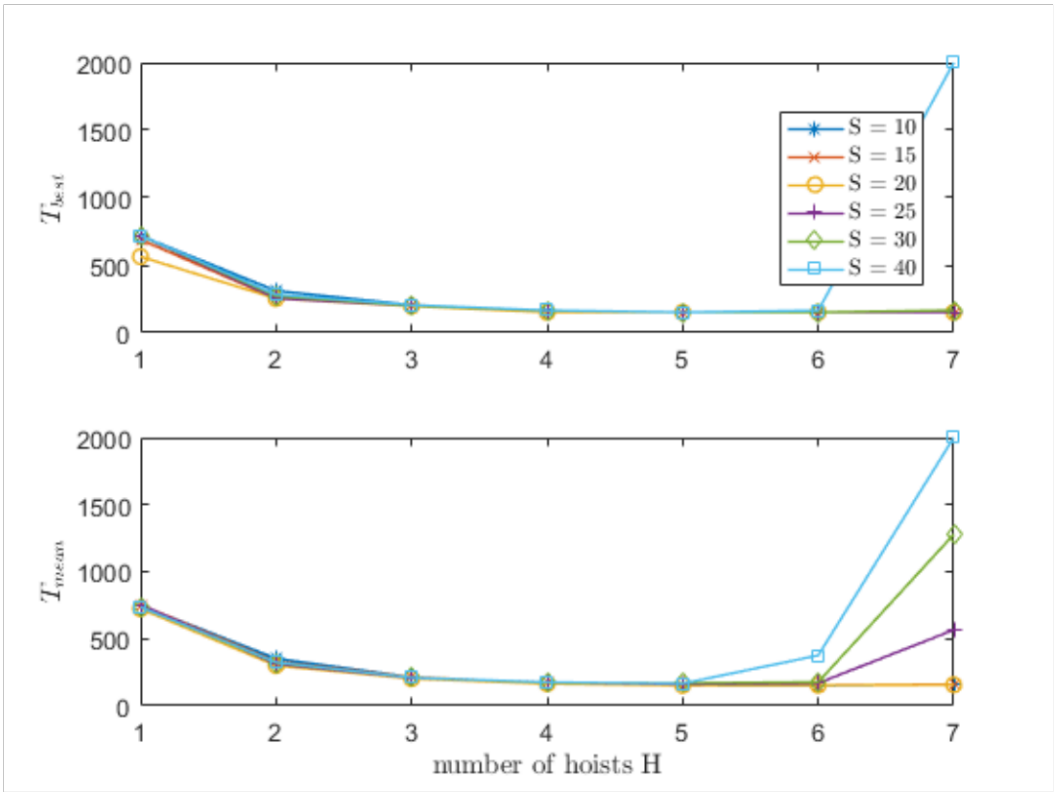

FIGURE 3.11 – T_{best} et T_{moy} sous les six valeurs testées de $Size$ pour l'instance $P\&U$.

3.5.3.5 Influence de la structure de décalage

Pendant la phase d'ajustement des paramètres, d'autres exécutions menant à de meilleurs résultats ont révélé qu'un autre changement doit être procédé lors de l'application des structures de voisinage. En effet, tel qu'il est conçu, l'algorithme aVNS génère de manière aléatoire les solutions voisines à chaque itération et il ne contient aucune contrainte qui évite de générer les mêmes solutions que celles déjà explorées auparavant. Par conséquent, l'algorithme pourrait répéter la génération d'un certain nombre de mêmes solutions. Cela limite l'exploration de l'espace de recherche, tout en consommant du temps de calcul. Ainsi, une idée pour réduire le nombre de répétitions est de rendre inactive la structure de voisinage de décalage, $N_{décalage}$. Assurément, comme elle est décrite dans le paragraphe 3.3.3.5, cette structure emploie essentiellement un mouvement circulaire pour modifier les solutions. En outre, lorsque le décodage est également circulaire, comme c'est le cas dans notre étude, la structure de décalage génère la même solution dans le cas où elle décale une partie de numéros de cuves délimitées par deux séparateurs. La solution voisine et la solution courante, une fois décodées, donnent alors les mêmes séquences de mouvements. Les résultats des exécutions réalisées confirmant ce choix sont fournis dans le tableau 3.5. Nous présentons T_{best} et T_{moy} obtenus en exécutant l'aVNS avec et sans la structure de décalage $N_{décalage}$, pour dix exécutions. Ceux obtenus

avec la stratégie de recherche sans $N_{décalage}$ sont nettement meilleurs que ceux avec maintien de la structure de décalage.

TABLE 3.5 – Influence de la structure de voisinage $N_{décalage}$ sur les résultats de l'instance $P\&U$.

H	$aVNS$			
	avec $N_{décalage}$		sans $N_{décalage}$	
	T_{best}	T_{moy}	T_{best}	T_{moy}
1	563	723.2	521	701.5
2	253	299.7	251	257.3
3	196	202.4	185.3	191.6
4	151	163.9	151	151.7
5	151	151.6	151	151
6	151	153.2	151	151
7	151	156	151	151

En conséquence, nous ne conservons dans notre algorithme aVNS que cinq structures de voisinage sur les six initialement envisagées. Notons que des tests effectués avec et sans les autres structures de voisinage nous ont permis de conclure sur l'intérêt de celles-ci.

3.5.4 Évaluation des performances

3.5.4.1 Tests sur l'instance *P*&*U*

Nous évaluons les performances de l'algorithme $aVNS$ et de sa version améliorée $aVNSBT$, en les comparant à celles d'autres méthodes. Nous commençons par les tests réalisés sur l'instance de $P\&U$ et nous présentons sur le tableau 3.6 les résultats obtenus de l'aVNS en termes de meilleurs temps de cycle T_{best} et de temps de cycle moyens T_{moy} sur 20 répétitions. Le tableau fournit également les temps de cycle optimaux T_{opt} trouvés dans la littérature et deux écarts, notés EC_{best} et EC_{moy}. Ces derniers désignent l'écart relatif de l'algorithme considéré (noté $target$) par rapport à l'algorithme comparé (noté $base$), respectivement pour T_{best} et T_{moy} :

$$EC_{best}(H) = 100\left(\frac{T_{best"base"}(H) - T_{best"target"}(H)}{T_{best"base"}(H)}\right) \tag{3.1}$$

$$EC_{moy}(H) = 100\left(\frac{T_{moy"base"}(H) - T_{moy"target"}(H)}{T_{moy"base"}(H)}\right) \tag{3.2}$$

Dans ce tableau, nous observons que l'aVNS est capable de trouver les temps de cycle optimaux à un ou deux robots : $T_{opt}(1) = 521$ secondes et $T_{opt}(2) = 251$ secondes. En second lieu, nous constatons que notre algorithme surpasse l'algorithme GA sur les meilleurs temps de cycle T_{best} avec un écart moyen positif de $13,26\%$ et sur les temps de cycle moyens T_{moy} avec un écart moyen positif de $11,06\%$. Les améliorations importantes n'ont concerné que les résultats des petits nombres de robots ($1 \leq H \leq 4$), car pour $5 \leq H \leq 7$, GA a déjà trouvé les meilleurs résultats. En effet, quand le nombre de robots est petit, les robots restent les ressources critiques de la ligne, car la productivité de la ligne dépend de l'efficacité de ces robots. Dès $H = 4$, c'est une des cuves qui devient critique, comme nous l'avons déjà expliqué (dans 2.5.3). Dans le cas de l'instance $P\&U$, la cuve critique est la cuve numéro 2 (voir annexe A.1, tableau A1.1), car elle a le plus grand temps de trempe minimal, $m_2 = 150$. Par conséquent, le temps de cycle stagne à $T = 151$ avec une différence d'une seconde qui est le résultat d'une contrainte du problème.

TABLE 3.6 – Comparaison entre aVNS et GA sur l'instance de $P\&U$.

		GA		aVNS			
H	T_{opt}	T_{best}	T_{moy}	T_{best}	T_{moy}	EC_{best}	EC_{moy}
1	521	665	673.8	**521**	706.6	21.65	-4.86
2	251	332	346	**251**	**255.9**	24.39	26.04
3	–	196	205.8	**185.3**	**191.1**	5.45	7.14
4	–	210	226.4	**151**	**152.1**	28.09	32.81
5	–	151	159.4	151	**151**	0	5.26
6	–	–	–	**151**	**151**	–	–
7	–	151	151	151	151	0	0
Moyennes						13.26	11.06

Par ailleurs, aVNS n'a été moins performant que sur le temps de cycle moyen $T_{mean}(1)$ du nombre de robots $H = 1$. Cela peut s'expliquer par le fait que les solutions à un robot ne sont pas nombreuses, voire rares, dans un espace de recherche immense. En conséquence, l'aVNS, bien qu'il se montre capable de fournir de meilleurs résultats que ceux de GA et d'atteindre les solutions optimales, doit être encore amélioré pour atteindre plus de solutions à un robot. Ce but est atteint grâce à la version aVNSBT. Nous affichons les résultats sur le tableau 3.7, toujours pour l'instance $P\&U$. Les conditions de test restent les mêmes que celles de l'aVNS sauf celles liées à la procédure de retour en arrière. Nous avons exécuté aVNSBT en fixant le seuil à 100 itérations ($Seuil = 100$). La procédure de retour en arrière ne prend en compte que les meilleures solutions enregistrées à 1 ou 2 robots ($O_{seuil}(1)$ ou $O_{seuil}(2)$, où $H_{seuil} \in \{1, 2\}$).

Il ressort clairement de ces résultats que l'algorithme aVNSBT est encore plus

TABLE 3.7 – Résultats de l'aVNSBT sur l'instance *P&U*.

		aVNS		aVNSBT			
H	T_{opt}	T_{best}	T_{moy}	T_{best}	T_{moy}	EC_{best}	EC_{moy}
1	521	521	706.6	**521**	**656.2**	0	7.13
2	251	251	255.9	**251**	**252.2**	0	1.44
3	–	185.3	191.1	**182**	**189.5**	1.78	0.83
4	–	151	152.1	**151**	**151.7**	0	0.26
5	–	151	151	151	**151**	0	0
6	–	151	151	**151**	**151**	0	0
7	–	151	151	151	151	0	0
Moyennes						0.25	1.38

performant que l'algorithme de base, aVNS. Cela confirme tout d'abord l'effet indéniable de la procédure intégrée de retour en arrière pour améliorer le défaut de l'aVNS. Cette approche améliorée est plus performante dans la fouille de l'espace de recherche et est capable d'atteindre plus de solutions à un robots. $T_{best} = 521$ étant déjà trouvé pour $H = 1$ avec l'aVNS, l'aVNSBT cependant, améliore plus le temps de cycle moyen T_{moy} à un robot, avec un écart positif $EC_{moy} = 7,13\%$. L'amélioration apportée par l'aVNSBT a surtout affecté les temps de cycles moyens des plus petits nombres de robots ($1 \leq H \leq 4$), avec une moyenne des écarts égale à $1,38\%$. La seule amélioration concernant les meilleurs temps de cycle, est de l'ordre de $1,78\%$ pour le nombre de robots $H = 3$. Ainsi, la procédure de retour en arrière améliore la robustesse de l'algorithme aVNS et montre son efficacité pour parvenir à un nombre plus grand de meilleures solutions.

3.5.4.2 Tests sur les instances *Ligne1* et *Ligne2*

Dans cette section, nous présentons les résultats obtenus pour les instances *Ligne1* et *Ligne2*. Dans la littérature, pour ces deux problèmes, nous ne trouvons le temps de cycle optimal T_{opt} que pour un robot $H = 1 : T_{opt}(1) = 425$ pour l'instance *Ligne1* et $T_{opt}(1) = 712$ pour l'instance *Ligne2*. Nous fournissons sur le tableau 3.8 les résultats de l'aVNS sur l'instance *Ligne1* et sur le tableau 3.9 ceux sur l'instance *Ligne2*. Les tableaux sont toujours organisés de la même façon où ils exhibent les temps T_{best} et T_{moy}, avec les écarts EC_{best} et EC_{moy} de l'aVNS par rapport à l'algorithme GA.

Pour l'instance *Ligne1*, aVNS améliore les résultats obtenus par l'approche GA sur le meilleur temps de cycle T_{best} avec un écart moyen positif de $0,75\%$ et sur le temps de cycle moyen T_{moy} avec un écart moyen positif de $5,22\%$. Pour les meilleurs temps de cycle, l'amélioration ne concerne que les résultats de la configuration à un robot et pour les temps de cycle moyens, elle ne concerne que les solutions à un ou deux robots, car pour $H > 2$, GA avait déjà détecté les meilleurs résultats. En effet,

TABLE 3.8 – Comparaison entre l'aVNS et les résultats de la littérature sur l'instance *Ligne1*.

		GA		aVNS			
H	T_{opt}	T_{best}	T_{moy}	T_{best}	T_{moy}	EC_{best}	EC_{moy}
1	425	452	677	**428**	**498.2**	5.31	26.41
2	–	361	402	361	**361**	0	10.19
3	–	361	361	361	361	0	0
4	–	361	361	361	361	0	0
5	–	361	361	361	361	0	0
6	–	361	361	361	361	0	0
7	–	361	361	361	361	0	0
Moyennes						0.75	5.22

TABLE 3.9 – Comparaison entre l'aVNS et les résultats de la littérature sur l'instance *Ligne2*.

		GA		aVNS			
H	T_{opt}	T_{best}	T_{moy}	T_{best}	T_{moy}	EC_{best}	EC_{moy}
1	712	858	1294	**712**	**772.4**	17.01	40.31
2	–	661	662	661	**661**	0	0.15
3	–	661	661	661	661	0	0
4	–	661	661	661	661	0	0
5	–	661	661	661	661	0	0
6	–	661	661	661	661	0	0
7	–	678	678	661	661	2.50	2.50
Moyennes						2.79	6.13

à partir de deux robots, la cuve 1 devient critique, avec $m_1 = 360$ (voir annexe A.2, tableau A2.1). Pour $H = 1$, aVNS est capable de trouver une très bonne solution avec $T_{best} = 428$ secondes, proche de l'optimum qui est de 425 secondes.

Quant à l'instance *Ligne2*, aVNS surpasse l'algorithme GA sur le meilleur temps de cycle T_{best} avec un écart positif moyen de $2,79\%$ et sur le temps de cycle moyen T_{moy} avec un écart positif moyen de $6,13\%$. L'amélioration du T_{best} ne concerne que les solutions à un ou sept robots, et l'amélioration du T_{moy} englobe les solutions à un, deux ou sept robots. En outre, aVNS retrouve le temps de cycle optimal à un robot $T_{opt}(1) = 712$ secondes à la différence de GA où, l'écart en T_{best} entre aVNS et GA pour $H = 1$ est de l'ordre $17,01\%$. Nous soulignons aussi que, comme pour la *Ligne1*, prévoir plus que 2 robots est inutile puisque dès 2 robots, c'est une cuve qui devient critique et conditionne le temps de cycle $T_{best} = 661$. Il s'agit ici de la cuve 14, avec $m_{14} = 660$ (voir annexe A.3, tableau A3.1).

Par ailleurs, l'aVNSBT a aussi été exécuté sur les problèmes *Ligne1* et *Ligne2* avec la même valeur du paramètre $Seuil = 100$. Cependant, la procédure de retour en arrière ici, ne prend en compte que les meilleures solutions enregistrées pour 1 robot ($O_{seuil}(1)$, où $H_{seuil} \in \{1\}$). Nous réunissons dans le tableau 3.10 les résultats obtenus par l'aVNSBT sur ces deux instances.

TABLE 3.10 – Résultats de l'aVNSBT sur les 2 instances *Ligne1* et *Ligne2*.

Instances			aVNS		aVNSBT			
	H	T_{opt}	T_{best}	T_{moy}	T_{best}	T_{moy}	EC_{best}	EC_{moy}
Ligne1	1	425	428	498.2	**425**	**480.4**	0.7	3.57
	2	–	361	361	361	361	0	0
	3	–	361	361	361	361	0	0
	4	–	361	361	361	361	0	0
	5	–	361	361	361	361	0	0
	6	–	361	361	361	361	0	0
	7	–	361	361	361	361	0	0
Ligne2	1	712	712	772.4	712	**731.1**	0	5.34
	2	–	661	661	661	661	0	0
	3	–	661	661	661	661	0	0
	4	–	661	661	661	661	0	0
	5	–	661	661	661	661	0	0
	6	–	661	661	661	661	0	0
	7	–	661	661	661	661	0	0

Ces résultats confirment l'intérêt de l'aVNSBT par rapport à l'aVNS. En effet, ce dernier améliore les temps de cycle moyens de l'aVNS pour $H = 1$, avec un écart positif EC_{moy} égal à $3,57\%$ pour l'instance *Ligne1* et à $5,34\%$ pour l'instance *Ligne2*. En ce qui concerne les meilleurs temps de cycle, aVNSBT améliore encore le T_{best} à 1 robot ($H = 1$) pour l'exemple *Ligne1*, jusqu'à trouver la solution optimale. Enfin, ces résultats valident la conclusion établie à l'issue des tests réalisés sur la première instance $P\&U$, à savoir que la procédure de retour en arrière intégrée dans l'aVNS est bénéfique car elle améliore la robustesse de l'algorithme aVNS, en accentuant la diversification des voisinages visités au cours de la recherche des solutions.

3.5.4.3 aVNSBT versus SGA

Dans les deux sections précédentes, nous avons montré la robustesse de l'algorithme aVNS et l'apport de la procédure intégrée de retour en arrière en les comparant aux résultats de la littérature. Néanmoins, il reste intéressant d'évaluer

TABLE 3.11 – Comparaison entre aVNSBT et SGA sur les instances *P&U*, *Ligne1* et *Ligne2*.

Instances			SGA		aVNSBT			
	H	T_{opt}	T_{best}	T_{moy}	T_{best}	T_{moy}	EC_{best}	EC_{moy}
P&U	1	521	521	1024	521	**656.2**	0	35.91
	2	251	264	360	**251**	**252.2**	4.92	29.94
	3	—	192.33	231.5	**182**	**189.5**	5.37	18.14
	4	—	163	207	**151**	**151.7**	7.36	26.71
	5	—	151.33	154.14	**151**	**151**	0.21	2.03
	6	—	151	—	151	**151**	0	—
	7	—	151	151	151	151	0	0
Moyennes							2.55	16.10
Ligne1	1	425	492	1032	**425**	**480.4**	13.61	53.44
	2	—	361	404	361	**361**	0	10.64
	3	—	361	361	361	361	0	0
	4	—	361	361	361	361	0	0
	5	—	361	361	361	361	0	0
	6	—	361	361	361	361	0	0
	7	—	361	361	361	361	0	0
Moyennes							1.94	9.15
Ligne2	1	712	712	1332	712	**731.1**	0	45.11
	2	—	661	662	661	**661**	0	0.15
	3	—	661	661	661	661	0	0
	4	—	661	661	661	661	0	0
	5	—	661	661	661	661	0	0
	6	—	661	661	661	661	0	0
	7	—	661	661	661	661	0	0
Moyennes							0	6.46

les performances de cette deuxième approche que nous proposons par rapport à SGA, que nous avons présenté dans le chapitre 2. Notons que la différence entre l'approche aVNS, éventuellement aVNSBT, et l'approche SGA repose essentiellement sur la méthode de recherche des meilleures solutions admises dans l'espace de recherche du problème. Nous présentons ainsi sur le tableau 3.11 les résultats obtenus par ces deux approches sur les instances *P&U*, *Ligne1* et *Ligne2*, toujours avec les meilleures durées de cycle T_{best}, les temps de cycle moyens T_{moy} et les deux écarts EC_{best} et EC_{moy}, qui traduisent le pourcentage d'amélioration de l'aVNSBT par rapport à SGA.

L'amélioration la plus importante correspond au benchmark *P&U*. L'algorithme aVNSBT est capable de trouver la solution optimale à 2 robots $T_{opt}(2) = 251$ se-

condes, ce qui n'a pas été le cas avec l'algorithme SGA. Il améliore aussi les meilleurs temps de cycle T_{best} pour les nombres de robots $3 \leq H \leq 5$. L'écart moyen EC_{best} sur tous les nombres de robots est de de $2,55\%$. En ce qui concerne les temps de cycle moyens T_{moy}, aVNSBT enregistre également de meilleurs résultats sur les nombres de robots $H \leq 6$, ce qui mène à un écart moyen positif de l'ordre de $16,10\%$.

Sur l'instance *Ligne1*, aVNSBT améliore les résultats obtenus par SGA en termes des meilleures durées de cycle T_{best} avec un écart moyen positif de $1,94\%$, et en termes des temps de cycle moyens T_{moy} avec un écart moyen positif de $9,15\%$. Ce nouvel algorithme est capable de trouver la solution optimale à un robot $T_{opt}(1) = 425$ secondes, pour laquelle SGA était un peu loin, et de booster les moyennes en temps de cycle pour 1 et 2 robots. Enfin, sur le benchmark *Ligne2*, SGA étant capable de détecter les meilleures solutions T_{best} pour tous les nombres de robots, l'aVNSBT n'améliore que les temps de cycle moyens T_{moy} pour un et deux robots, avec un écart moyen positif de $6,46\%$.

En conséquence, ces améliorations apportées par l'algorithme aVNSBT par rapport à SGA, sur les meilleurs temps de cycle T_{best} soulignent sa capacité à atteindre plus de meilleures solutions de l'espace de recherche et montrent une meilleure trajectoire de recherche. Ses améliorations sur les temps de cycle moyens T_{moy} confirment sa robustesse et mettent en vigueur la diversification accentuée de cette méthode de recherche.

3.5.5 Expérimentations supplémentaires

Suite à ces premiers résultats encourageants, nous avons étendu nos test à cinq instances supplémentaires : *Mini-Phil*, *BO1-v2*, *BO2-v2*, *Cuivre-v2* et *Zinc-v2*, sur le tableau 3.12. Nous organisons ce tableau de la même façon que le précédent, avec les T_{best} et les T_{moy} pour chaque algorithme, ainsi que les deux écarts EC_{best} et EC_{moy} qui présentent l'amélioration apportée par aVNSBT via SGA. Nous notons que les optima T_{opt} fournis pour ces benchmarks sont les temps de cycle optimaux trouvés dans [Leung *et al.* 2004], et que la borne supérieure en temps de cycle pour tous les nombres de robots avec laquelle l'algorithme SGA commence la recherche est égale à 4000 (celle de l'aVNSBT est de 2000).

En analysant ces résultats, nous observons que l'aVNSBT est capable de trouver les solutions optimales à un robot ($H = 1$) pour les trois dernières instances, à la différence de l'algorithme SGA qui est moins performant. Toujours pour un robot, l'aVNSBT, bien qu'il donne une solution à 11.25% de l'optimum pour l'instance *BO1-v2*, améliore la meilleure solution de SGA avec un écart positif égal à 27.63%. Sur l'instance *Mini-Phil*, les deux algorithmes enregistrent la même performance en T_{best} pour tous les nombres de robots, en restant à 20.99% de l'optimum pour un robot ($T_{opt}(1) = 281$ secondes). Pour deux robots, aVNSBT et SGA ont des performances similaires en T_{best} qui sont égales aux optima pour toutes les ins-

TABLE 3.12 – Performances de l'aVNSBT et de SGA sur les instances *Mini-Phil*, *BO1-v2*, *BO2-v2*, *Cuivre-v2* et *Zinc-v2*.

Instances			SGA		aVNSBT			
	H	T_{opt}	T_{best}	T_{moy}	T_{best}	T_{moy}	EC_{best}	EC_{moy}
Mini-Phil	1	281	340	346.36	**340**	**340**	0	1.83
	2	165.5	165.5	166.23	165.5	**165.5**	0	0.44
	3	—	151	153.18	151	**151**	0	1.42
	4	—	151	151	151	151	0	0
	5	—	151	151	151	151	0	0
	6	—	151	151	151	151	0	0
	7	—	151	151	151	151	0	0
Moyennes							0	0.53
BO1-v2	1	299.5	460,4	460,4	**333.2**	**333.95**	27.63	27.46
	2	265.9	275.65	285.29	275.65	**275.65**	0	3.38
	3	—	247.3	247.3	247.3	247.3	0	0
	4	—	247.3	247.3	247.3	247.3	0	0
	5	—	247.3	247.3	247.3	247.3	0	0
	6	—	247.3	247.3	247.3	247.3	0	0
	7	—	247.3	247.3	247.3	247.3	0	0
Moyennes							3.95	4.41
BO2-v2	1	279.3	327.6	337.6	**279.3**	**283.27**	14.74	16.09
	2	240.1	240.1	240.1	**240.1**	**240.1**	0	0
	3	—	240.1	240.1	240.1	240.1	0	0
	4	—	240.1	240.1	240.1	240.1	0	0
	5	—	240.1	240.1	240.1	240.1	0	0
	6	—	240.1	240.1	240.1	240.1	0	0
	7	—	240.1	240.1	240.1	240.1	0	0
Moyennes							2.11	2.30
Cuivre-v2	1	1847.2	2063.3	2096.66	**1847.2**	**1847.2**	10.47	11.90
	2	1800.1	1800.1	1800.1	**1800.1**	**1800.1**	0	0
	3	—	1800.1	1800.1	1800.1	1800.1	0	0
	4	—	1800.1	1800.1	1800.1	1800.1	0	0
	5	—	1800.1	1800.1	1800.1	1800.1	0	0
	6	—	1800.1	1800.1	1800.1	1800.1	0	0
	7	—	1800.1	1800.1	1800.1	1800.1	0	0
Moyennes							1.50	1.70
Zinc-v2	1	1743.4	3041.6	3041.6	**1743.4**	**1743.4**	42.68	42.68
	2	1680.1	1680.1	1738.49	**1680.1**	**1680.1**	0	3.36
	3	—	1680.1	1680.1	1680.1	1680.1	0	0
	4	—	1680.1	1680.1	1680.1	1680.1	0	0
	5	—	1680.1	1680.1	1680.1	1680.1	0	0
	6	—	1680.1	1680.1	1680.1	1680.1	0	0
	7	—	1680.1	1680.1	1680.1	1680.1	0	0
Moyennes							6.10	6.58

tances sauf pour *BO1-v2*, avec un décalage de 3.67% ($T_{best}(2) = 275,65$ s alors que $T_{opt}(2) = 265.9$ s). Quand $H \geq 3$ et pour les 5 instances, les meilleurs temps de cycle T_{best} sont les mêmes pour aVNSBT et SGA.

Pour l'aVNSBT, nous notons que les T_{best} stagnent pour le benchmark *Mini-Phil* à partir de $H = 3$ avec un $T_{best} = 151$ secondes (même valeur obtenue pour l'instance originale *P&U* pour $H = 4$), ce qui signifie que ce sont les cuves qui deviennent les ressources critiques de la ligne et non plus les robots. Il en va de même pour l'instance *BO1-v2*, dès trois robots le meilleur temps de cycle stagne sur la valeur $T_{best} = 247.3$ secondes. Notons que pour cette instance n'avons pas trouvé d'optima pour $H \geq 3$, mais nous pouvons expliquer cette stagnation par le fait que probablement les cuves deviennent les ressources critiques de la ligne. Pour ce jeu de données, deux cuves ont le plus grand temps minimal de trempe qui sont la cuve 1 et la cuve 12. Ce temps est de l'ordre de 240 secondes (voir annexe A.4, tableau A4.1). Pour les trois dernières instances, le meilleur temps de cycle ne change plus à partir du nombre de robots $H = 2$, avec la cuve 12 qui devient critique pour le problème *BO2-v2*, la cuve 7 le devient pour le problème *Cuivre-v2* et la cuve 9 le devient pour le problème *Zinc-v2*. Leurs temps minimaux de trempe sont respectivement $m_{12} = 240$ s, $m_7 = 1800$ s et $m_9 = 1680$ s (voir annexes A.5, A.6 et A.7, tableaux A5.1, A6.1 et A7.1).

Concernant les temps de cycle moyens T_{moy}, aVNSBT retrouve les optima T_{opt} sur les deux dernières instances et toujours les mêmes valeurs que T_{best} pour les nombres de robots $H \geq 2$ pour toutes les instances sauf *BO1-v2* pour $H = 2$, où le décalage avec T_{opt} est de l'ordre de 11.25%. Sur le nombre de robots $H = 1$, aVNSBT améliore les temps de cycle moyens T_{moy} par rapport à ceux de SGA avec des écarts positifs : 1.83% pour *Mini-Phil*, 27.46% pour *BO1-v2*, 16.09% pour *BO2-v2*, 11.90% pour *CUIVRE-v2* et 42.68% pour *ZINC-v2*. Sur cette contribution, nous notons que, même si aVNSBT ne retrouve pas les solutions optimales pour des cas rares, il a pu améliorer les résultats de SGA et fournir de meilleurs temps de cycle moyens. Ceci met encore une fois en lumière la performance et la stabilité de cet algorithme. Nous pouvons expliquer les trois cas où il ne détecte pas les solutions optimales (pour 1 robot sur l'instance *Mini-Phil* et pour un et deux robots pour l'instance *BO1-v2*) par deux faits : le premier, évident, est que nous utilisons une méthode de résolution approchée qui pourrait mener à de telles situations car ne garantit pas l'optimalité des solutions trouvées, et le second est que les solutions pour un robot sont rares dans l'espace de recherche.

Par conséquent, nous pouvons déduire à la vue de tous ces résultats, que l'aVNSBT est capable de résoudre des problèmes de tailles différentes, sur lesquels, il a montré de bonnes performances, qui vont jusqu'à la détection des solutions optimales. Son aptitude à fournir de meilleures moyennes sur les exécutions, comparé à SGA, met en exergue sa robustesse à trouver un plus grand nombre de meilleures solutions. Cette stabilité reste un avantage très cherché pour les méthodes approchées,

telles que notre aVNSBT, car ces dernières exercent une recherche stochastique qui, loin d'être optimale, elle doit être optimalement guidée.

3.6 Conclusion

Dans ce chapitre, nous avons proposé une deuxième approche de résolution pour le CHDSP, appelée aVNS, qui se base sur la métaheuristique de type recherche locale : La Recherche à Voisinage Variable. L'approche fait appel à la même méthode de codage des solutions et au même MILP exploités dans le chapitre précédent. Dans un objectif d'amélioration du processus de la recherche des meilleures solutions, nous lui avons intégré une procédure de retour en arrière. Cette deuxième version de l'algorithme est nommée aVNSBT. Pour accroître l'efficacité de la méthode à résoudre ce problème bi-objectif, nous avons adopté différentes structures de voisinage pour accentuer la diversification des voisinages visités. L'algorithme proposé a montré sa capacité à fournir des résultats de qualité, voire des résultats optimaux en termes de temps de cycle, par rapport aux travaux de la littérature. L'aVNS, et l'aVNSBT, comparés au SGA, sont plus performants car ils ont pu relever le défi de la fouille dans l'immense espace de recherche afin de parvenir à des solutions plus prometteuses. Néanmoins, des expérimentations supplémentaires sur des jeux de données aléatoires pourront encore confirmer cette efficacité. Les méthodes que nous avons développées et présentées jusqu'ici permettent de résoudre une version relaxée du CHDSP, celle qui ne prend pas en compte les contraintes spatiales étudiant les risques de collisions entre les robots au cours de leurs déplacements. Afin de traiter le problème dans son intégralité, le chapitre suivant est dédié à proposer une nouvelle modélisation mathématique qui examine et intègre ces contraintes.

Chapitre 4

CHDSP : Gestion des collisions

Dans ce chapitre, nous étudions la variante du CHDSP introduite au chapitre 2, mais cette fois-ci en prenant en compte l'évitement des situations de collisions entre les robots qui circulent sur le même rail. Nous proposons une version étendue du modèle où nous examinons et intégrons les contraintes spatiales permettant la gestion des risques de collisions entre les robots pendant leurs déplacements. Ce modèle est validé en vérifiant a posteriori la faisabilité des meilleures solutions issues de nos deux algorithmes approchés proposés.

Sommaire

Publication

Laajili E., Lamrous S., Manier M-A. et Nicod J-M., "Collision Free Based Model for the Cyclic Multi-Hoist Scheduling Problem", *IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC 2019)*, pp. 873–878, Bari, Italy (October 06-09, 2019).

4.1 Introduction

Dans ce chapitre, nous présentons un modèle mathématique pour le CHDSP où nous considérons les contraintes d'évitement des collisions entre les robots. Dans les deux algorithmes que nous avons développés, SGA et aVNSBT, le MILP employé dans l'évaluation des solutions ne tient pas compte de ces contraintes spatiales.

Nous proposons ici une version étendue de ce modèle qui inclut ces contraintes, de manière à garantir la faisabilité totale de toute solution testée. Un nombre de ces contraintes est inspiré du modèle de Leung et al. [Leung *et al.* 2004] et adapté pour correspondre à notre méthode de codage et à notre choix de variables. En outre, nous avons également ajouté d'autres équations de contraintes qui font face à certaines situations particulières possibles qui n'étaient pas examinées par ces chercheurs.

4.2 Modélisation mathématique

Par souci d'exhaustivité, et pour éviter des renvois du lecteur au chapitre 2, nous présentons ici le modèle complet : la première partie qui concerne le modèle sans les contraintes spatiales, et la deuxième partie où nous formulons les nouvelles contraintes traitant l'évitement des collisions entre les robots. Dans la suite, nous définissons d'abord les notations utilisées pour les données et les variables du problème, qui reprennent en partie celles déjà utilisées dans le MILP initial, puis nous déployons la formulation du modèle mathématique.

4.2.1 Notations

4.2.1.1 Les données

— N : nombre de cuves de la ligne ; la cuve 1 est la station de chargement-déchargement et les cuves de 2 à N sont les cuves de trempe ;

Pour $i, j \in \{1, 2, ..., N\}$:

— $E(i, j)$: mouvement à vide de la cuve i à la cuve j ;

— $d_{i,j}$: durée de $E(i, j)$;

— r_i : durée du mouvement en charge $L(i, i+1)$. $r_i = d_{i,i+1} + c$, où c est une constante de temps requise par le robot pour soulever un porteur de la cuve i, faire une pause si nécessaire au-dessus de la cuve i et laisser le porteur s'égoutter, se stabiliser à l'arrivée de la cuve $i+1$ et descendre le porteur dans la cuve $i+1$. $d_{i,j}$ et r_i sont des données du problème ;

— m_i : temps de trempe minimal dans la cuve i ;

— M_i : temps de trempe maximal dans la cuve i ;

— M : très grand nombre qui représente la valeur de l'infini $(+\infty)$;

— H : nombre de robots associés à la ligne, déduit par la procédure de décodage ;

— h : numéro de robot où $h \in \{1, 2, ..., H\}$; Nous supposons que le robot 1 est affecté au mouvement en charge $L(1, 2)$, et donc au mouvement à vide $E(2, j)$;

— z_h : nombre de mouvements à vide réalisés par le robot h, avec :

$$\sum_{h=1}^{H} z_h = N,$$

— $v_{h,u}$: $u^{\text{ème}}$ mouvement à vide identifié du robot h, $u \in \{1, 2, ..., z_h\}$;

— $Moves_h$: liste ordonnée des mouvements de transport effectués par le robot h, obtenue par la procédure d'affectation ; si le mouvement $L(i, i+1)$ est assuré par le robot h, alors $L(i, i+1) \in \{Moves_h\}$;

— p : cuve d'origine du mouvement de transport $L(p, p+1)$ précédant le mouvement $L(i, i+1)$; si $L(i, i+1) \in \{Moves_h\}$ alors $L(p, p+1) \in \{Moves_h\}$;

— $Moves_k$: liste ordonnée des mouvements de transport effectués par le robot k. Si le robot k assure le mouvement $L(j, j+1)$, alors $L(j, j+1) \in \{Moves_k\}$;

— q : cuve d'origine du mouvement de transport $L(q, q+1)$ précédant le mouvement $L(j, j+1)$; si $L(j, j+1) \in \{Moves_k\}$ alors $L(q, q+1) \in \{Moves_k\}$;

— z_i^h : variable booléenne telle que :

$$z_i^h = \begin{cases} 1 \text{ si le mouvement à vide partant de la cuve i est réalisé par le robot h ;} \\ 0 \text{ sinon.} \end{cases}$$

Les valeurs de z_i^h sont instanciées à l'issue de la procédure de décodage, donc avant l'évaluation des séquences de mouvements S_h par le MILP. Notons par ailleurs que

$$\sum_{i=1}^{N} z_i^h = z_h.$$

4.2.1.2 Les variables

Pour $i, j \in \{1, 2, ..., N\}$, $h \in \{1, 2, ..., H\}$ et $u \in \{1, 2, ..., z_h\}$:

— T : période du cycle (temps de cycle) ;

— $t_{i,j}$: date de début du mouvement à vide $E(i, j)$; pour faciliter son utilisation, nous l'écrivons simplement t_i, en faisant référence à la cuve i du début du mouvement ;

— b_i : variable booléenne, telle que :

$$b_i = \begin{cases} 0 \text{ si un porteur est déposé dans la cuve i et en est retiré durant le} \\ \quad \text{même cycle,} \\ 1 \text{ sinon.} \end{cases}$$

— $a_{h,u}$: variable booléenne, telle que :

$$a_{h,u} = \begin{cases} 0 \text{ si les mouvements à vide consécutifs } v_{h,u} \text{ et } v_{h,u+1} \text{ sont effectués} \\ \quad \text{par le robot h dans le même cycle,} \\ 1 \text{ sinon.} \end{cases}$$

Notons que $v_{h,z_h+1} = v_{h,1}$.

— $y_{i,j}$: variable booléenne, telle que :

$$y_{i,j} = \begin{cases} 1 \text{ si le mouvement en charge L(i-1, i) commence avant L(j-1, j)} \\ 0 \text{ sinon.} \end{cases}$$

4.2.2 Mixed Integer Linear Programming Model

Le modèle qui évalue la faisabilité des séquences de mouvements S_h des robots vis-à-vis toutes les contraintes régnant le problème CHDSP et les ordonnance pour calculer le temps de cycle minimal dans le cas de robots multiples, est formulé comme un programme linéaire en nombres entiers mixtes (MILP), comme suit :

$$Minimiser\ T \tag{4.1}$$

Sous les contraintes :

$$0 \leq t_i \leq T, \quad \forall i \in \{1, 2, ..., N\} \tag{4.2}$$

$$t_2 = r_1 \tag{4.3}$$

$$t_1 + m_{N+1} \leq T \tag{4.4}$$

$$m_i \leq T, \qquad \forall i \in \{1, 2, ..., N\} \tag{4.5}$$

$$(b_i - 1).M \leq t_i - (t_{i+1} - r_i) \leq b_i.M, \qquad \forall i \in \{1, 2, ..., N-1\} \tag{4.6}$$

$$(b_N - 1).M \leq t_N - (t_1 - r_N) \leq b_N.M \tag{4.7}$$

$$m_i - b_i.M \leq t_{i+1} - r_i - t_i \leq M_i + b_i.M \qquad \forall i \in \{1, 2, ..., N-1\} \tag{4.8}$$

$$m_i + (b_i - 1).M \leq T + t_{i+1} - r_i - t_{i+1} \leq M_i + (1 - b_i).M \qquad \forall i \in \{1, 2, ..., N-1\} \tag{4.9}$$

$$m_N - b_N.M \leq t_1 - r_N - t_N \leq M_N + b_N.M \tag{4.10}$$

$$m_N + (b_N - 1).M \leq T + t_1 - r_N - t_N \leq M_N + (1 - b_N).M \tag{4.11}$$

$$\sum_{u=1}^{z_h} a_{h,u} = 1 \qquad \forall h \in \{1, 2, ..., H\} \tag{4.12}$$

$$\text{Si } h = 1, \quad a_{1,u} = 0 \ et \ a_{1,z_1} = 1 \qquad \forall u \in \{1, 2, ..., z_1 - 1\}, \tag{4.13}$$

$$\forall h \in \{1, 2, ..., H\}; \quad \forall u \in \{1, 2, ..., z_h\} :$$

$$t_j + d_{j,i} \leq t_{i+1} - r_i + a_{h,u}.M \tag{4.14}$$

$$t_j + d_{j,i} \leq T + t_{i+1} - r_i + (1 - a_{h,u}).M \tag{4.15}$$

$$\text{avec } v_{h,u} = E(j, i),\ v_{h,u+1} = E(i + 1, l) \ et \ v_{h,z_h+1} = v_{h,1}$$

$$t_{j+1} - r_j - (t_{i+1} - r_i) \leq M.y_{i+1,j+1} \qquad \forall i, j \in \{1, 2, ..., N\}, i \neq j \tag{4.16}$$

$$y_{i,j} + y_{j,i} = 1 \qquad \forall i, j \in \{1, 2, ..., N\}, i \neq j \tag{4.17}$$

$$\forall i, j \in \{1, 2, ..., N\} \text{ où } L(i, i + 1) \in \{Moves_h\} \ et \ L(j, j + 1) \in \{Moves_k\};$$

$$\forall h \in \{1, 2, ..., H\} :$$

$$t_{i+1} + d_{i+1,j} - t_{q+1} - d_{q+1,j} < M.(3 - y_{i+1,j+1} - z^h_{i+1} - \sum_{k=h}^{H} z^k_{j+1}) \qquad k > h; j < i \tag{4.18}$$

$$t_{j+1} + d_{j+1,i} - t_{p+1} - d_{p+1,i} < M.(3 - y_{j+1,i+1} - z^h_{i+1} - \sum_{k=h}^{H} z^k_{j+1}) \qquad k > h; j < i \tag{4.19}$$

$$t_{i+1} + d_{i+1,j} - t_{q+1} - d_{q+1,j} < M.(3 - y_{i+1,j+1} - z^h_{i+1} - \sum_{k=1}^{h} z^k_{j+1}) \qquad k < h; j > i \tag{4.20}$$

$$t_{j+1} + d_{j+1,i} - t_{p+1} - d_{p+1,i} < M.(3 - y_{j+1,i+1} - z^h_{i+1} - \sum_{k=1}^{h} z^k_{j+1}) \qquad k < h; j > i \tag{4.21}$$

$$t_{i+1} + d_{i+1,j} - t_{q+1} - d_{q+1,j} - T < M.(2 - z^h_{i+1} - \sum_{k=h}^{H} z^k_{j+1}) \qquad k > h; j < i \tag{4.22}$$

$$t_{j+1} + d_{j+1,i} - t_{p+1} - d_{p+1,i} - T < M.(2 - z^h_{i+1} - \sum_{k=h}^{H} z^k_{j+1}) \qquad k > h; j < i \tag{4.23}$$

$$t_{i+1} + d_{i+1,j} - t_{q+1} - d_{q+1,j} - T < M.(2 - z^h_{i+1} - \sum_{k=1}^{h} z^k_{j+1}) \qquad k < h; j > i \tag{4.24}$$

$$t_{j+1} + d_{j+1,i} - t_{p+1} - d_{p+1,i} - T < M.(2 - z^h_{i+1} - \sum_{k=1}^{h} z^k_{j+1}) \qquad k < h; j > i \tag{4.25}$$

$$\forall i, j \in \{1, 2, ..., N\} \text{ où } L(i, i+1) \in \{Moves_h\} \text{ et } L(j, j+1) \in \{Moves_k\};$$

$$\forall h \in \{1, 2, ..., H\} :$$

$$t_{j+1} - t_{p+1} - d_{p+1,i} < M.(3 - y_{j+1,i+1} - z^h_{j+1} - \sum_{k=h}^{H} z^k_{i+1}) \qquad k > h; i = j+1 \tag{4.26}$$

$$t_{i+1} - t_{q+1} - d_{q+1,j} < M.(3 - y_{i+1,j+1} - z^h_{j+1} - \sum_{k=1}^{h} z^k_{i+1}) \qquad k < h; i+1 = j \tag{4.27}$$

$$t_{j+1} - t_{p+1} - d_{p+1,i} - T < M.(2 - z^h_{j+1} - \sum_{k=h}^{H} z^k_{i+1}) \qquad k > h; i = j+1 \tag{4.28}$$

$$t_{i+1} - t_{q+1} - d_{q+1,j} - T < M.(2 - z^h_{j+1} - \sum_{k=1}^{h} z^k_{i+1}) \qquad k < h; i+1 = j \tag{4.29}$$

$$b_i \in \{0, 1\} \qquad \forall i \in \{1, 2, ..., N\} \tag{4.30}$$

$$a_{h,u} \in \{0, 1\} \qquad \forall h \in \{1, 2, ..., H\}; \forall u \in \{1, 2, ..., z_h\} \tag{4.31}$$

$$y_{i,j} \in \{0, 1\} \qquad \forall i, j \in \{1, 2, ..., N\} \tag{4.32}$$

La première partie qui reprend le modèle sans les contraintes spatiales, proposé par [Manier & Lamrous 2008] et présenté dans le chapitre 2, va de l'équation (4.1) à l'équation (4.15). Nous rappelons ici brièvement les enjeux de ces contraintes : La fonction objectif est représentée par l'équation (4.1). Elle vise à minimiser uniquement le temps de cycle T, puisque le modèle est utilisé pour évaluer une solution associée à un nombre donné de robots déduit de la phase de décodage. Les contraintes (4.2) à (4.5) définissent et bornent les variables de décision T et t_i et les contraintes (4.6) et (4.7) définissent les variables booléennes b_i. Les contraintes (4.8) à (4.11) assurent le respect des limites des temps de trempe. Enfin, les équations (4.12) à (4.15) sont associées aux ressources de transport et traduisent les contraintes de capacité et les contraintes de disponibilité (pour plus de détails, voir chapitre 2, section 2.3.2).

Les dernières équations du modèle, (4.30), (4.31) et (4.32), ont l'objectif de vérifier les intervalles de définition des variables booléennes $b_i, a_{h,u}$ et $y_{i,j}$.

Dans la suite, nous explicitons toutes les nouvelles équations qui traitent les risques de collision entre les robots. La première série de ces équations a été inspirée des travaux de Leung et al. [Leung *et al.* 2004]. Cependant, nous avons proposé la deuxième série pour compléter le modèle, en vue d'examiner d'autres situations de collisions particulières qui n'étaient pas considérées par ces chercheurs.

4.2.3 Première série des contraintes spatiales

Cette série comprend les équations de (4.16) à (4.25). Les contraintes (4.16) et (4.17) servent à définir les variables booléennes $y_{i,j}$. Les contraintes (4.18) à (4.25) évitent huit situations possibles de collisions qui peuvent se produire entre les robots. Nous considérons une ligne de traitement de surface avec plusieurs robots. Nous précisons que ces ressources de manutention sont numérotées de gauche à droite, en partant de celui situé le plus à gauche (au-dessus du poste de chargement/déchargement). Le robot 1 se voit ainsi attribué l'opération de déchargement de la cuve 1, qui équivaut au mouvement en charge $L(1, 2)$. Il réalise donc le mouvement à vide $E(2, j)$. Chaque robot ajouté à la ligne va se placer à droite de son prédécesseur, jusqu'au dernier robot de la ligne, numéro H, qui sera ainsi le plus proche de la cuve N. Sur une ligne donnée, plus le nombre de robots est important, plus le risque de collisions augmente. Ce risque varie également en fonction de la manière dont les opérations de transport sont affectées aux robots.

Il est évident que des collisions peuvent se produire lorsque les robots placés le plus à droite (ceux ayant les numéros élevés) doivent assurer des mouvements de transport à partir de cuves placées le plus à gauche (ayant les numéros plus bas), ou vice versa. Par conséquent, pour traduire ce scénario mathématiquement, nous considérons deux robots h et k qui effectuent respectivement les deux mouvements en charge $L(i, i + 1)$ et $L(j, j + 1)$. Deux cas ici peuvent apparaître, susceptibles de

générer des collisions : soit $[k > h$ et $j < i]$ ou $[k < h$ et $j > i]$. Dans chacun de ces deux cas, l'opération $L(i, i+1)$ peut être effectuée avant $L(j, j+1)$ $(y_{i+1,j+1} = 1)$, ou inversement $(y_{j+1,i+1} = 1)$. Ainsi, le nombre de scénarios résultants est de 4. Ils sont examinés grâce aux contraintes (4.18), (4.19), (4.20) et (4.21). Nous schématisons ces 4 scénarios possibles sur la figure 4.1.

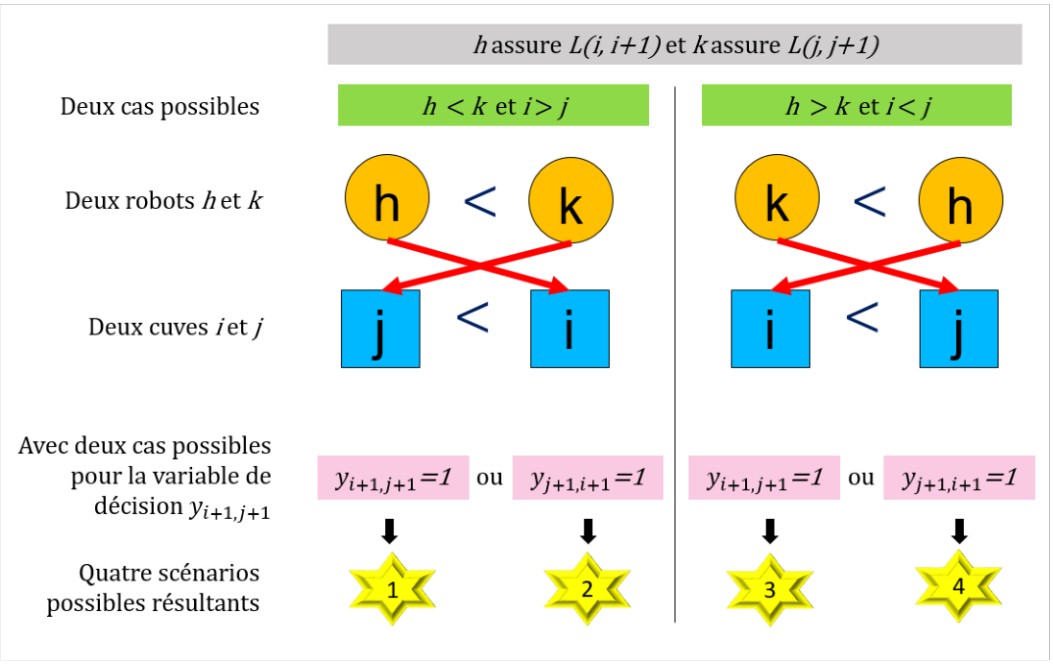

FIGURE 4.1 – Énumération des situations conflictuelles entre deux robots h et k.

Nous donnons sur les figures 4.2 et 4.3 les diagrammes de GANTT représentant les scénarios d'évitement de collisions pour les contraintes (4.18), (4.19), (4.20) et (4.21). La figure 4.2 affiche à gauche le scénario numéro 1 explicité sur la figure précédente 4.1, et à droite le scénario 2. La figure 4.3 illustre à gauche le scénario 3 et à droite le scénario 4. Prenons par exemple, le scénario 1 placé à gauche sur la figure 4.2. Une collision peut se produire entre les deux robots h et k au-dessus de la cuve j, si les deux points bleus se superposent. Ainsi, pour éviter cette collision, il faut assurer une *distance de sécurité* entre les deux robots pendant leurs déplacements. Nous schématisons cette distance de sécurité par la double-flèche rouge. Ainsi, cette collision est évitée si :

$$t_{i+1} + d_{i+1,j} < t_{q+1} + d_{q+1,j}$$

Par conséquent, la zone grisée est interdite pour tout robot $h \leq k$.

Par ailleurs, ces mêmes scénarios peuvent également surgir lorsque les deux mouvements de transport $L(i, i+1)$ et $L(j, j+1)$ sont effectués dans deux cycles consécutifs. Dans ce cas, si le mouvement $L(i, i+1)$ est effectué au cours du cycle suivant, alors sa date de départ est augmentée de la valeur T, ce qui donne les deux

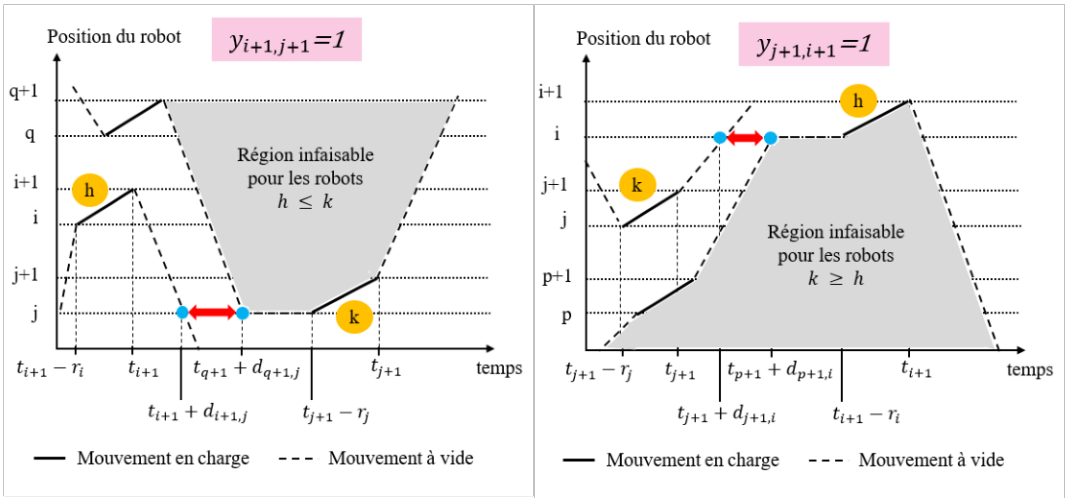

FIGURE 4.2 – Évitement des collisions avec les contraintes (4.18) à gauche et (4.19) à droite.

contraintes (4.23) et (4.25). Dans le cas inverse, si c'est le mouvement $L(j, j+1)$ qui commence au cycle suivant, alors c'est sa date de départ qui doit également être incrémentée de la valeur de T. Ce cas mène aux deux contraintes (4.22) et (4.24). Nous illustrons sur la figure 4.4 le scénario examiné par la contrainte (4.22).

4.2.4 Deuxième série des contraintes spatiales

Cette série contient les équations (4.26) à (4.29). Ces dernières décrivent d'autres scénarios possibles d'évitement des collisions, qui n'ont pas été considérés par Leung

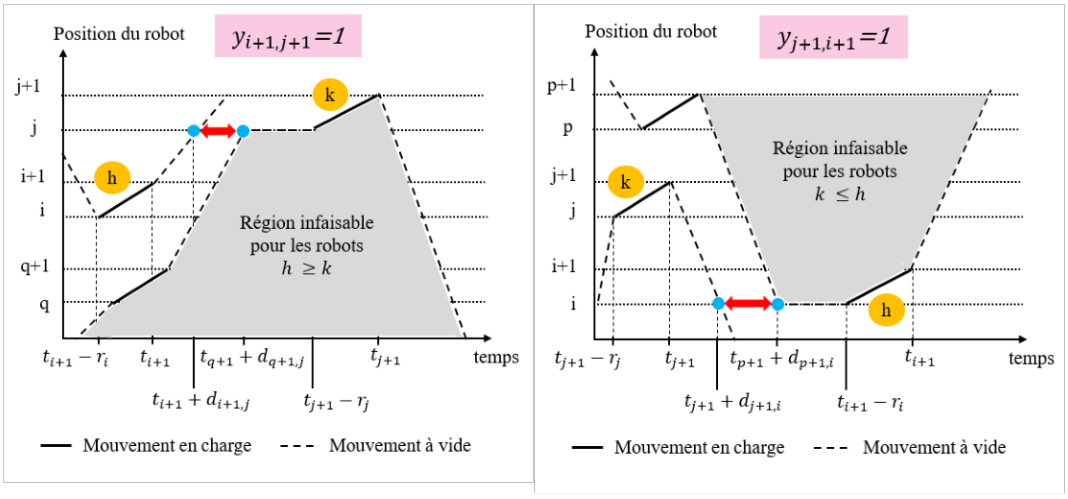

FIGURE 4.3 – Diagrammes de GANTT illustrant les deux scénarios examinés par la contrainte (4.20) à gauche et la contrainte (4.21) à droite.

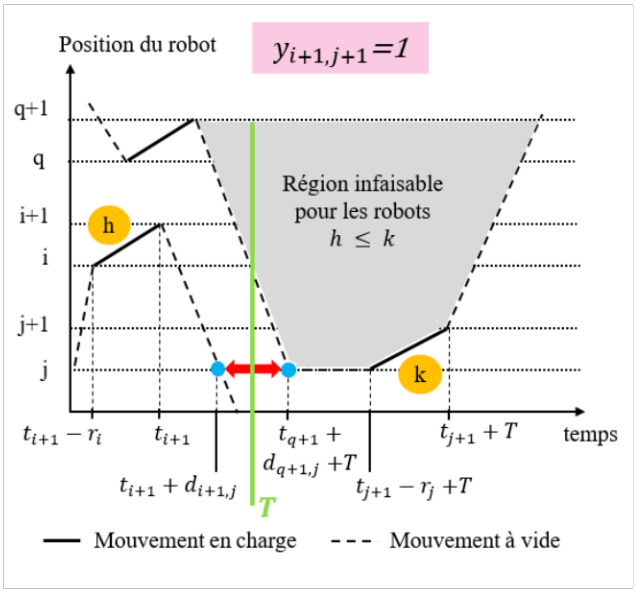

FIGURE 4.4 – Diagramme de GANTT illustrant le scénario examiné par la contrainte (4.22).

et al. [Leung *et al.* 2004]. Il s'agit de 4 situations au total : les deux premières, qui correspondent aux équations (4.26) et (4.27), sont schématisées sur la figure 4.5. Les deux dernières, qui sont traduites par les équations (4.28) et (4.29), évaluent les deux mêmes scénarios précédents mais en considérant la situation de deux cycles consécutifs.

En effet, la première situation qui est formulée par l'équation (4.26), survient lorsque $h > k$, $i = j + 1$, le robot h effectue le transport $L(j, j + 1)$, et le robot k effectue le mouvement $L(i, i+1) = L(j+1, j+2)$. Dans ce cas, une collision doit être évitée au-dessus de la cuve $i = j + 1$. Parallèlement, la contrainte (4.27) est destinée à empêcher une collision dans la situation où $h < k$, $j = i + 1$, le robot h effectue le mouvement $L(j, j + 1)$, et le robot k effectue le mouvement $L(i, i + 1) = L(i, j)$. Comme exemple, nous illustrons sur la figure 4.6 la situation qui correspond à la contrainte (4.26).

En conséquence, le modèle que nous proposons pour le CHDSP, complète le premier modèle de [Manier & Lamrous 2008] présenté dans le chapitre 2 et utilisé dans les deux approches hybrides que nous avons développé, SGA et aVNS (éventuellement aVNSBT). Il examine et intègre les contraintes inhibant les éventuelles collisions qui pourraient se produire entre les robots partageant le même rail. Il améliore également le modèle de Leung et al. [Leung *et al.* 2004], dans le sens où il étend le nombre de situations de collisions considérées par ces chercheurs. Ces situations traduisent les scénarios où un des robots a besoin d'attendre à vide au-dessus d'une cuve avant de réaliser son mouvement de transport à partir de

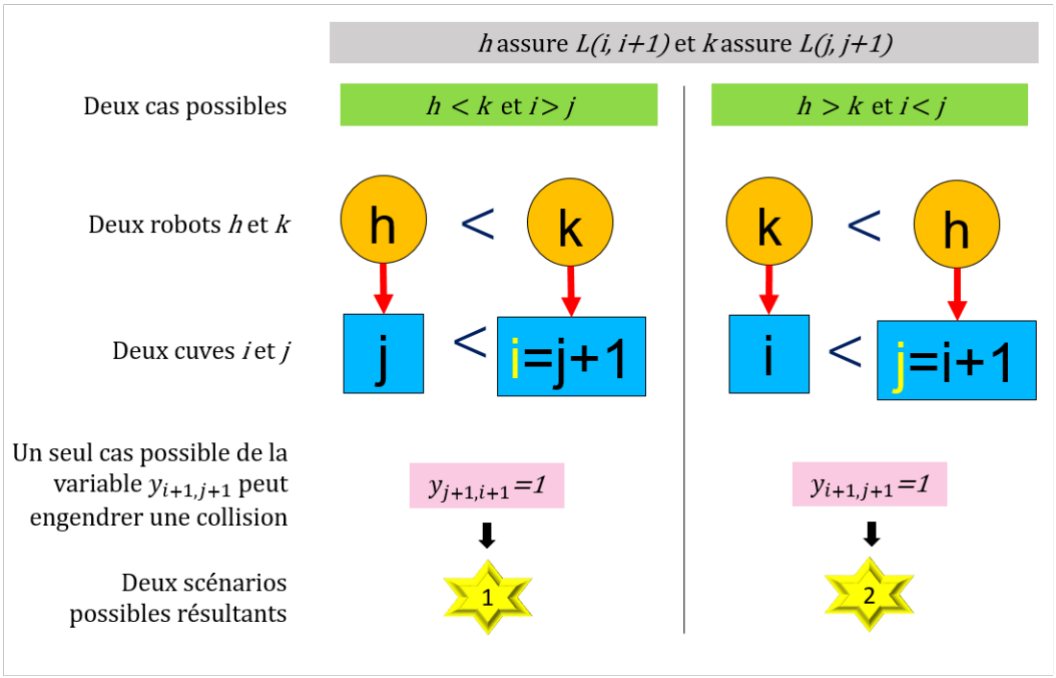

FIGURE 4.5 – Les nouvelles situations conflictuelles entre deux robots h et k.

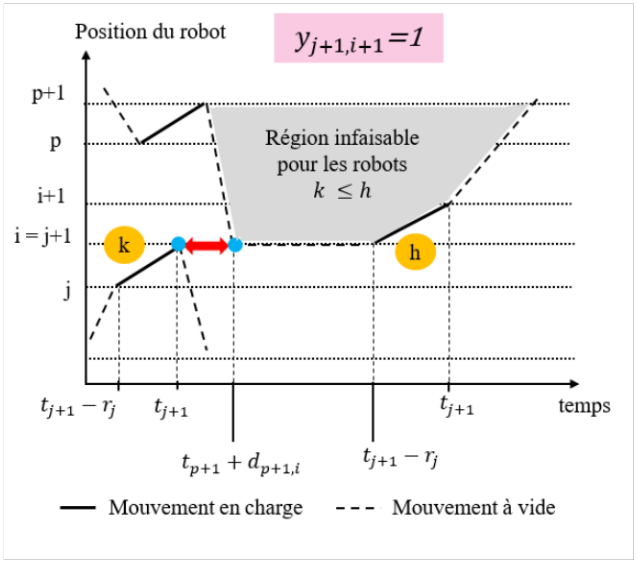

FIGURE 4.6 – Diagramme de GANTT illustrant la contrainte (4.26).

cette cuve, et un autre robot qui vient déposer un porteur de pièces dans la même cuve durant cette durée d'attente du premier robot. La collision peut se produire ainsi au-dessus de cette cuve. La figure 4.6 visualise bien un de ces cas. Avec le modèle de Leung et al. [Leung *et al.* 2004], l'évaluation de séquences de mouvements présentant un de ces cas renvoie une solution faisable avec un temps de cycle $T \neq 0$ (voir l'illustration à gauche de la figure 4.7). Néanmoins, cette solution est en réalité

non faisable car elle présente une collision entre les deux robots au-dessus de la cuve commune de leurs deux mouvements de transport : le premier y attend pour retirer un porteur d'où cette cuve commune est la cuve de départ de son prochain mouvement en charge, alors que le deuxième y arrive pour déposer un porteur donc cette cuve commune est la cuve de destination de son mouvement en charge actuel.

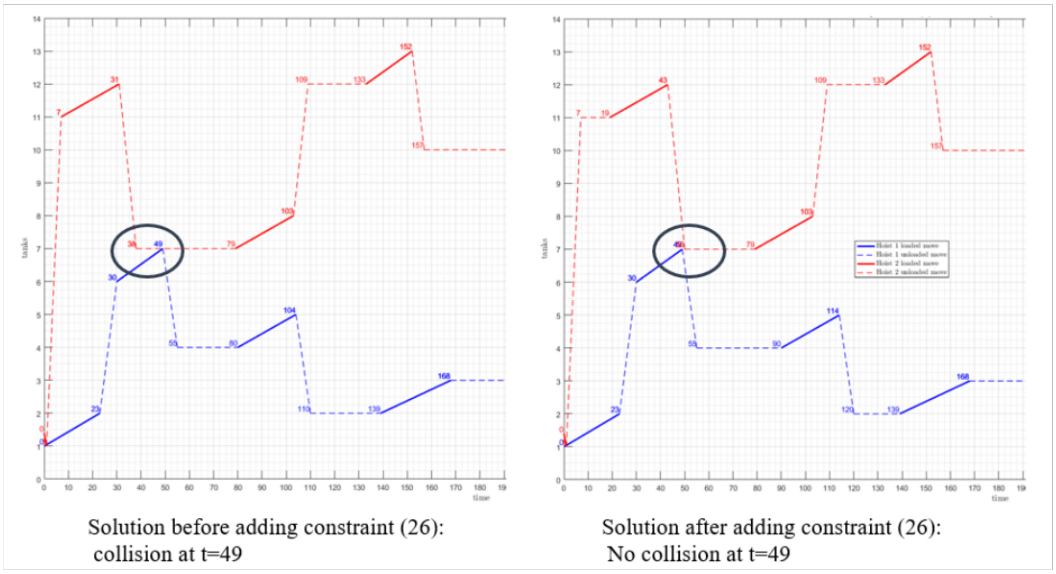

FIGURE 4.7 – Évitement de collision quand un des robots est immobile.

Toutefois, avec notre modèle complet, où la contrainte (4.26) n'est encore pas intégrée, l'évaluation des mêmes séquences de mouvements renvoie la mention non faisable avec un temps de cycle de valeur nul ($T = 0$). En ajoutant cette dernière équation dans notre modèle, les mêmes séquences de mouvements peuvent devenir faisables et le modèle renvoie alors un temps de cycle $T \neq 0$. Ici, la situation précédente de collision ne surgit plus. Elle est évitée et corrigée grâce à l'équation (4.26) de notre modèle. Cette nouvelle situation sans collision est illustrée sur la même figure 4.7 à droite.

4.3 Expérimentation et résultats

4.3.1 Environnement de test

Le modèle étendu proposé a été implémenté sur le logiciel *Matlab* (version R2016a) et expérimenté sur un processeur Intel®Core™i5-6500 CPU @ 3,20 GHz 3,19 GHz, 16 Go de RAM, 64 bits, sous le système d'exploitation Windows. Nous faisons toujours appel à *Hybrid Toolbox*[1] pour la résolution.

1. *A. Bemporad. "Hybrid Toolbox" - User's Guide, 2004. http ://cse.lab.imtlucca.it/ bemporad/hybrid/toolbox.*

4.3.2 Instances testées

Nous avons testé notre modèle sur les trois instances de référence *P&U*, *Ligne1* et *Ligne2* (voir annexes A.1, A.2, et A.3).

4.3.3 Expérimentations

Les tests sont effectués sur les meilleures solutions obtenues pour différents nombres de robots, à l'issue de l'algorithme aVNSBT qui utilisait la première version du MILP ne traitant pas l'évitement des collisions (celle du chapitre 2). Ces solutions sont partiellement faisables car elles satisfont toutes les contraintes sauf celles qui concernent la prévention des collisions. Par conséquent, afin de valider le nouveau modèle proposé intégrant les nouvelles contraintes spatiales, nous l'avons appliqué à ces solutions dans le but d'une vérification *a posteriori*. Notons que cette procédure de vérification ne dépend pas de la méthode initialement utilisée pour générer les solutions partiellement faisables.

Nous présentons sur le tableau 4.1 un extrait de ces résultats. Nous affichons chaque fois les temps de cycle T_i obtenus en testant 7 meilleures solutions initiales (partiellement faisables) pour les trois instances considérées. Les solutions testées ont un nombre de robots $2 \leq H \leq 5$. Le temps de cycle $T1$ correspond à la période de cycle initiale obtenue à l'issue du MILP ne contenant pas les contraintes de collisions. Nous appelons cette version du modèle *V1*. Le temps $T2$ représente le nouveau temps de cycle obtenu à l'issue du nouveau modèle n'intégrant que les contraintes de collision inspirées de [Leung *et al.* 2004] (les contraintes (4.16) à (4.25)). Nous nommons cette version du modèle *V2*. Enfin, le temps $T3$ est le nouveau temps de cycle obtenu après application intégrale de notre modèle étendu, contenant ainsi toutes les nouvelles contraintes spatiales (de (4.16) à (4.29)). Cette dernière version complète du modèle est nommée *V3* . Nous ne testons que des solutions à plusieurs robots, car l'évaluation des contraintes d'interdiction des collisions sur des solutions mono-robots n'apportera aucune modification sur les ordonnancements résultants. De même, nous ne rapportons pas les solutions à plus de cinq robots car nous jugeons que cela n'est plus pertinent pour les instances considérées.

4.3.4 Analyse des performances

Dans un premier temps, nous notons que nous sommes capables de retrouver à nouveau des solutions faisables grâce à notre modèle, plus particulièrement sur ces instances. Cela nous permet donc de valider l'aptitude de notre nouveau modèle à fournir des solutions faisables sans collisions. Dans un deuxième temps, en comparant les résultats fournis sur le tableau 4.1, nous concluons sur l'intérêt de la procédure de vérification d'évitement de collisions appliquée a posteriori. En effet, grâce à cette vérification a posteriori, nous montrons que certaines des meilleures solutions, jugées faisables avec la première version *V1* du modèle ou même avec la deuxième version

TABLE 4.1 – Tests avec les trois versions de modèles sur les meilleures solutions trouvées des trois instances *P&U*, *Ligne1* et *Ligne2*.

Instance	2 robots			3 robots			4 robots			5 robots		
	$T1$	$T2$	$T3$	$T1$	$T2$	$T3$	$T1$	$T2$	$T3$	$T1$	$T2$	$T3$
P&U	251	251	251	179	0	0	151	382	0	151	238	0
	252	0	0	182	0	0	151	398	0	151	264	0
	254	0	0	188	0	0	151	411	0	151	280	0
	255	0	0	188	636	0	151	423	0	151	285	0
	257	0	0	188	349	0	151	579	0	151	303	0
	258	0	0	190	0	0	151	816	0	151	315	0
	264	0	0	190.67	492	0	151	0	0	151	320	0
Ligne1	361	361	361	361	361	399	361	361	0	361	361	402
	361	361	361	361	361	510	361	361	0	361	361	0
	361	367	0	361	361	0	361	365	0	361	364	0
	361	376	0	361	365	0	361	367	0	361	366	0
	361	379	0	361	366	0	361	391	0	361	374.5	0
	361	382	0	361	369	0	361	409	0	361	376	0
	361	0	0	361	0	0	361	0	0	361	0	0
Ligne2	661	661	0	661	661	0	661	661	0	661	661	0
	661	661	0	661	661	0	661	...	...	...	...	...
	661	667	0	661	684	0	661	661	0	661	661	0
	661	712	0	661	698	0	661	661	0	661	661	0
	661	712	0	661	712	0	661	712	0	661	661	0
	661	796	0	661	744	0	661	712	0	661	661	0
	661	0	0	661	0	0	661	0	0	661	661	0

T1 (s) : temps de cycle sans les contraintes d'évitement de collisions ;
T2 (s) : nouveau temps de cycle avec seulement l'ajout des contraintes spatiales inspirées du modèle de [Leung *et al.* 2004] ;
T3 (s) : nouveau temps de cycle avec toutes les nouvelles contraintes spatiales intégrées dans notre modèle.

V2, sont en réalité non réalisables à l'issue de l'exécution de la version complète *V3* du modèle. Cela signifie que ces solutions testées contiennent des situations de collisions qui n'ont pas été traitées avec les deux premières versions du modèle. Sur l'instance *Ligne2*, ce résultat est particulièrement évident. Aucune des solutions testées, de 2 à 5 robots, n'est finalement retenue comme faisable avec le modèle *V3*. Néanmoins, cela n'est pas trop pénalisant puisque nous avions précédemment trouvé la meilleure solution à un robot pour cette instance. Notons également que les nombreuses solutions qui restent faisables en les testant avec la version *V2* deviennent non faisables après application de la version complète du modèle. Cela

s'explique par le fait que ces solutions correspondent aux situations particulières de collisions que nous avons identifiés et formulées grâce aux contraintes (4.26) à (4.29) et qui ne sont pas examinées dans [Leung *et al.* 2004].

Par ailleurs, nous observons également sur le tableau 4.1, que pour une instance donnée et un nombre de robots donné, de nombreuses solutions testées ont le même temps de cycle initial $T1$, ce qui donne une fausse impression de répétition des mêmes solutions. En effet, il s'agit de solutions différentes en termes de codages et de séquences de mouvements associées mais qui résultent en un même temps de cycle. Ainsi, les temps de cycle 151, 361 et 661 sur les trois instances respectives étudiées *P&U*, *Ligne1* et *Ligne2*, proviennent des caractéristiques temporelles de la cuve critique de chaque problème testé, celle ayant le plus grand temps minimal de trempe.

Globalement, au vu de ces résultats, nous distinguons trois cas pour notre modèle complet : dans le premier, la meilleure solution testée obtenue par l'algorithme hybride aVNSBT, reste la même après application de la version *V3* du modèle ($T3 = T1$). Deux exemples à ce cas sont la solution à 2 robots de l'instance de *P&U* et la solution à 2 robots de l'instance *Ligne1* figurant sur les premières lignes. Nous fournissons sur les deux figures 4.8 et 4.9 les diagrammes temporels respectifs de ces deux solutions. Le deuxième cas correspond à la situation où la meilleure solution testée devient non faisable ($T3 = 0$). Nous donnons l'exemple de toutes les solutions testées de l'instance *Ligne2*. Dans le troisième cas, la meilleure solution testée reste faisable après l'application du modèle version *V3*, avec les mêmes séquences de mouvements associées, mais le temps de cycle résultant augmente ($T3 > T1$), ce qui signifie que les dates de transport t_i calculées grâce au modèle ont éventuellement changées. Ce cas correspond par exemple aux solutions de la première ligne de l'instance *Ligne1* pour 3 et 5 robots.

Ces résultats montrent que la recherche des solutions faisables sans collisions n'est pas triviale. Les temps de cycle trouvés après le processus de post-vérification, soulignent le risque important de ne plus trouver aucune solution faisable parmi les anciennes meilleures solutions évaluées faisables à l'issue de la résolution du problème relâché (ne traitant pas les risques de collisions). Ils mettent également en lumière la rareté des solutions totalement réalisables, et cela est vrai pour tous les nombres de robots et pour toutes les instances testées. Par conséquent, pour remédier à ce problème, deux pistes sont à étudier et vérifier : soit conserver et tester beaucoup plus de solutions partiellement faisables, voire toutes, issues de la résolution relâchée du CHDSP, soit employer le nouveau modèle complet (version *V3*) dans la procédure d'évaluation des solutions dans les deux approches hybrides que nous avons développées, SGA et aVNSBT. Cependant, dans ces deux algorithmes hybrides, nous avons déjà identifié que c'est la procédure d'évaluation des solutions faisant appel au MILP qui consomme la plus grande partie du temps de calcul de l'algorithme. Ainsi, un compromis entre les deux propositions serait de tester toutes les solutions possibles avec le MILP initial (version *V1*), et ensuite appliquer les

nouvelles contraintes de collision proposées sur toutes les solutions jugées faisables avec *V1*.

4.4 Conclusion

Dans ce chapitre, nous avons proposé une extension du premier modèle MILP utilisé pour l'évaluation des solutions dans nos deux approches hybrides décrites dans le chapitre 2 et le chapitre 3. Cette extension étudie les situations éventuelles de risques de collisions entre les robots qui partagent la même voie de circulation dans les lignes de galvanoplastie multi-robots. Nous avons modélisé les scénarios identifiés sous la forme de 14 contraintes supplémentaires. Le nouveau modèle complet obtenu est dédié à la phase d'évaluation des solutions dans une approche hybride de résolution du problème CHDSP. Pour évaluer ses performances, nous l'avons testé dans une procédure de post-vérification appliquée sur des meilleures solutions partiellement faisables issues de l'approche aVNSBT. Cette vérification a posteriori permet de tester et garantir l'entière faisabilité de ces solutions.

De futurs recherches pourront se focaliser sur une utilisation plus efficace de ce nouveau modèle au sein des métaheuristiques développées.

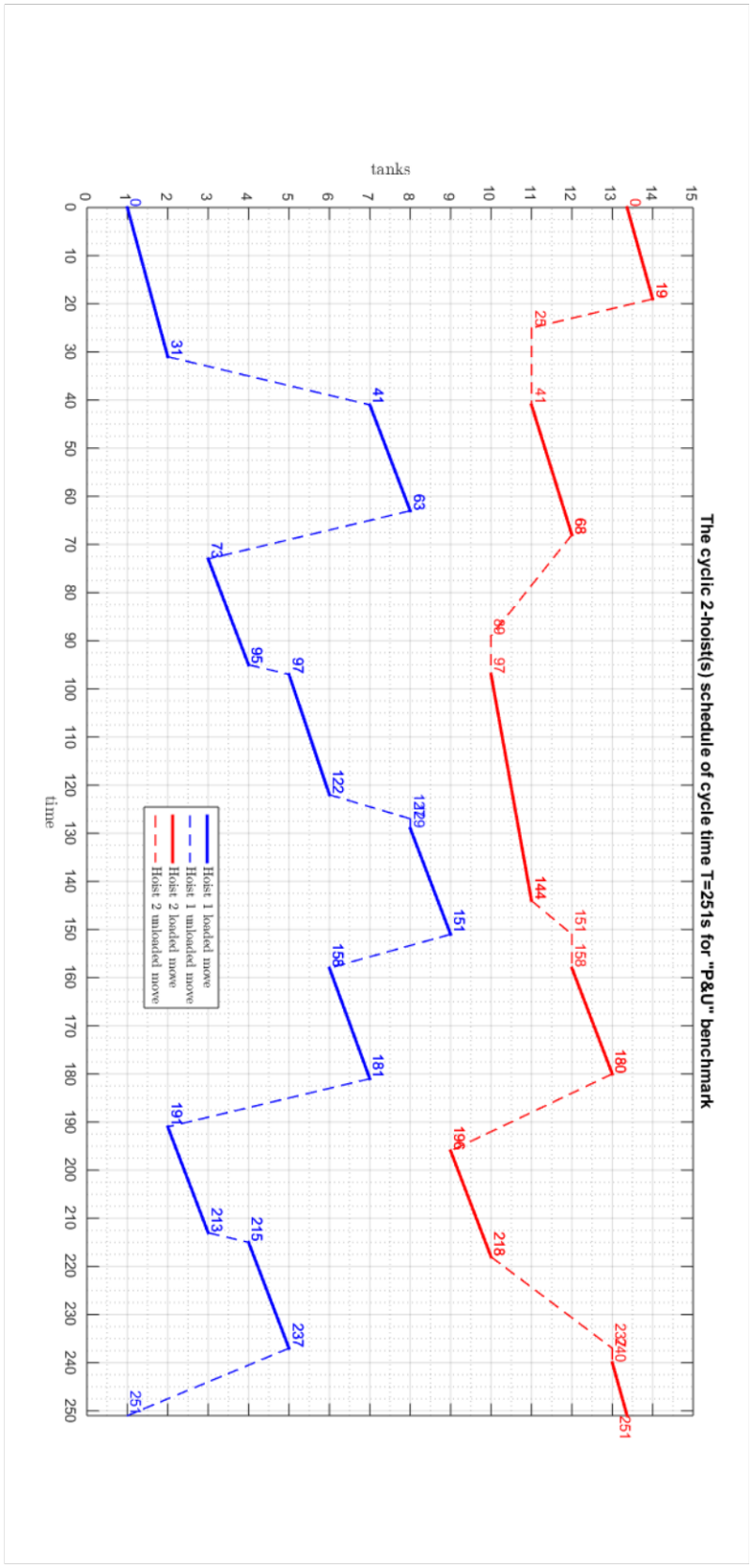

FIGURE 4.8 – Diagramme temporel montrant l'ordonnancement des mouvements des deux robots pour la solution à $T_3 = 251$ s de l'instance de *P&U*.

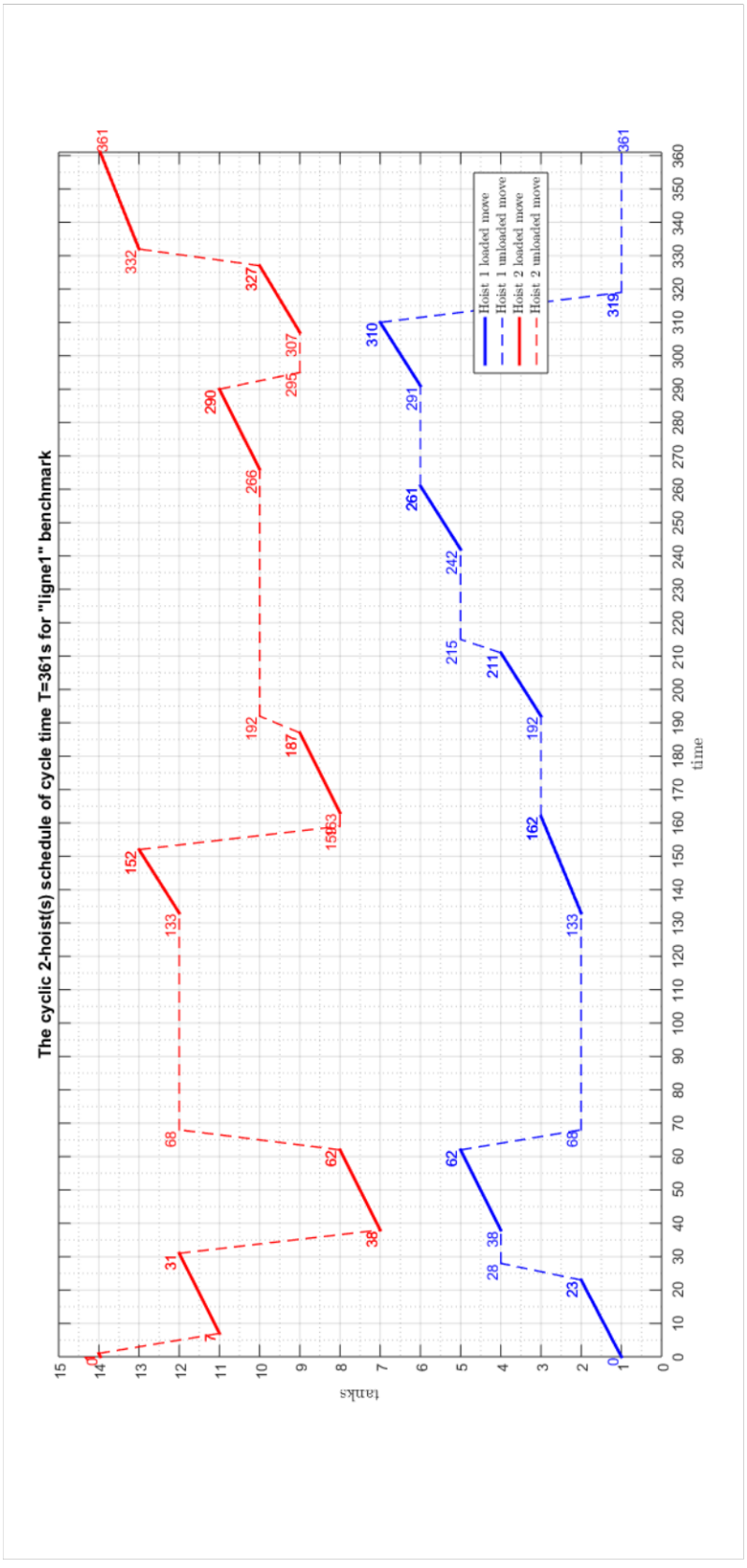

FIGURE 4.9 – Diagramme temporel montrant l'ordonnancement des mouvements des deux robots pour la solution à $T_3 = 361$ s de l'instance *Linge1*.

Conclusion générale et perspectives

Les problèmes d'ordonnancement sont très répandus dans la littérature scientifique et sont étudiés pour toutes sortes d'ateliers (jobshops, flowshops, etc.) et considérés dans une multitude de scénarios : productions cycliques, prédictives, dynamiques, réactives, etc. Cette importance relève de l'enjeu réel de l'ordonnancement sur l'optimisation des opérations de production, pour la réduction des temps et des coûts ainsi que l'amélioration de la productivité des chaînes de production. Les industriels concourent pour atteindre ces objectifs afin de générer plus de bénéfices.

Dans ce contexte, le présent travail de thèse s'est attaqué au problème de dimensionnement et d'ordonnancement cyclique dans les ateliers de traitement de surface à robots multiples (*CHDSP : Cyclic Hoist Design and Scheduling Problem*). Il s'agit d'une variante du problème bien connu de la littérature : le *CHSP, the Cyclic Hoist Scheduling Problem*. Une des principales difficultés consiste à traiter conjointement les deux dimensions de conception et d'ordonnancement dans les lignes multi-robots. Notre objectif consistait à optimiser le couple (temps de cycle, nombre de robots). Sur ce problème bi-objectif, nous espérons que notre contribution pourra conduire à terme à l'élaboration d'un système d'aide à la décision.

Les contributions de cette thèse se situent à plusieurs niveaux :

— Nous avons tout d'abord adopté un codage des solutions basé sur les mouvements à vide des robots que nous avons amélioré pour représenter plus de solutions de l'espace de recherche. Cette amélioration repose sur l'intégration de séparateurs dans le codage initial ce qui mène à obtenir des nouvelles solutions qui n'étaient pas atteignables sans les séparateurs. Pour s'assurer de l'efficacité de cette approche, nous avons élaboré un *algorithme génétique* hybride, qui fait appel à une procédure de décodage basée sur le même principe des mouvements à vide et à un modèle mathématique de type Programme Linéaire en Nombres Entiers Mixtes (MILP) pour l'évaluation des solutions. Les résultats encourageants obtenus des tests accomplis sur des instances de la littérature mettent en évidence la capacité supplémentaire de l'approche améliorée de codage à atteindre l'objectif visé ;

— Dans un deuxième temps, en vue d'optimiser la recherche des meilleures solutions dans l'espace de recherche, nous avons développé une deuxième approche hybride pour la résolution du CHDSP, basée sur la métaheuristique *Recherche à Voisinage Variable*. Cette approche emploie la même procédure de décodage et le même MILP que la précédente. Nous avons conduit une série

de tests afin de choisir les meilleurs réglages des paramètres de l'algorithme, qui a été ensuite testé sur plusieurs instances de la littérature, sur lesquelles il a montré sa performance et robustesse ;

— Enfin, et parce-que le premier MILP utilisé dans l'évaluation des solutions ne tenait pas compte des situations conflictuelles entre les robots pendant leurs déplacements sur le même rail, nous avons étudié les scénarios de collisions qui pourraient se produire entre les robots et nous les avons modélisés en 14 contraintes additionnelles. Ces contraintes spatiales, sont ajoutées à la première version du MILP utilisé, pour offrir un nouveau modèle complet pour le CHDSP. Ce modèle permet l'évaluation de la faisabilité des solutions en vérifiant l'intégralité des contraintes associées au CHDSP. Les tests réalisés, sous forme d'une procédure de vérification a posteriori de meilleures solutions issues de la deuxième approche hybride, sur les benchmarks de la littérature montrent l'efficacité de ce modèle complet.

Nous schématisons nos contributions sur le diagramme de la figure 4.10. Parmi les deux approches hybrides développées, la deuxième basée sur la recherche à voisinage variable et appelée aVNS, a montré de meilleures performances que la première, appelée SGA. En outre, l'aVNS a été amélioré par une procédure de retour en arrière qui donne la version appelée aVNSBT, afin de booster le processus de la recherche des meilleures solutions.

Grâce à ces méthodes, en particulier, l'AVNSBT, nous sommes ainsi capables d'obtenir des couples de solutions : pour chaque nombre de robots, son meilleur temps de cycle, parfois l'optimal. Par conséquent, cette nouvelle approche pourra servir de première brique pour la conception future d'un système d'aide à la décision qui permettra aux décideurs de choisir le nombre de ressources de manutention le plus adapté à leurs chaînes de production, en tenant compte des coûts d'investissement et des enjeux de productivité. Par ailleurs, le modèle complet que nous avons proposé vient compléter la liste des apports de nos travaux. Bien que le problème considéré soit très complexe ce qui nous menés à utiliser une première version du modèle qui relâche les contraintes spatiales, la version étendue que nous avons développée permet de traiter le CHDSP dans son intégralité. Ce nouveau modèle, testé dans le cadre d'une procédure de vérification a posteriori, montre son aptitude à gérer les risques de collisions entre les robots et à ne renvoyer que des solutions faisables : qui respectent l'intégralité des contraintes du CHDSP, notamment les contraintes spatiales.

En conclusion, nous avons obtenu des résultats très encourageants, qui surpassent ceux de la littérature sur le même type de problème. Néanmoins ils restent à confirmer, en particulier :

— Nous devons nous assurer que l'aVNSBT conserve le même niveau de performance pour d'autres instances de plus grande taille ou sur des jeux de données aléatoirement générés. Nous devons ainsi confirmer que son efficacité reste

FIGURE 4.10 – Les axes de contributions de cette thèse sur le CHDSP.

entière en termes de qualité des solutions et de temps de calcul ;

— Il est probable que la vérification à posteriori des solutions partiellement faisables soit chronophage en fonction du nombre des solutions à tester, d'autant plus que sa durée doit être additionnée à la durée de résolution de l'approche hybride qui constitue une première phase de résolution pour le CHDSP multi-robot. En outre, la mise en oeuvre de cette procédure de vérification mise en œuvre pourrait conduire à ne retenir aucune des solutions initialement archivées et donc à ne pas fournir au moins une solution faisable par nombre de robots.

Suite à cette synthèse mettant en lumière les forces et les faiblesses de nos travaux, nous envisageons plusieurs pistes de recherche dans le futur :

— Une première piste intéressante concerne l'extension de notre étude à des cas de lignes plus complexes, à savoir, les lignes disposant de cuves multi-bacs (où la capacité est non unitaire) et/ou multi-fonctions (où il faudra encore gérer les contraintes de réentrance) ;

— Une seconde piste sera d'améliorer davantage la méthode de résolution proposée. Cela pourra se faire en approfondissant les tests sur les meilleurs paramétrages de l'algorithme : par exemple, nous pouvons tester une stratégie différente de choix de la structure de voisinage à appliquer à chaque itération, ou encore

appliquer plus qu'une seule structure de voisinage par itération. Nous pouvons aussi valider les meilleurs paramétrages en testant différentes instances dans la phase de réglages ;

— Une troisième piste sera d'opérer une hybridation entre les deux métaheuristiques que nous avons développées pour améliorer davantage la recherche des meilleures solutions. Les deux méthodes, employées chacune à part, demeurent performantes, aussi le couplage entre les deux pourrait être bénéfique sur leur aptitude résultante. Néanmoins, il reste à déterminer comment les intégrer. Une première idée serait d'intégrer les structures de voisinages utilisées dans l'aVNSBT comme méthodes de mutation des individus de l'algorithme génétique, ou aussi, intégrer la procédure de retour en arrière dans l'algorithme génétique pour lui permettre de revenir sur des voisinages prometteurs ;

— Une quatrième piste intéressante sera d'examiner l'amélioration possible des contraintes spatiales en permettant à un robot de changer de position afin de laisser la place à un autre robot. Cela implique de nouvelles contraintes pour vérifier la faisabilité de ces mouvements atypiques ;

— Une dernière piste qui mérite d'être étudiée sera de déterminer la meilleure manière d'intégrer les contraintes d'évitement de collisions dans le processus de résolution. L'idée est de ne pas se limiter à la procédure de vérification a posteriori, mais de définir une procédure intermédiaire visant un compromis entre :

  (a) une vérification systématique des contraintes spatiales pour toutes les solutions générées tout au long du processus de recherche, en utilisant le modèle complet dans la phase d'évaluation de l'algorithme de résolution. Cette stratégie assurera que les solutions rendues sont entièrement faisables mais reste à savoir si la procédure de recherche est assez puissante pour atteindre ces solutions, ce qui risque d'être très coûteux en temps de calcul ou négatif sur le choix des meilleures solutions voisines ;

  (b) et une vérification a posteriori qui complète le test de la faisabilité des solutions trouvées à l'issue de l'exécution de la méthode de recherche. Cette stratégie paraît moins coûteuse, peut-être, en temps de calcul, mais, comme nous l'avons déjà analysé, elle présente le risque de ne renvoyer aucune solution entièrement faisable parmi toutes les solutions testées.

Bibliographie

[Aarts *et al.* 2003] Emile Aarts, Emile H.L. Aarts et Jan Karel Lenstra. Local search in combinatorial optimization. Princeton University Press, 2003. (Cité en pages 9 et 88.)

[Aarts *et al.* 2005] Emile Aarts, Jan Korst et Wil Michiels. *Simulated annealing.* In Search methodologies, pages 187–210. Springer, 2005. (Cité en page 88.)

[Alsaffar & Suliman 2020] Alaa Jameel Alsaffar et Saad MA Suliman. *Hoist scheduling for an anodizing plant at an extrusion company.* Arab Journal of Basic and Applied Sciences, vol. 27, no. 1, pages 93–104, 2020. (Cité en pages 45 et 50.)

[Anand *et al.* 2015] Ellur Anand, Ramasamy Panneerselvam*et al. Literature review of open shop scheduling problems.* Intelligent Information Management, vol. 7, no. 01, page 33, 2015. (Cité en page 14.)

[Armstrong *et al.* 1994] Ronald Armstrong, Lei Lei et Shanhong Gu. *A bounding scheme for deriving the minimal cycle time of a single-transporter N-stage process with time-window constraints.* European Journal of Operational Research, vol. 78, no. 1, pages 130–140, 1994. (Cité en pages 27, 32, 34, 38, 43 et 50.)

[Armstrong *et al.* 1996] Ronald Armstrong, Shanhong Gu et Lei Lei. *A greedy algorithm to determine the number of transporters in a cyclic electroplating process.* IIE transactions, vol. 28, no. 5, pages 347–355, 1996. (Cité en pages 45, 46, 50 et 56.)

[Baptiste *et al.* 1992] Pierre Baptiste, Bruno Legeard et Christophe Varnier. *Hoist scheduling problem : an approach based on constraint logic programming.* In Robotics & Automation, 1992. Proceedings., 1992 IEEE International Conference on, pages 1139–1144. IEEE, 1992. (Cité en pages 27, 32, 34, 38 et 50.)

[Baptiste *et al.* 1993] Pierre Baptiste, Bruno Legeard, Marie-Ange Manier et Christophe Varnier. *Optimization with constraint logic programming : the hoist scheduling problem solved with various solvers.* Application of Artificial Intelligence in Engineering (AIENG 93), Toulouse, France, vol. 2, pages 599–614, 1993. (Cité en pages 32, 34, 38 et 50.)

[Basnet & Mize 1994] Chuda Basnet et Joe H. Mize. *Scheduling and control of flexible manufacturing systems : a critical review.* International Journal of Computer Integrated Manufacturing, vol. 7, no. 6, pages 340–355, 1994. (Cité en page 15.)

[Bertsimas *et al.* 1993] Dimitris Bertsimas, John Tsitsiklis*et al. Simulated annealing.* Statistical science, vol. 8, no. 1, pages 10–15, 1993. (Cité en page 88.)

[Bilge & Ulusoy 1995] Ümit Bilge et Gündüz Ulusoy. *A time window approach to simultaneous scheduling of machines and material handling system in an FMS.* Operations Research, vol. 43, no. 6, pages 1058–1070, 1995. (Cité en page 75.)

[Biskup *et al.* 2008] Dirk Biskup, Jan Herrmann et Jatinder ND Gupta. *Scheduling identical parallel machines to minimize total tardiness.* International Journal of Production Economics, vol. 115, no. 1, pages 134–142, 2008. (Cité en page 13.)

[Błażewicz *et al.* 1996] Jacek Błażewicz, Wolfgang Domschke et Erwin Pesch. *The job shop scheduling problem : Conventional and new solution techniques.* European journal of operational research, vol. 93, no. 1, pages 1–33, 1996. (Cité en page 14.)

[Bloch & Manier 1999] Christelle Bloch et M-A Manier. *Notation and typology for the hoist scheduling problem.* In IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028), volume 6, pages 475–480. IEEE, 1999. (Cité en pages 24 et 27.)

[Bloch *et al.* 1997] Christelle Bloch, Astrid Bachelu, Christophe Varnier et Pierre Baptiste. *Hoist Scheduling Problem : state-of-the-art.* IFAC Proceedings Volumes, vol. 30, no. 14, pages 127–133, 1997. (Cité en pages 15, 24, 27 et 31.)

[Bloch *et al.* 1999] Christelle Bloch, Christophe Varnier et Pierre Baptiste. *A taboo search combined with a modified shifting bottleneck heuristic for solving a blocking scheduling problem with bounded processing times.* In 15^{th} Conference of the International Federation of Operational Research Societies, Beijing, China, pages 125–126, 1999. (Cité en page 28.)

[Bloch *et al.* 2008] C. Bloch, M.-A. Manier, P. Baptiste et C. Varnier. *Hoist scheduling Problem.* In Pierre Lopez et François Roubellat, editeurs, Production Scheduling, chapitre 8. ISTE - Willey, London, 2008. (Cité en pages 24 et 31.)

[Bonhomme *et al.* 2002] Patrice Bonhomme, Pascal Aygalinc et Soizick Calvez. *Firing instant approach to control time critical systems in multi-product processing.* In Sixth International Workshop on Discrete Event Systems, 2002. Proceedings., pages 97–102. IEEE, 2002. (Cité en pages 38 et 50.)

[Bräysy & Gendreau 2005] Olli Bräysy et Michel Gendreau. *Vehicle routing problem with time windows, Part I : Route construction and local search algorithms.* Transportation science, vol. 39, no. 1, pages 104–118, 2005. (Cité en page 88.)

[Buzacott & Yao 1986] John A. Buzacott et David D. Yao. *Flexible manufacturing systems : a review of analytical models.* Management science, vol. 32, no. 7, pages 890–905, 1986. (Cité en page 15.)

[Candan & Yazgan 2015] Gökçe Candan et Harun Resit Yazgan. *Genetic algorithm parameter optimisation using Taguchi method for a flexible manufacturing system scheduling problem.* International Journal of Production Research, vol. 53, no. 3, pages 897–915, 2015. (Cité en page 15.)

[Carlier & Chretienne 1982] J. Carlier et Ph. Chretienne. *Un domaine très ouvert : les problèmes d'ordonnancement.* RAIRO-Operations Research-Recherche Opérationnelle, vol. 16, no. 3, pages 175–217, 1982. (Cité en page 6.)

[Carlier & Chrétienne 1988] Jacques Carlier et Philippe Chrétienne. Problèmes d'ordonnancement : modélisation, complexité, algorithmes. Masson Paris, 1988. (Cité en pages 11 et 13.)

[Carlier *et al.* 1993] Jacques Carlier, Philippe Chrétienne, P. Villon, J. Erschler et C. Hanen. *Les problèmes d'ordonnancement.* RAIRO. Recherche opérationnelle, vol. 27, no. 1, pages 77–150, 1993. (Cité en pages ix, 6, 8 et 9.)

[Caux & Pierreval 1997] Christophe Caux et Henri Pierreval. *Solving a hoist scheduling problem as a sequencing problem.* IFAC Proceedings Volumes, vol. 30, no. 19, pages 315–319, 1997. (Cité en page 28.)

[Caux *et al.* 1995] C. Caux, G. Fleury, Gourgand M. et Kellert P. *Couplage méthodes d'ordonnancement-simulati on pour l'ordonnancement de systèmes industriels de traitement de surface.* RAIRO-Recherche Opérationnelle, vol. 29, no. 4, pages 391–413, 1995. (Cité en page 28.)

[Charon *et al.* 1996] Irène Charon, Anne Germa et Olivier Hudry. Méthodes d'optimisation combinatoire. Masson Paris, 1996. (Cité en pages 10 et 11.)

[Chaudhry & Khan 2016] Imran Ali Chaudhry et Abid Ali Khan. *A research survey : review of flexible job shop scheduling techniques.* International Transactions in Operational Research, vol. 23, no. 3, pages 551–591, 2016. (Cité en page 14.)

[Che & Chu 2004] A. Che et C. Chu. *Single-track multi-hoist scheduling problem : a collision-free resolution based on a branch-and-bound approach.* International Journal of Production Research, vol. 42, no. 12, pages 2435–2456, 2004. (Cité en pages 39, 42, 45, 50 et 56.)

[Che & Chu 2005] Ada Che et Chengbin Chu. *A polynomial algorithm for no-wait cyclic hoist scheduling in an extended electroplating line.* Operations Research Letters, vol. 33, no. 3, pages 274–284, 2005. (Cité en pages 32, 36, 38 et 50.)

[Che & Chu 2007] Ada Che et Chengbin Chu. *Cyclic hoist scheduling in large real-life electroplating lines.* OR Spectrum, vol. 29, no. 3, pages 445–470, 2007. (Cité en pages 32, 34, 38 et 50.)

[Che & Chu 2008] Ada Che et Chengbin Chu. *Optimal scheduling of material handling devices in a PCB production line : problem formulation and a polynomial algorithm.* Mathematical Problems in Engineering, vol. 2008, 2008. (Cité en pages 39, 41, 45, 50 et 56.)

[Che *et al.* 2001] A. Che, F. Chu et C. Chu. *Un algorithme d'ordonnancement multi-cycliques d'un robot.* In MOSIM, France, 2001. (Cité en page 27.)

[Che *et al.* 2002] Ada Che, Chengbin Chu et Feng Chu. *Multicyclic Hoist Scheduling With Constant Processing Times.* IEEE Transactions on robotics and automation, vol. 18, no. 1, pages 69–80, 2002. (Cité en pages 38 et 50.)

[Che *et al.* 2003] Ada Che, Chengbin Chu et Eugene Levner. *A polynomial algorithm for 2-degree cyclic robot scheduling.* European Journal of Operational Research, vol. 145, no. 1, pages 31–44, 2003. (Cité en page 27.)

[Che *et al.* 2010] Ada Che, Pengyu Yan, Naiding Yang et Chengbin Chu. *Optimal cyclic scheduling of a hoist and multi-type parts with fixed processing times.* International Journal of Production Research, vol. 48, no. 5, pages 1225–1243, 2010. (Cité en pages 32, 34, 38 et 50.)

[Che *et al.* 2011] Ada Che, Zhen Zhou, Chengbin Chu et Haoxun Chen. *Multi-degree cyclic hoist scheduling with time window constraints.* International journal of production research, vol. 49, no. 19, pages 5679–5693, 2011. (Cité en page 27.)

[Che *et al.* 2013] Ada Che, Weidong Lei, Jianguang Feng et Chengbin Chu. *An improved mixed integer programming approach for multi-hoist cyclic scheduling problem.* IEEE Transactions on Automation Science and Engineering, vol. 11, no. 1, pages 302–309, 2013. (Cité en pages 39, 42, 45, 50 et 56.)

[Che *et al.* 2015] Ada Che, Jianguang Feng, Haoxun Chen et Chengbin Chu. *Robust optimization for the cyclic hoist scheduling problem.* European Journal of Operational Research, vol. 240, no. 3, pages 627–636, 2015. (Cité en page 27.)

[Chen *et al.* 1994] H. Chen, C. Chu et J.-M. Proth. *Cyclic scheduling of a hoist with the time window constraint.* Tech. Rep. 2307, INRIA, 1994. (Cité en page 27.)

[Chen *et al.* 1998a] Bo Chen, Chris N. Potts et Gerhard J. Woeginger. *A review of machine scheduling : Complexity, algorithms and approximability.* In Handbook of combinatorial optimization, pages 1493–1641. Springer, 1998. (Cité en page 13.)

[Chen *et al.* 1998b] Haoxun Chen, Chengbin Chu et Jean-Marie Proth. *Cyclic Scheduling of a Hoist with Time Window Constraints.* IEEE Transactions on robotics and automation, vol. 14, no. 1, pages 144–152, 1998. (Cité en pages 27, 32, 34, 38, 42 et 50.)

[Cheng & Sin 1990] T.C.E. Cheng et C.C.S. Sin. *A state-of-the-art review of parallel-machine scheduling research.* European Journal of Operational Research, vol. 47, no. 3, pages 271–292, 1990. (Cité en page 13.)

[Cheng & Smith 1995] Cheng-Chung Cheng et Stephen F. Smith. *A constraint-posting framework for scheduling under complex constraints.* In Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA'95, volume 1, pages 269–280. IEEE, 1995. (Cité en page 29.)

[Cheng *et al.* 2000] T.C. Edwin Cheng, Jatinder N.D. Gupta et Guoqing Wang. *A review of flowshop scheduling research with setup times.* Production and operations management, vol. 9, no. 3, pages 262–282, 2000. (Cité en page 13.)

[Chtourou *et al.* 2013] Sameh Chtourou, Marie-Ange Manier*et al.* *A hybrid algorithm for the cyclic hoist scheduling problem with two transportation resources.*

Computers and Industrial Engineering, vol. 65, no. 3, pages 426–437, 2013. (Cité en pages 39, 40, 45, 50 et 56.)

[Clerc 2010] Maurice Clerc. Particle swarm optimization, volume 93. John Wiley & Sons, 2010. (Cité en page 76.)

[Coello 2005] Carlos A. Coello Coello. *An introduction to evolutionary algorithms and their applications.* In International Symposium and School on Advancex Distributed Systems, pages 425–442. Springer, 2005. (Cité en pages 9 et 76.)

[Collart D. & Denat 1998] S. Collart D. et J-P. Denat. *P-time Petri nets and the Hoist Scheduling Problem.* In IEEE International Conference on Systems, Man and Cybernatics, volume 1, pages 558–563. IEEE, 1998. (Cité en pages 38 et 50.)

[Colorni *et al.* 1991] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo*et al. Distributed optimization by ant colonies.* In Proceedings of the first European conference on artificial life, volume 142, pages 134–142. Paris, France, 1991. (Cité en page 76.)

[Crama & Van de Klundert 1997] Yves Crama et Joris Van de Klundert. *Robotic flowshop scheduling is strongly NP-complete.* Ten Years LNMB, pages 277–286, 1997. (Cité en page 24.)

[Davis 1991] Lawrence Davis. *Handbook of genetic algorithms.* 1991. (Cité en page 76.)

[Dawande *et al.* 2005] Milind Dawande, H. Neil Geismar, Suresh P. Sethi et Chelliah Sriskandarajah. *Sequencing and scheduling in robotic cells : Recent developments.* Journal of Scheduling, vol. 8, no. 5, pages 387–426, 2005. (Cité en page 15.)

[Deb *et al.* 2002] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal et TAMT Meyarivan. *A fast and elitist multiobjective genetic algorithm : NSGA-II.* IEEE transactions on evolutionary computation, vol. 6, no. 2, pages 182–197, 2002. (Cité en pages 77 et 81.)

[Deroussi & Norre 2010] L. Deroussi et S. Norre. *Simultaneous scheduling of machines and vehicles for the flexible job shop problem.* In International conference on metaheuristics and nature inspired computing, pages 1–2. Djerba Island Tunisia, 2010. (Cité en page 75.)

[Dorigo & Blum 2005] Marco Dorigo et Christian Blum. *Ant colony optimization theory : A survey.* Theoretical computer science, vol. 344, no. 2-3, pages 243–278, 2005. (Cité en page 76.)

[Dorigo & Stützle 2019] Marco Dorigo et Thomas Stützle. *Ant colony optimization : overview and recent advances.* Handbook of metaheuristics, pages 311–351, 2019. (Cité en page 76.)

[Dorigo *et al.* 2006] Marco Dorigo, Mauro Birattari et Thomas Stutzle. *Ant colony optimization.* IEEE computational intelligence magazine, vol. 1, no. 4, pages 28–39, 2006. (Cité en page 76.)

[Dorigo 1992] Marco Dorigo. *Optimization, learning and natural algorithms*. Ph. D. Thesis, Politecnico di Milano, 1992. (Cité en page 76.)

[Dowsland & Thompson 2012] Kathryn Anne Dowsland et Jonathan Thompson. *Simulated annealing*. Handbook of natural computing, pages 1623–1655, 2012. (Cité en page 88.)

[Eberhart & Kennedy 1995] Russell Eberhart et James Kennedy. *Particle swarm optimization*. In Proceedings of the IEEE international conference on neural networks, volume 4, pages 1942–1948. Citeseer, 1995. (Cité en page 76.)

[El Amraoui & Elhafsi 2016] Adnen El Amraoui et Mohsen Elhafsi. *An efficient new heuristic for the hoist scheduling problem*. Computers and Operations Research, vol. 67, pages 184–192, 2016. (Cité en page 29.)

[El Amraoui *et al.* 2008] Adnen El Amraoui, Marie-Ange Manier, Abdellah El Moudni et Mohamed Benrejeb. *A mixed linear program for a multi-part cyclic hoist scheduling problem*. International Journal of Sciences and Techniques of Automatic control and computer engineering (IJ-STA), Special issue on CEM, vol. 2, pages 612–623, 2008. (Cité en pages 38 et 50.)

[El Amraoui *et al.* 2009] Adnen El Amraoui, Marie-Ange Manier, Abdellah El Moudni et Mohamed Benrejeb. *Robustness integration in a transport scheduling problem*. International REview of Automatic COntrol (IREACO), vol. 2, no. 4, pages 476–480, 2009. (Cité en pages 38 et 50.)

[El Amraoui *et al.* 2012] Adnen El Amraoui, Marie-Ange Manier, Abdellah El Moudni et Mohamed Benrejeb. *Resolution of the two-part cyclic hoist scheduling problem with bounded processing times in complex lines configuration*. European Journal of Industrial Engineering, vol. 6, no. 4, pages 454–473, 2012. (Cité en pages 38 et 50.)

[El Amraoui *et al.* 2013a] Adnen El Amraoui, Marie-Ange Manier, Abdellah El Moudni et Mohamed Benrejeb. *A genetic algorithm approach for a single hoist scheduling problem with time windows constraints*. Engineering Applications of Artificial Intelligence, vol. 26, no. 7, pages 1761–1771, 2013. (Cité en page 38.)

[El Amraoui *et al.* 2013b] Adnen El Amraoui, Marie-Ange Manier, Abdellah El Moudni et Mohamed Benrejeb. *A linear optimization approach to the heterogeneous r-cyclic hoist scheduling problem*. Computers & Industrial Engineering, vol. 65, no. 3, pages 360–369, 2013. (Cité en page 27.)

[Elmi & Topaloglu 2017] Atabak Elmi et Seyda Topaloglu. *Multi-degree cyclic flow shop robotic cell scheduling problem with multiple robots*. International Journal of Computer Integrated Manufacturing, vol. 30, no. 8, pages 805–821, 2017. (Cité en page 27.)

[Esquirol & Lopez 2001] P. Esquirol et P. Lopez. *Concepts et méthodes de base en ordonnancement de la production*. Ordonnancement de la production, pages 25–51, 2001. (Cité en pages 7 et 12.)

[Esquirol *et al.* 1999] Patrick Esquirol, Pierre Lopez et Pierre Lopez. L'ordonnancement. Economica, 1999. (Cité en page 12.)

[Fargier & Lamothe 2001] Hélène Fargier et Jacques Lamothe. *Handling soft constraints in hoist scheduling problems : the fuzzy approach.* Engineering Applications of Artificial Intelligence, vol. 14, no. 3, pages 387–399, 2001. (Cité en page 29.)

[Feng *et al.* 2015] Jianguang Feng, Ada Che et Chengbin Chu. *Dynamic hoist scheduling problem with multi-capacity reentrant machines : A mixed integer programming approach.* Computers & Industrial Engineering, vol. 87, pages 611–620, 2015. (Cité en page 29.)

[Feng *et al.* 2018] Jianguang Feng, Chengbin Chu et Ada Che. *Cyclic jobshop hoist scheduling with multi-capacity reentrant tanks and time-window constraints.* Computers & Industrial Engineering, 2018. (Cité en pages 32, 33, 38 et 50.)

[Feng 2017] Jianguang Feng. *Modélisation et optimisation des Hoist Scheduling Problems.* PhD thesis, Université Paris-Saclay / Northwestern Polytechnical University (China), 2017. (Cité en page 38.)

[Fleury *et al.* 1996] Gérard Fleury, J.-Y. Goujon, Michel Gourgand et Philippe Lacomme. *A hoist sheduling problem, containing a fixed number of carriers, solved with an opportunistic approach.* In CESA'96 IMACS Multiconference : computational engineering in systems applications, pages 473–478, 1996. (Cité en page 28.)

[Fleury 1995] Gérard Fleury. *Applications de méthodes stochastiques inspirées du recuit simulé à des problèmes d'ordonnancement.* Automatique-productique informatique industrielle, vol. 29, no. 4-5, pages 445–470, 1995. (Cité en page 28.)

[Gao *et al.* 2019] Kaizhou Gao, Zhiguang Cao, Le Zhang, Zhenghua Chen, Yuyan Han et Quanke Pan. *A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems.* IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 4, pages 904–916, 2019. (Cité en page 14.)

[Garey & Johnson 1979] Michael R. Garey et David S. Johnson. Computers and intractabilityty : a guide to the theory of np-completeness, volume 174. freeman San Francisco, 1979. (Cité en page 10.)

[Ge & Yih 1995] Y. Ge et Y. Yih. *Crane scheduling with time windows in circuit board production lines.* The International Journal of Production Research, vol. 33, no. 5, pages 1187–1199, 1995. (Cité en page 29.)

[Gendreau & Potvin 2005] Michel Gendreau et Jean-Yves Potvin. *Tabu search.* In Search methodologies, pages 165–186. Springer, 2005. (Cité en page 88.)

[Glover & Laguna 1998] Fred Glover et Manuel Laguna. *Tabu search.* In Handbook of combinatorial optimization, pages 2093–2229. Springer, 1998. (Cité en page 88.)

[Glover 1989] Fred Glover. *Tabu search—part I.* ORSA Journal on computing, vol. 1, no. 3, pages 190–206, 1989. (Cité en page 88.)

[Glover 1990a] Fred Glover. *Tabu search : A tutorial.* Interfaces, vol. 20, no. 4, pages 74–94, 1990. (Cité en page 88.)

[Glover 1990b] Fred Glover. *Tabu search—part II.* ORSA Journal on computing, vol. 2, no. 1, pages 4–32, 1990. (Cité en pages 9 et 88.)

[Golberg 1989] David E Golberg. *Genetic algorithms in search, optimization, and machine learning.* Addion wesley, vol. 1989, no. 102, page 36, 1989. (Cité en page 76.)

[Goldberg & Holland 1988] David E. Goldberg et John Henry Holland. *Genetic algorithms and machine learning.* Machine learning, vol. 3, no. 2, pages 95–99, 1988. (Cité en page 76.)

[Gonzalez & Sahni 1976] Teofilo Gonzalez et Sartaj Sahni. *Open shop scheduling to minimize finish time.* Journal of the ACM (JACM), vol. 23, no. 4, pages 665–679, 1976. (Cité en page 14.)

[Grötschel *et al.* 1985] Martin Grötschel, Michael Jünger et Gerhard Reinelt. *Facets of the linear ordering polytope.* Mathematical programming, vol. 33, no. 1, pages 43–60, 1985. (Cité en page 41.)

[Gultekin *et al.* 2008] Hakan Gultekin, M. Selim Akturk et Oya Ekin Karasan. *Bicriteria robotic cell scheduling.* Journal of Scheduling, vol. 11, no. 6, pages 457–473, 2008. (Cité en page 15.)

[Gupta & Kyparisis 1987] Sushil K. Gupta et Jerzy Kyparisis. *Single machine scheduling research.* Omega, vol. 15, no. 3, pages 207–227, 1987. (Cité en page 13.)

[Hanen & Munier 1993] Claire Hanen et Alix Munier. *Ordonnancement cyclique d'un robot sur une ligne de galvanoplastie : modeles et algorithmes.* Rapport de recherche, LITP/IBP, vol. 93, 1993. (Cité en pages 38 et 50.)

[Hanen & Munier 1994] Claire Hanen et A. Munier. *Periodic scheduling of several hoists.* In Proceedings of the Fourth International Workshop on Project Management and Scheduling, pages 12–15, 1994. (Cité en pages 39, 42, 45 et 50.)

[Heinrichs & Moll 1997] Udo Heinrichs et Christoph Moll. *On the scheduling of one-dimensional transport systems.* 1997. (Cité en page 41.)

[Hindi & Fleszar 2004] Khalil S. Hindi et Krzysztof Fleszar. *A constraint propagation heuristic for the single-hoist, multiple-products scheduling problem.* Computers and Industrial Engineering, vol. 47, no. 1, pages 91–101, 2004. (Cité en page 29.)

[Hitomi 1996] Katsundo Hitomi. *Manufacturing excellence for 21st century production.* Technovation, vol. 16, no. 1, pages 33–41, 1996. (Cité en page 6.)

[Holland *et al.* 1975] John Henry Holland*et al.* Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1975. (Cité en page 76.)

[Holland *et al.* 1992] John Henry Holland*et al.* Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1992. (Cité en page 76.)

[Holland 1975] John Holland. *Adaptation in natural and artificial systems : an introductory analysis with application to biology.* Control and artificial intelligence, 1975. (Cité en page 76.)

[Holland 1992] John H Holland. *Genetic algorithms.* Scientific american, vol. 267, no. 1, pages 66–73, 1992. (Cité en page 76.)

[Ingber 1993] Lester Ingber. *Simulated annealing : Practice versus theory.* Mathematical and computer modelling, vol. 18, no. 11, pages 29–57, 1993. (Cité en page 88.)

[Jégou *et al.* 2006] David Jégou, Dae-Won Kim, Pierre Baptiste et Kwang H. Lee. *A contract net based intelligent agent system for solving the reactive hoist scheduling problem.* Expert systems with applications, vol. 30, no. 2, pages 156–167, 2006. (Cité en page 29.)

[Jiang & Liu 2007] Yun Jiang et Jiyin Liu. *Multihoist cyclic scheduling with fixed processing and transfer times.* IEEE Transactions on Automation Science and Engineering, vol. 4, no. 3, pages 435–450, 2007. (Cité en pages 45, 50 et 56.)

[Jiang & Liu 2014] Yun Jiang et Jiyin Liu. *A new model and an efficient branch-and-bound solution for cyclic multi-hoist scheduling.* Iie Transactions, vol. 46, no. 3, pages 249–262, 2014. (Cité en pages 39, 42, 45, 50 et 56.)

[Johnson *et al.* 1988] David S Johnson, Christos H Papadimitriou et Mihalis Yannakakis. *How easy is local search ?* Journal of computer and system sciences, vol. 37, no. 1, pages 79–100, 1988. (Cité en page 88.)

[Jullien 1999] Christophe Jullien. *Ordonnancement d'une chaîne de galvanoplastie.* PhD thesis, DEA de Génie industriel, ENSGI-INPG, 1999. (Cité en page 18.)

[Karzanov & Livshits 1978] A.V. Karzanov et E.M. Livshits. *Minimal quantity of operators for serving a homogeneous linear technological process.* Automation & Remote Control, vol. 39, no. 3, pages 445–450, 1978. (Cité en pages 37, 46 et 50.)

[Kats & Levner 1997a] Vladimir Kats et Eugene Levner. *Minimizing the number of robots to meet a given cyclic schedule.* Annals of Operations Research, vol. 69, pages 209–226, 1997. (Cité en pages 46 et 50.)

[Kats & Levner 1997b] Vladimir Kats et Eugene Levner. *A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling.* Operations Research Letters, vol. 21, no. 4, pages 171–179, 1997. (Cité en page 41.)

[Kats & Levner 2002] Vladimir Kats et Eugene Levner. *Cyclic scheduling in a robotic production line.* Journal of Scheduling, vol. 5, no. 1, pages 23–41, 2002. (Cité en pages 45 et 50.)

[Kats & Mikhayletskii 1980] V.B ; Kats et Z.N. Mikhayletskii. *On accurate solution t the problem of optimal scheduling of a cyclic process.* Avtomatika i Telemekhanika, no. 3, pages 187–190, 1980. (Cité en page 41.)

[Kats *et al.* 1999] V. Kats, E. Levner et Meyzin L. *Multiple-Part Cyclic Hoist Scheduling Using a Sieve Method.* IEEE Transactions on robotics and automation, vol. 15, no. 4, pages 704–713, 1999. (Cité en pages 27, 38 et 50.)

[Kats *et al.* 2008] Vladimir Kats, Lei Lei et Eugene Levner. *Minimizing the cycle time of multiple-product processing networks with a fixed operation sequence, setups, and time-window constraints.* European Journal of Operational Research, vol. 187, no. 3, pages 1196–1211, 2008. (Cité en pages 38 et 50.)

[Kats 1982] Vladimir B. Kats. *An exact optimal cyclic scheduling algorithm for multi-operator service of a production line.* Automation & Remote Control, vol. 43, pages 538–542, 1982. (Cité en page 37.)

[Kennedy & Eberhart 1995] James Kennedy et Russell Eberhart. *Particle swarm optimization.* In Proceedings of ICNN'95-international conference on neural networks, volume 4, pages 1942–1948. IEEE, 1995. (Cité en page 76.)

[Kharrat 2012] Samah Kharrat. *Contribution à l'ordonnancement des ateliers de traitement de surface avec deux robots.* PhD thesis, Université de Technologie de Belfort-Montbéliard, 2012. (Cité en pages 37, 45, 50 et 56.)

[Kim *et al.* 2002] Dong-Won Kim, Kyong-Hee Kim, Wooseung Jang et F Frank Chen. *Unrelated parallel machine scheduling with setup times using simulated annealing.* Robotics and Computer-Integrated Manufacturing, vol. 18, no. 3-4, pages 223–231, 2002. (Cité en page 13.)

[Kirkpatrick *et al.* 1983] Scott Kirkpatrick, C. Daniel Gelatt et Mario P. Vecchi. *Optimization by simulated annealing.* science, vol. 220, no. 4598, pages 671–680, 1983. (Cité en pages 9 et 88.)

[Koulamas & Kyparisis 2000] Christos Koulamas et George J. Kyparisis. *Scheduling on uniform parallel machines to minimize maximum lateness.* Operations Research Letters, vol. 26, no. 4, pages 175–179, 2000. (Cité en page 13.)

[Koulamas 2010] Christos Koulamas. *The single-machine total tardiness scheduling problem : review and extensions.* European Journal of Operational Research, vol. 202, no. 1, pages 1–7, 2010. (Cité en page 13.)

[Kuntay *et al.* 2006] Isik Kuntay, Qiang Xu, Korkut Uygun et Yinlun Huang. *Environmentally conscious hoist scheduling for electroplating facilities.* Chemical Engineering Communications, vol. 193, no. 3, pages 273–292, 2006. (Cité en pages 38 et 50.)

[Laajili *et al.* 2019a] Emna Laajili, Sid Lamrous, M.-A. Manier et J.-M. Nicod. *Collision-Free Based Model for the Cyclic Multi-Hoist Scheduling Problem.* In 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), pages 873–878. IEEE, 2019. (Cité en page 45.)

[Laajili *et al.* 2019b] Emna Laajili, Sid Lamrous, Marie-Ange Manier et Jean-Marc Nicod. *Genetic Algorithm Based Approach for the Multi-Hoist Design and Scheduling Problem.* In The International Conference on Industrial Engineering and Systems Management. IEEE, 2019. (Cité en page 45.)

[Lacomme *et al.* 2013] Philippe Lacomme, Mohand Larabi et Nikolay Tchernev. *Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles.* International Journal of Production Economics, vol. 143, no. 1, pages 24–34, 2013. (Cité en page 75.)

[Lacomme 1998] Philippe Lacomme. *Optimisation des systèmes de production : méthodes stochastiques et approche multi-agents.* PhD thesis, Clermont-Ferrand, 1998. (Cité en page 28.)

[Lamothe *et al.* 1996] Jacques Lamothe, C. Thierry et J. Delmas. *A multihoist model for the real time hoist scheduling problem.* In CESA'96 IMACS Multiconference : computational engineering in systems applications, pages 461–466, 1996. (Cité en page 29.)

[Lamothe 1996] J. Lamothe. *Une approche pour l'ordonnancement dynamique d'un atelier de traitement de surface.* PhD thesis, L'École Nationale Supérieure de l'Aéronotique et de l'Espace, 1996. (Cité en page 29.)

[Lamrous & Manier 2006] Sid Lamrous et Marie-ange Manier. *Diversification de populations pour un algorithme génétique appliqué à l'ordonnancement de lignes de traitement de surface.* In 1er workshop Métaheuristiques : Théorie et Applications (META'06). Hammamet, Tunisie, 2-4 novembre 2006. (Cité en page 45.)

[Lamrous & Manier 2008] Sid Lamrous et Marie-ange Manier. *Recherches locales itératives parallèles pour l'ordonnancement d'ateliers de traitement de surface multi-robots.* In Neuvième Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'08), session Bermudes : Ordonnancement, Planification, Transport, pp. 285-286. Clermont-Ferrand, France, 25-27 février 2008. (Cité en page 45.)

[Land & Doig 1960] A.H. Land et A.G. Doig. *An automatic method of solving discrete programming problems.* Econometrica : Journal of the Econometric Society, vol. 28, no. 3, pages 497–520, 1960. (Cité en page 9.)

[Lei & Wang 1989a] Lei Lei et Tzyh-Jong Wang. *On the optimal cyclic schedules of single hoist electroplating processes.* Rutgers University, Reasearch report, pages 89–0006, 1989. (Cité en pages 27, 32, 33 et 50.)

[Lei & Wang 1989b] Lei Lei et Tzyh-Jong Wang. *A proof : the cyclic hoist scheduling problem is NP-complete.* Graduate School of Management, Rutgers University, Working Paper, pages 89–0016, 1989. (Cité en pages 24 et 27.)

[Lei & Wang 1991] Lei Lei et Tzyh-Jong Wang. *The minimum common-cycle algorithm for cyclic scheduling of two material handling hoists with time window constraints.* Management Science, vol. 37, no. 12, pages 1629–1639, 1991. (Cité en pages 39, 40, 41, 45, 50, 56 et 106.)

[Lei & Wang 1994a] L. Lei et T. Wang. *On the optimal cyclic schedules of single hoist electroplating processes.* Rapport technique 2, 1994. (Cité en pages 38 et 50.)

[Lei & Wang 1994b] L. Lei et T.J. Wang. *Determining optimal cyclic hoist schedules in a single-hoist electroplating line.* IIE Transactions, vol. 26, no. 2, pages 25–33, 1994. (Cité en pages 27, 38 et 50.)

[Lei *et al.* 1993] Lei Lei, Ronald Armstrong et Shanhong Gu. *Minimizing the fleet size with dependent time-window and single-track constraints.* Operations Research Letters, vol. 14, no. 2, pages 91–98, 1993. (Cité en pages 45, 46, 50 et 56.)

[Lei *et al.* 2017] Weidong Lei, Hervé Manier, Marié-Ange Manier et Xinping Wang. *A hybrid quantum evolutionary algorithm with improved decoding scheme for a robotic flow shop scheduling problem.* Mathematical Problems in Engineering, vol. 2017, 2017. (Cité en pages 36, 38 et 50.)

[Lei 1993] Lei Lei. *Determining the optimal starting times in a cyclic schedule with a given route.* Computers & operations research, vol. 20, no. 8, pages 807–816, 1993. (Cité en pages 32, 35, 36, 38 et 50.)

[Leung & Levner 2006] Janny MY Leung et Eugene Levner. *An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times.* Operations Research Letters, vol. 34, no. 4, pages 465–472, 2006. (Cité en pages 37, 45, 46, 50 et 56.)

[Leung & Zhang 2003] Janny Leung et Guoqing Zhang. *Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence.* IEEE transactions on robotics and automation, vol. 19, no. 3, pages 480–484, 2003. (Cité en pages 39, 41, 45, 50 et 56.)

[Leung *et al.* 2004] Janny M.Y. Leung, Guoqing Zhang, Xiaoguang Yang, Raymond Mak et Kokin Lam. *Optimal cyclic multi-hoist scheduling : A mixed integer programming approach.* Operations Research, vol. 52, no. 6, pages 965–976, 2004. (Cité en pages 39, 41, 45, 50, 56, 106, 119, 124, 128, 131, 132, 134, 135 et 136.)

[Levner & Kats 1998] E. Levner et V. Kats. *A parametric critical path problem and an application for cyclic scheduling.* Discrete Applied Mathematics, vol. 87, pages 149–158, 1998. (Cité en page 27.)

[Levner *et al.* 1996] Eugene Levner, Vladimir Kats et Chelliah Sriskandarajah. *A geometric algorithm for finding two-unit cyclic schedules in no-wait robotic flowshop.* In Proceedings of the international workshop in intelligent scheduling of robots and fms, wisor, volume 96, pages 101–112, 1996. (Cité en page 27.)

[Levner *et al.* 1997] E. Levner, V. Kats et V. E. Levit. *An improved algorithm for cyclic flowshop scheduling in a robotic cell.* European Journal of Operational Research, pages 500–508, 1997. (Cité en pages 27, 32, 36, 38, 39 et 50.)

[Li & Fung 2013] Xin Li et Richard Y.K. Fung. *A mixed integer linear programming approach for multi-degree cyclic multi-hoist scheduling problems without overlapping.* In 2013 IEEE international conference on automation science and engineering (CASE), pages 274–279. IEEE, 2013. (Cité en pages 27, 45, 50 et 56.)

[Li & Fung 2014] Xin Li et Richard Y.K. Fung. *A mixed integer linear programming solution for single hoist multi-degree cyclic scheduling with reentrance.* Engineering Optimization, vol. 46, no. 5, pages 704–723, 2014. (Cité en page 27.)

[Li & Fung 2017] Xin Li et Richard Y.K. Fung. *Optimal multi-degree cyclic solution of multi-hoist scheduling without overlapping.* IEEE Transactions on Automation Science and Engineering, vol. 14, no. 2, pages 1064–1074, 2017. (Cité en pages 27, 45, 50 et 56.)

[Li *et al.* 2015] Xin Li, Felix T.S. Chan et S.H. Chung. *Optimal multi-degree cyclic scheduling of multiple robots without overlapping in robotic flowshops with parallel machines.* Journal of Manufacturing Systems, vol. 36, pages 62–75, 2015. (Cité en pages 27, 39, 42, 45 et 50.)

[Li *et al.* 2019] Xin Li, Yanchun Pan et Richard YK Fung. *Optimal Scheduling of the Reentrant Multi-Degree Cyclic Multi-Hoist Scheduing Problem.* In 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pages 1319–1323. IEEE, 2019. (Cité en pages 42, 45 et 50.)

[Lim 1997] Joon-Mook Lim. *A genetic algorithm for a single hoist scheduling in the printed-circuit-board electroplating line.* Computers & industrial engineering, vol. 33, no. 3-4, pages 789–792, 1997. (Cité en pages 27, 32, 36, 38 et 50.)

[Liu & Jiang 2005] Jiyin Liu et Yun Jiang. *An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem.* Operations Research, vol. 53, no. 2, pages 313–327, 2005. (Cité en pages 39, 40, 45, 50, 56 et 75.)

[Liu *et al.* 2002] Jiyin Liu, Yun Jiang et Zhili Zhou. *Cyclic scheduling of a single hoist in extended electroplating lines : a comprehensive integer programming solution.* IIE Transactions, vol. 34, no. 10, pages 905–914, 2002. (Cité en pages 32, 33, 34, 38 et 50.)

[Liu *et al.* 2012] Chaowei Liu, Chuanyu Zhao et Qiang Xu. *Integration of electroplating process design and operation for simultaneous productivity maximization, energy saving, and freshwater minimization.* Chemical engineering science, vol. 68, no. 1, pages 202–214, 2012. (Cité en pages 44, 45 et 50.)

[Logendran *et al.* 2007] Rasaratnam Logendran, Brent McDonell et Byran Smucker. *Scheduling unrelated parallel machines with sequence-dependent setups.* Computers and Operations Research, vol. 34, no. 11, pages 3420–3438, 2007. (Cité en page 13.)

[Lopez & Roubellat 2001] Pierre Lopez et François Roubellat. Ordonnancement de la production. Hermès science publications, 2001. (Cité en page 11.)

[Mahalean & Bloch 2004] Manier Hervé Manier Marie-Ange Mahalean Maria et Christelle Bloch. *CHSP : extension of an evolutionary approach to multifunction tanks.* Bulletin de liaison n°30, groupe de travail HSP du GDR CNRS MACS, 2004. (Cité en pages 45 et 50.)

[Mak *et al.* 2000] R. Mak, Y. Wong, J.M.Y. Leung, K. Lam et S.M. Gupta. *The hoist scheduling problem for no-wait production lines—A survey of research.* Systems Engineering and Engineering Management Departement, Chinese University of Hong Kong, Hong Kong, Technical Report SEEM2000-05, 2000. (Cité en page 29.)

[Mangione *et al.* 2003a] Fabien Mangione, Nadia Brauner et Bernard Penz. *Optimal cycles for the robotic balanced no-wait flow shop.* 2003. (Cité en pages 27 et 38.)

[Mangione *et al.* 2003b] Fabien Mangione, Nadia Brauner et Bernard Penz. *Three-tank hoist scheduling problem with unbounded or zero-width processing windows.* 2003. (Cité en pages 38 et 50.)

[Mangione 2003] Fabien Mangione. *Ordonnancement des ateliers de traitement de surface pour une production cyclique et mono-produit.* PhD thesis, Institut National Polytechnique de Grenoble, 2003. (Cité en pages 14, 16 et 20.)

[Manier & Baptiste 1994] M.-A. Manier et P. Baptiste. *Etat de l'art : ordonnancement de robots de manutention en galvanoplastie.* Automatique Productique Informatique Industrielle (A.P.I.I.-RAIRO), vol. 28, no. 1, pages 7–35, 1994. (Cité en pages 2 et 31.)

[Manier & Bloch 2003] Marie-Ange Manier et Christelle Bloch. *A classification for hoist scheduling problems.* International Journal of Flexible Manufacturing Systems, vol. 15, no. 1, pages 37–55, 2003. (Cité en pages 15, 24, 27, 29 et 56.)

[Manier & Lamrous 2006] Marie-ange Manier et Sid Lamrous. *Design and scheduling of electroplating facilities.* In the International Conference on Service Systems and Service Management, volume 2, pages 1114–1119. IEEE, 2006. (Cité en pages 45, 46, 48, 50, 56, 62, 63 et 67.)

[Manier & Lamrous 2008] Marie-Ange Manier et Sid Lamrous. *An evolutionary approach for the design and scheduling of electroplating facilities.* Journal of Mathematical Modelling and Algorithms, vol. 7, no. 2, pages 197–215, 2008. (Cité en pages 39, 43, 45, 46, 48, 50, 56, 59, 72, 73, 75, 76, 81, 82, 83, 92, 128 et 131.)

[Manier *et al.* 2000] Marie-ange Manier, Christophe Varnier et Pierre Baptiste. *Constraint-based model for the cyclic multi-hoists scheduling problem.* Production Planning and Control, vol. 11, no. 3, pages 244–257, 2000. (Cité en pages 24, 39, 43, 45 et 50.)

[Manier 1994] Marie-Ange Manier. *Contribution à l'ordonnancement cyclique du système de manutention d'une ligne de galvanoplastie.* PhD thesis, Besançon, 1994. (Cité en pages 32, 35, 37, 38, 39, 42, 45, 50 et 82.)

[Manne 1960] Alan S. Manne. *On the job-shop scheduling problem.* Operations Research, vol. 8, no. 2, pages 219–223, 1960. (Cité en page 14.)

[Mao *et al.* 2018] Yong-nian Mao, Qiu-hua Tang, Zi-xiang Li et Li-ping Zhang. *Mixed-integer linear programming method for multi-degree and multi-hoist cyclic scheduling with time windows.* Engineering Optimization, vol. 50, no. 11, pages 1978–1995, 2018. (Cité en pages 27, 39, 42, 45, 50 et 56.)

[Mateo & Companys 2006] Manuel Mateo et Ramon Companys. *Hoist scheduling in a chemical line to produce batches with identical sizes of different products.* In Sixième conférence francophone de Modélisation et SIMulation, MOSIM, volume 6, pages 677–684, 2006. (Cité en pages 38 et 50.)

[Mateo & Companys 2007] Manuel Mateo et Ramón Companys. *New computational experiences on the Hoist Scheduling Problem for cyclic manufacturing of different products.* 2007. (Cité en pages 38 et 50.)

[Mateo *et al.* 2000] M. Mateo, R. Companys et J. Bautista. *Bounded cycle time for the cyclic Hoist Scheduling Problem.* In World Conference on Production and Operations Management (Sevilla), 2000. (Cité en pages 38 et 50.)

[Mateo *et al.* 2002] M. Mateo, R. Companys et J. Bautista. *Resolution of graphs with Bounded Cycle Time for the cyclic Hoist Scheduling Problem.* In 8th International Workshop on Project Management an Scheduling, pages 257–260, 2002. (Cité en pages 27, 38 et 50.)

[Mitchell 1998] Melanie Mitchell. An introduction to genetic algorithms. MIT press, 1998. (Cité en page 76.)

[Mitchell 2010] John E. Mitchell. *Branch and cut.* Wiley encyclopedia of operations research and management science, 2010. (Cité en page 9.)

[Mladenović & Hansen 1997] Nenad Mladenović et Pierre Hansen. *Variable neighborhood search.* Computers & operations research, vol. 24, no. 11, pages 1097–1100, 1997. (Cité en page 88.)

[Mokotoff 2001] Ethel Mokotoff. *Parallel machine scheduling problems : A survey.* Asia-Pacific Journal of Operational Research, vol. 18, no. 2, page 193, 2001. (Cité en page 13.)

[Nethercote *et al.* 2007] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck et Guido Tack. *MiniZinc : Towards a standard CP modelling language.* In International Conference on Principles and Practice of Constraint Programming, pages 529–543. Springer, 2007. (Cité en page 35.)

[Ng & Leung 1997] WC Ng et J. Leung. *Determining the optimal move times for a given cyclic schedule of a material handling hoist.* Computers & industrial engineering, vol. 32, no. 3, pages 595–606, 1997. (Cité en pages 27, 32, 38 et 50.)

[Ng 1995] WC Ng. *Determining the optimal number of duplicated process tanks in a single-hoist circuit board production line.* Computers & industrial engineering, vol. 28, no. 4, pages 681–688, 1995. (Cité en pages 38 et 50.)

[Ng 1996] WC Ng. *A branch and bound algorithm for hoist scheduling of a circuit board production line.* International Journal of Flexible Manufacturing Systems, vol. 8, no. 1, pages 45–65, 1996. (Cité en page 38.)

[Papadimitriou 1994] Christos H. Papadimitriou. *Computational Complexity.* Reading, Massachusetts, Addison Welsey, 1994. (Cité en page 11.)

[Paul *et al.* 2007] Henrik J. Paul, Christian Bierwirth et Herbert Kopfer. *A heuristic scheduling procedure for multi-item hoist production lines.* International Journal of Production Economics, vol. 105, no. 1, pages 54–69, 2007. (Cité en page 29.)

[Phillips & Unger 1976] Larry W. Phillips et Philip S. Unger. *Mathematical programming solution of a hoist scheduling program.* AIIE transactions, vol. 8, no. 2, pages 219–225, 1976. (Cité en pages xi, 26, 27, 31, 32, 33, 36, 38, 40, 50, 63, 64, 82 et 106.)

[Pinedo 1996] Michael Pinedo. *Scheduling : Theory, Algorithms, and Systems.* IIE Transactions, vol. 28, no. 8, pages 695–697, 1996. (Cité en page 30.)

[Poli *et al.* 2007] Riccardo Poli, James Kennedy et Tim Blackwell. *Particle swarm optimization.* Swarm intelligence, vol. 1, no. 1, pages 33–57, 2007. (Cité en page 76.)

[Qu *et al.* 2017] Honglin Qu, Sujing Wang et Qiang Xu. *Simultaneous 2D hoist scheduling and production line design for multi-recipe and multi-stage material handling processes.* Chemical Engineering Science, vol. 167, pages 251–264, 2017. (Cité en pages 44, 45 et 50.)

[Ramin *et al.* 2020] Danial Ramin, Egidio Leo, Leonardo Nicolosi, Stefano Spinelli et Alessandro Brusaferri. *A Unified PSO-based method for multi-hoist scheduling in advanced Galvanic plants.* In 2020 7th International Conference on Control, Decision and Information Technologies (CoDIT), volume 1, pages 656–661. IEEE, 2020. (Cité en page 29.)

[Reza Hejazi & Saghafian 2005] S. Reza Hejazi et S. Saghafian. *Flowshop-scheduling problems with makespan criterion : a review.* International Journal of Production Research, vol. 43, no. 14, pages 2895–2929, 2005. (Cité en page 13.)

[Ribas *et al.* 2010] Imma Ribas, Rainer Leisten et Jose M Framiñan. *Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective.* Computers and Operations Research, vol. 37, no. 8, pages 1439–1454, 2010. (Cité en page 14.)

[Riera & Yorke-S. 2002] Daniel Riera et Neil Yorke-S. *An improved hybrid model for the generic hoist scheduling problem.* Annals of Operations Research, vol. 115, no. 1-4, pages 173–191, 2002. (Cité en pages 45, 50 et 75.)

[Rodammer & White 1988] Frederick A. Rodammer et K. Preston White. *A recent survey of production scheduling.* IEEE Transactions on Systems, Man, and Cybernetics, vol. 18, no. 6, pages 841–851, 1988. (Cité en page 23.)

[Rodošek & Wallace 1998] Robert Rodošek et Mark Wallace. *A generic model and hybrid algorithm for hoist scheduling problems.* In International Conference on Principles and Practice of Constraint Programming, pages 385–399. Springer, 1998. (Cité en page 24.)

[Rosse Bloch 1999] Christelle Rosse Bloch. *Contribution à l'ordonnancement dynamique de lignes de traitement de surface.* PhD thesis, Besançon, 1999. (Cité en page 29.)

[Ruiz & Vázquez-Rodríguez 2010] Rubén Ruiz et José Antonio Vázquez-Rodríguez. *The hybrid flow shop scheduling problem.* European journal of operational research, vol. 205, no. 1, pages 1–18, 2010. (Cité en page 14.)

[Sastry *et al.* 2005] Kumara Sastry, David Goldberg et Graham Kendall. *Genetic algorithms.* In Search methodologies, pages 97–125. Springer, 2005. (Cité en page 76.)

[Selman & Gomes 2006] Bart Selman et Carla P Gomes. *Hill-climbing search.* Encyclopedia of cognitive science, vol. 81, page 82, 2006. (Cité en page 88.)

[Serafini & Ukovich 1989] Paolo Serafini et Walter Ukovich. *A mathematical model for periodic scheduling problems.* SIAM Journal on Discrete Mathematics, vol. 2, no. 4, pages 550–581, 1989. (Cité en page 24.)

[Shapiro & Nuttle 1988] Gerald W. Shapiro et Henry LW. Nuttle. *Hoist scheduling for a PCB electroplating facility.* IIE transactions, vol. 20, no. 2, pages 157–167, 1988. (Cité en pages 24, 26, 30, 31, 32, 33, 34, 35, 38, 50, 82 et 106.)

[Shapiro 1985a] Gerald W. Shapiro. *Hoist scheduling for a PCB Electroplating Facility.* Program in operation research, Graduate Faculty of North Carolina State University, 1985. (Cité en page 24.)

[Shapiro 1985b] Gerald W. Shapiro. *Hoist scheduling for a PCB electroplating facility.* PhD thesis, Graduate Faculty of North Carolina State University, 1985. (Cité en pages 37, 45 et 50.)

[Sivanandam & Deepa 2008] SN Sivanandam et SN Deepa. *Genetic algorithms.* In Introduction to genetic algorithms, pages 15–37. Springer, 2008. (Cité en page 76.)

[Słowinski 1981] Roman Słowinski. *Multiobjective network scheduling with efficient use of renewable and nonrenewable resources.* European Journal of Operational Research, vol. 7, no. 3, pages 265–273, 1981. (Cité en page 7.)

[Solnon 2010] Christine Solnon. *Résolution de problèmes combinatoires et optimisation par colonies de fourmis.* Université Lyon, vol. 1, 2010. (Cité en page 10.)

[Song *et al.* 1993] W. Song, Z. B. Zabinsky et R. L. Storch. *An algorithm for scheduling a chemical processing tank line.* Production Planning & Control : The Management of Operations, vol. 4, no. 4, pages 323–332, 1993. (Cité en pages 27, 32, 38 et 50.)

[Song *et al.* 1995] Wenwei Song, Richard L Storch et Zelda B Zabinsky. *An example for scheduling a chemical processing tank line.* In Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA'95, volume 1, pages 475–482. IEEE, 1995. (Cité en pages 27, 38 et 50.)

[Spacek *et al.* 1999] Pavel Spacek, Marie-Ange Manier et Abdellah El Moudni. *Control of an electroplating line in the max and min algebras.* International Journal of Systems Science, vol. 30, no. 7, pages 759–778, 1999. (Cité en page 29.)

[Subai *et al.* 2003] Corinne Subai, Eric Niel et Pierre Baptiste. *Vers un pilotage propre des lignes de traitement de surface.* In the 4ème Conférence Francophone de MOdélidation et SIMulation, "Organisation et conduite d'activités dans l'industrie et les services". MOSIM'03, 2003. (Cité en pages 38 et 50.)

[Subai *et al.* 2006] Corinne Subai, Pierre Baptiste et Eric Niel. *Scheduling issues for environmentally responsible manufacturing : The case of hoist scheduling in an electroplating line.* International Journal of Production Economics, vol. 99, no. 1-2, pages 74–87, 2006. (Cité en pages 38 et 50.)

[Sun *et al.* 1994] T_C. Sun, K.K. Lai, Kokin Lam et K.P. So. *A study of heuristics for bidirectional multi-hoist production scheduling systems.* International Journal of Production Economics, vol. 33, no. 1-3, pages 207–214, 1994. (Cité en pages 42 et 45.)

[Tahar *et al.* 2006] Djamel Nait Tahar, Farouk Yalaoui, Chengbin Chu et Lionel Amodeo. *A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times.* International journal of production economics, vol. 99, no. 1-2, pages 63–73, 2006. (Cité en page 13.)

[Tangour 2007] Fatma Tangour. *Ordonnancement dynamique dans les industries agroalimentaires.* PhD thesis, École Centrale de Lille et École Nationale d'Ingénieurs de Tunis, 2007. (Cité en page 10.)

[Thesen & Lei 1990] Arne Thesen et Lei Lei. *An expert scheduling system for material handling hoists.* Journal of Manufacturing Systems, vol. 9, no. 3, pages 247–252, 1990. (Cité en page 29.)

[Thomas 1983] Müller Thomas. *Automated Guided Vehicles.* IFS (Publications) Ltd./Springer-Verlag, UK/Berlin, 1983. (Cité en page 15.)

[Tsamardinos *et al.* 2006] Ioannis Tsamardinos, Laura E Brown et Constantin F Aliferis. *The max-min hill-climbing Bayesian network structure learning algorithm.* Machine learning, vol. 65, no. 1, pages 31–78, 2006. (Cité en page 88.)

[Van Laarhoven & Aarts 1987] Peter JM Van Laarhoven et Emile HL Aarts. *Simulated annealing.* In Simulated annealing : Theory and applications, pages 7–15. Springer, 1987. (Cité en page 88.)

[Varnier & Baptiste 1995] Christophe Varnier et Pierre Baptiste. *A CLP approach for finding a transition schedule between two cyclic mono-product productions in electroplating facilities.* In International Conference on Industrial Engineering and Production Management, pages 372–381, 1995. (Cité en page 28.)

[Varnier *et al.* 1997] Christophe Varnier, Astrid Bachelu et Pierre Baptiste. *Resolution of the cyclic multi-hoists scheduling problem with overlapping partitions.* INFOR : Information Systems and Operational Research, vol. 35, no. 4, pages 309–324, 1997. (Cité en pages 39, 42, 45 et 50.)

[Varnier 1996] Christophe Varnier. *Extensions du "Hoist Scheduling Problem" cyclique. Résolution basée sur un traitement des contraintes disjonctives en programmation en logique avec contraintes.* PhD thesis, Besançon, 1996. (Cité en page 28.)

[Wallace & Yorke-S. 2020] Mark Wallace et Neil Yorke-S. *A new constraint programming model and solving for the cyclic hoist scheduling problem.* Constraints, pages 1–19, 2020. (Cité en pages 35 et 50.)

[Xu & Huang 2004] Qiang Xu et Yinlun Huang. *Graph-assisted cyclic hoist scheduling for environmentally benign electroplating.* Industrial & engineering chemistry research, vol. 43, no. 26, pages 8307–8316, 2004. (Cité en pages 38, 44 et 50.)

[Yan *et al.* 2016] Pengyu Yan, Guanhua Wang, Ada Che et Yaoyiran Li. *Hybrid discrete differential evolution algorithm for biobjective cyclic hoist scheduling with reentrance.* Computers and Operations Research, vol. 76, pages 155–166, 2016. (Cité en pages 38 et 50.)

[Yang *et al.* 2001] Guangwen Yang, D.P. Ju et W.M. Zheng. *Solving multiple hoist scheduling problems by use of simulated annealing.* Journal of Software, vol. 12, no. 1, pages 11–17, 2001. (Cité en pages 39, 43, 45, 50 et 56.)

[Yeh *et al.* 2015] Wei-Chang Yeh, Mei-Chi Chuang et Wen-Chiung Lee. *Uniform parallel machine scheduling with resource consumption constraint.* Applied Mathematical Modelling, vol. 39, no. 8, pages 2131–2138, 2015. (Cité en page 13.)

[Yenisey & Yagmahan 2014] Mehmet Mutlu Yenisey et Betul Yagmahan. *Multi-objective permutation flow shop scheduling problem : Literature review, classification and current trends.* Omega, vol. 45, pages 119–135, 2014. (Cité en page 13.)

[Yih & Chiu 1993] Yuehwern Yih et Chanshing Chiu. *Incremental Learning for Hoist Scheduling Problems in Circuit Board Electroplating Lines.* ASME-PUBLICATIONS-PED, vol. 65, pages 119–119, 1993. (Cité en page 29.)

[Yih *et al.* 1993] Yuehwern Yih, Ting-Peng Liang et Herbert Moskowttz. *Robot scheduling in a circuit board production line : a hybrid OR/ANN approach.* IIE transactions, vol. 25, no. 2, pages 26–33, 1993. (Cité en page 29.)

[Yih 1994] Yuehwern Yih. *An algorithm for hoist scheduling problems.* The International Journal of Production Research, vol. 32, no. 3, pages 501–516, 1994. (Cité en page 29.)

[Zhang *et al.* 2012] Qiao Zhang, Hervé Manier et M-A Manier. *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, pages 1713–1723, 2012. (Cité en page 75.)

[Zhang *et al.* 2014] Qiao Zhang, Hervé Manier, Marie-Ange Manier et Christelle Bloch. *Heuristics for predictive hoist scheduling problems.* European Journal of Industrial Engineering, vol. 8, no. 5, pages 695–715, 2014. (Cité en page 75.)

[Zhang *et al.* 2019] Jian Zhang, Guofu Ding, Yisheng Zou, Shengfeng Qin et Jianlin Fu. *Review of job shop scheduling research and its new perspectives under Industry 4.0.* Journal of Intelligent Manufacturing, vol. 30, no. 4, pages 1809–1830, 2019. (Cité en page 14.)

[Zhang 2012] Qiao Zhang. *Contribution à l'ordonnancement d'ateliers avec ressources de transports.* PhD thesis, Université de Technologie de Belfort-Montbéliard, 2012. (Cité en page 19.)

[Zhao *et al.* 2013] Chuanyu Zhao, Jie Fu et Qiang Xu. *Production-ratio oriented optimization for multi-recipe material handling via simultaneous hoist scheduling and production line arrangement.* Computers and Chemical Engineering, vol. 50, pages 28–38, 2013. (Cité en pages 44, 45 et 50.)

[Zhou & Li 2003] Zhili Zhou et Ling Li. *Single hoist cyclic scheduling with multiple tanks : a material handling solution.* Computers & Operations Research, vol. 30, no. 6, pages 811–819, 2003. (Cité en pages 27, 32, 33, 38 et 50.)

[Zhou & Li 2009] Zhili Zhou et Ling Li. *A solution for cyclic scheduling of multi-hoists without overlapping.* Annals of Operations Research, vol. 168, no. 1, pages 5–21, 2009. (Cité en pages 39, 41, 45 et 50.)

[Zhou & Liu 2008] Zhili Zhou et Jiyin Liu. *A heuristic algorithm for the two-hoist cyclic scheduling problem with overlapping hoist coverage ranges.* IIE Transactions, vol. 40, no. 8, pages 782–794, 2008. (Cité en pages 39, 40, 41, 45, 50 et 56.)

Annexes

A.1 Tables des données de l'instance "P&U"

TABLE A1.1 – Bornes temporelles de trempe m_i et M_i et temps de transport r_i

Cuve i	m_i	M_i	r_i
1	120		31
2	150	200	22
3	90	120	22
4	120	180	22
5	90	125	25
6	30	40	23
7	60	120	22
8	60	120	22
9	45	75	22
10	130	∞	47
11	120	∞	27
12	90	120	22
13	30	60	30
1	0	∞	

TABLE A1.2 – Temps de déplacement à vide $d_{i,j}$ des robots

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	0	11	14	16	14	19	22	24	26	29	6	8	10
2	11	0	2	5	2	8	10	13	15	17	10	3	1
3	14	2	0	2	0	5	8	10	13	15	12	6	3
4	16	5	2	0	2	3	5	8	10	13	15	8	6
5	14	2	0	2	0	5	8	10	13	15	12	6	3
6	19	8	5	3	5	0	3	5	7	10	18	11	9
7	22	10	8	5	8	3	0	2	5	7	20	14	11
8	24	13	10	8	10	5	2	0	2	5	23	16	14
9	26	15	13	10	13	7	5	2	0	2	25	19	16
10	29	17	15	13	15	10	7	5	2	0	27	21	19
11	6	10	12	15	12	18	20	23	25	27	0	7	9
12	8	3	6	8	6	11	14	16	19	21	7	0	2
13	10	1	3	6	3	9	11	14	16	19	9	2	0

A.2 Tables des données de l'instance "ligne1"

TABLE A2.1 – Bornes temporelles de trempe m_i et M_i et temps de transport r_i

Cuve i	m_i	M_i	r_i
1	360	∞	23
2	60	120	29
3	30	90	19
4	120	240	24
5	180	240	19
6	30	90	19
7	30	90	24
8	60	120	24
9	60	120	20
10	300	420	24
11	60	120	24
12	60	120	19
13	120	180	30
1	0	∞	23

TABLE A2.2 – Temps de déplacement à vide $d_{i,j}$ des robots

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	0	13	12	11	10	9	9	8	8	6	6	5	5
2	13	0	4	5	6	7	7	8	9	10	10	11	12
3	12	4	0	4	5	6	6	7	8	9	9	10	11
4	11	5	4	0	4	5	6	6	7	8	9	10	10
5	10	6	5	4	0	4	5	6	6	7	8	9	9
6	9	7	6	5	4	0	4	5	5	6	7	8	8
7	9	7	6	6	5	4	0	4	4	5	6	7	8
8	8	8	7	6	6	5	4	0	4	4	5	6	7
9	8	9	8	7	6	5	4	4	0	5	5	6	7
10	6	10	9	8	7	6	5	4	5	0	4	5	5
11	6	10	9	9	8	7	6	5	5	4	0	4	5
12	5	11	10	10	9	8	7	6	6	5	4	0	4
13	5	12	11	10	9	8	8	7	7	5	5	4	0

A.3 Tables des données de l'instance "ligne2"

TABLE A3.1 – Bornes temporelles de trempe m_i et M_i et temps de transport r_i

Cuve i	m_i	M_i	r_i
1	300	∞	15
2	180	300	24
3	60	120	14
4	60	120	25
5	180	240	29
6	60	120	19
7	30	120	20
8	60	120	24
9	60	120	20
10	180	300	25
11	60	180	19
12	60	120	20
13	60	150	25
14	660	720	18
15	240	∞	35
1	0	0	

TABLE A3.2 – Temps de déplacement à vide $d_{i,j}$ des robots

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	0	5	5	6	8	9	10	12	13	15	16	18	19	21	25
2	5	0	4	5	7	8	9	11	12	14	15	16	18	19	23
3	5	4	0	4	6	7	7	10	11	12	14	15	17	18	22
4	6	5	4	0	5	6	6	8	9	11	13	14	15	17	21
5	8	7	6	5	0	4	5	7	8	10	11	12	14	15	20
6	9	8	7	6	4	0	4	6	7	8	10	11	13	14	18
7	10	9	7	6	5	4	0	5	6	7	9	10	12	13	17
8	12	11	10	8	7	6	5	0	4	6	7	8	10	11	16
9	13	12	11	9	8	7	6	4	0	5	6	7	9	10	14
10	15	14	12	11	10	8	7	6	5	0	5	5	7	8	13
11	16	15	14	13	11	10	9	7	6	5	0	4	6	7	11
12	18	16	15	14	12	11	10	8	7	5	4	0	5	6	10
13	19	18	17	15	14	13	12	10	9	7	6	5	0	5	9
14	21	19	18	17	15	14	13	11	10	8	7	6	5	0	8
15	25	23	22	21	20	18	17	16	14	13	11	10	9	8	0

A.4 Tables des données de l'instance "Black Oxide 1"

TABLE A4.1 – Bornes temporelles de trempe m_i et M_i et temps de transport r_i

Cuve i	m_i	M_i	r_i
1	240	∞	25.9
2	180	240	18.6
3	60	90	12.4
4	60	240	13
5	60	90	18.7
6	60	120	12.6
7	120	180	18.6
8	60	90	13.1
9	120	145	18.8
10	60	90	12.7
11	90	120	20.6
12	240	420	14.6
1	0	0	

TABLE A4.2 – Temps de déplacement à vide $d_{i,j}$ des robots

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	14.3	13.3	12.5	11.1	10	9	8	6.4	5.2	4.1	3
2	14.3	0	1	1.8	3.2	4.3	5.3	6.4	7.9	9.1	10.2	12
3	13.3	1	0	0.8	2.2	3.3	4.3	5.3	6.9	8.1	9.2	11
4	12.5	1.8	0.8	0	1.4	2.5	3.5	4.5	6.1	7.3	8.4	10.2
5	11.1	3.2	2.2	1.4	0	1.1	2.1	3.1	4.7	5.9	7	8.8
6	10	4.3	3.3	2.5	1.1	0	1	2	3.6	4.8	5.9	7.7
7	9	5.3	4.3	3.5	2.1	1	0	1	2.6	3.8	4.9	6.7
8	8	6.4	5.3	4.5	3.1	2	1.1	0	1.5	2.8	3.9	5.7
9	6.4	7.9	6.9	6.1	4.7	3.6	2.6	1.5	0	1.2	2.3	4.1
10	5.2	9.1	8.1	7.3	5.9	4.8	3.8	2.8	1.2	0	1.1	2.9
11	4.1	10.2	9.2	8.4	7	5.9	4.9	3.9	2.3	1.1	0	1.8
12	3	12	11	10.2	8.8	7.7	6.7	5.7	4.1	2.9	1.8	0

A.5 Tables des données de l'instance "Black Oxide 2"

TABLE A5.1 – Bornes temporelles de trempe m_i et M_i et temps de transport r_i

Cuve i	m_i	M_i	r_i
1	120	∞	25.9
2	180	240	18.6
3	50	90	12.4
4	60	240	13
5	50	90	18.7
6	60	120	12.6
7	120	180	18.6
8	60	90	13.1
9	120	180	18.8
10	40	90	12.7
11	35	120	20.6
12	240	422	14.6
1	0	0	

TABLE A5.2 – Temps de déplacement à vide $d_{i,j}$ des robots

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	14.3	13.3	12.5	11.1	10	9	8	6.4	5.2	4.1	3
2	14.3	0	1	1.8	3.2	4.3	5.3	6.4	7.9	9.1	10.2	12
3	13.3	1	0	0.8	2.2	3.3	4.3	5.3	6.9	8.1	9.2	11
4	12.5	1.8	0.8	0	1.4	2.5	3.5	4.5	6.1	7.3	8.4	10.2
5	11.1	3.2	2.2	1.4	0	1.1	2.1	3.1	4.7	5.9	7	8.8
6	10	4.3	3.3	2.5	1.1	0	1	2	3.6	4.8	5.9	7.7
7	9	5.3	4.3	3.5	2.1	1	0	1	2.6	3.8	4.9	6.7
8	8	6.4	5.3	4.5	3.1	2	1.1	0	1.5	2.8	3.9	5.7
9	6.4	7.9	6.9	6.1	4.7	3.6	2.6	1.5	0	1.2	2.3	4.1
10	5.2	9.1	8.1	7.3	5.9	4.8	3.8	2.8	1.2	0	1.1	2.9
11	4.1	10.2	9.2	8.4	7	5.9	4.9	3.9	2.3	1.1	0	1.8
12	3	12	11	10.2	8.8	7.7	6.7	5.7	4.1	2.9	1.8	0

A.6 Tables des données de l'instance "Cuivre"

TABLE A6.1 – Bornes temporelles de trempe m_i et M_i et temps de transport r_i

Cuve i	m_i	M_i	r_i
1	120	1000	16.7
2	240	420	16.8
3	60	180	17.6
4	60	120	16.8
5	60	180	19.2
6	120	300	22.5
7	1800	3000	23.6
8	60	180	20.8
9	120	300	18
10	600	840	18.3
11	45	1000	16.9
12	120	1000	18.9
1	0	0	

TABLE A6.2 – Temps de déplacement à vide $d_{i,j}$ des robots

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	8.8	9.9	11.7	10.6	14	15.2	12.9	7.9	5.6	4.2	3.1
2	8.8	0	1	2.8	1.8	5.2	6.3	4	1	3.2	4.6	5.7
3	9.9	1	0	1.8	0.8	4.1	5.3	3	2	4.3	5.6	6.8
4	11.7	2.8	1.8	0	1	2.3	3.5	1.2	3.8	6.1	7.4	8.6
5	10.6	1.8	0.8	1	0	3.4	4.5	2.2	2.8	5	6.4	7.5
6	14	5.2	4.1	2.3	3.4	0	1.2	1.1	6.2	8.4	9.8	10.9
7	15.2	6.3	5.3	3.5	4.5	1.2	0	2.3	7.3	9.6	1.09	12.1
8	12.9	4	3	1.2	2.2	1.1	2.3	0	5	7.3	8.6	9.8
9	7.9	1	2	3.8	2.8	6.2	7.3	5	0	2.2	3.6	4.7
10	5.6	3.2	4.3	6.1	5	8.4	9.6	7.3	2.2	0	1.4	2.5
11	4.2	4.6	5.6	7.4	6.4	9.8	10.9	8.6	3.6	1.4	0	1.1
12	3.1	5.7	6.8	8.6	7.5	10.9	12.1	9.8	4.7	2.5	1.1	0

A.7 Tables des données de l'instance "Zinc"

TABLE A7.1 – Bornes temporelles de trempe m_i et M_i et temps de transport r_i

Cuve i	m_i	M_i	r_i
1	60	∞	16.9
2	240	300	16.3
3	180	240	18.1
4	120	300	10.8
5	120	300	13.9
6	240	300	16
7	30	60	17.3
8	6.5	7.5	18.1
9	1680	1800	18.1
10	57	180	10.7
11	120	180	15.5
12	180	300	10.8
13	11.5	12.5	15.6
14	30	60	10.8
15	50	90	14.1
16	25	45	14
1	0	0	

TABLE A7.2 – Temps de déplacement à vide $d_{i,j}$ des robots

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	0	12.3	14.3	18.2	19.8	21.5	23.3	28.4	30.3	26.6	25.1	10.3	8.7	5.4	3.7	1.8
2	12.3	0	2.1	5.9	7.5	9.2	11	16.1	18	14.4	12.8	2	3.6	6.9	8.5	10.5
3	14.3	2.1	0	3.9	5.5	7.2	9	14.1	15.9	12.3	10.8	4	5.6	9	10.6	12.5
4	18.2	5.9	3.9	0	1.6	3.3	5.1	10.2	12.1	8.5	6.9	7.9	9.5	12.8	14.4	16.4
5	19.8	7.5	5.5	1.6	0	1.7	3.5	8.6	10.5	6.8	5.3	9.5	11.1	14.4	16	18
6	21.5	9.2	7.2	3.3	1.7	0	1.8	6.9	8.8	5.1	3.6	11.2	12.8	16.2	17.8	19.7
7	23.3	11	9	5.1	3.5	1.8	0	5.1	7	3.4	1.8	13	14.6	17.9	19.5	21.5
8	28.4	16.1	14.1	10.2	8.6	6.9	5.1	0	1.9	1.7	3.3	18.1	19.7	23	24.6	26.6
9	30.3	18	15.9	12.1	10.5	8.8	7	1.9	0	3.6	5.2	20	21.6	24.9	26.5	28.5
10	26.6	14.4	12.3	8.5	6.8	5.1	3.4	1.7	3.6	0	1.5	16.3	17.9	21.3	22.9	24.8
11	25.1	12.8	10.8	6.9	5.3	3.6	1.8	3.3	5.2	1.5	0	14.8	16.4	19.8	21.4	23.3
12	10.3	2	4	7.9	9.5	11.2	13	18.1	20	16.3	14.8	0	1.6	5	6.6	8.5
13	8.7	3.6	5.6	9.5	11.1	12.8	14.6	19.7	21.6	17.9	16.4	1.6	0	3.4	5	6.9
14	5.4	6.9	9	12.8	14.4	16.2	17.9	23	24.9	21.3	19.8	5	3.4	0	1.6	3.5
15	3.7	8.5	10.6	14.4	16	17.8	19.5	24.6	26.5	22.9	21.4	6.6	5	1.6	0	1.9
16	1.8	10.5	12.5	16.4	18	19.7	21.5	26.6	28.5	24.8	23.3	8.5	6.9	3.5	1.9	0

école doctorale sciences pour l'ingénieur et microtechniques SPIM

Titre : Modélisation et algorithmes pour le dimensionnement et l'ordonnancement cyclique d'atelier de traitement de surface

Mots clés : ordonnancement d'atelier, dimensionnement de ressources de transport, atelier de traitement de surface, métaheuristiques, programmation linéaire en nombres entiers mixtes

Résumé : Dans un environnement industriel compétitif, les entreprises sont en concurrence pour produire des biens de qualité, avec des coûts moindres et des délais raccourcis. Dans ce contexte, l'ordonnancement des activités permet d'améliorer les processus de production. Parmi la multitude des problèmes étudiés dans la littérature scientifique, l'ordonnancement des ateliers de traitement de surface cherche à planifier les déplacements des robots pour maximiser la productivité. Dans les lignes de galvanoplastie, les pièces doivent tremper dans plusieurs cuves contenant des solutions chimiques en suivant une gamme spécifique de production. Les robots programmables qui circulent en général au-dessus de la ligne, assurent la manutention et le transfert des pièces entre les cuves. Ils constituent souvent les ressources critiques de la ligne, quand le nombre de cuves et/ou des produits à manipuler devient grand. Le traitement des pièces, à la différence d'autres ateliers de production, se réalise à durées variables qui doivent se situer dans des fenêtres temporelles définies. Nous considérons des ateliers de production de masse où un nombre important de pièces doit être traité. Les robots répètent alors indéfiniment une même séquence de mouvements, formant un cycle, pour assurer la manutention des pièces tout au long de la ligne. Le problème qui recherche la séquence cyclique minimisant la période du cycle est connu sous le nom de problème d'ordonnancement cyclique des ateliers de traitement de surface (*Cyclic Hoist Scheduling Problem* ou CHSP).

Dans cette thèse, nous étudions les problèmes conjoints de dimensionnement et d'ordonnancement des ateliers de traitement de surface à plusieurs robots. Il s'agit d'une variante du CHSP, appelée Cyclic Hoist Design and Scheduling Problem (CHDSP). Notre objectif est d'optimiser le couple temps de cycle/nombre de robots. Sur ce problème bi-objectif, nous souhaitons apporter une contribution qui conduirait à terme à l'élaboration d'un système d'aide à la décision. Pour résoudre le CHDSP, nous développons des méthodes approchées adaptées, pour lesquelles nous proposons tout d'abord une nouvelle approche de codage des solutions basée sur les mouvements à vide des robots. Un algorithme génétique hybridé avec un programme linéaire en nombres entiers mixte (MILP), est élaboré, qui nous permet de valider ce nouveau codage et d'obtenir de premiers résultats intéressants. Nous développons également une deuxième métaheuristique hybride basée sur la recherche à voisinage variable et faisant appel au même MILP, pour optimiser la recherche des meilleures solutions dans un espace de recherche très vaste. Nous avons conduit une série de tests pour déterminer les meilleurs paramétrages de l'algorithme. Ce dernier a été testé sur plusieurs instances de la littérature sur lesquelles il montre sa performance et sa robustesse. Enfin, nous étendons la modélisation du problème CHDSP pour prendre en compte les contraintes d'évitement des collisions entre les robots qui partagent la même voie de circulation. Les tests réalisés sur les benchmarks de la littérature montrent l'efficacité de ce modèle complet.

Title: Model and algorithms for the Cyclic Hoist Design and Scheduling Problem

Keywords: Shop Scheduling, Sizing of transportation resources, Treatment surface facility, metaheuristics, mixed integer linear programming

Abstract: In a competitive industrial environment, companies compete to provide products of good quality, with lower costs and shorter lead times. In this context, the scheduling of activities allows to improve the production processes. Among the numerous problems studied in the scientific literature, the scheduling of surface treatment workshops seeks to plan the movements of hoists to maximize productivity. In electroplating lines, parts must soak in several tanks containing chemical solutions according to a given processing sequence. Programmable robots, called hoists, circulate above the line to ensure the handling and transfer of parts between the tanks. They often represent the line critical resources, when the number of tanks and/or products to be handled becomes large. The processing of parts, unlike other production workshops, is carried out at times which must be within defined time windows. We consider mass production systems where numerous parts must be processed. The hoists then repeat the same sequence of movements indefinitely, forming a cycle, to ensure the handling of parts throughout the line. The problem that searches the cyclic sequence which minimizes the cycle period is known as the Cyclic Hoist Scheduling Problem (CHSP).

In this work, we study the joint sizing and scheduling problems of multiple hoists in surface treatment facilities. This is a variant of CHSP, called Cyclic Hoist Design and Scheduling Problem (CHDSP). Our goal is to optimize both criteria cycle time and number of hoists. For this bi-objective problem, our contribution should ultimately lead to the development of a decision support system. To solve the CHDSP, we develop appropriate approximate methods, for which we first propose a new approach for coding solutions based on empty hoist moves. A genetic algorithm hybridized with a mixed integer linear program (MILP) is developed, which allows to validate this new encoding and to obtain interesting first results. We also develop a second hybrid metaheuristic based on variable neighborhood search and using the same MILP to optimize the search for the best solutions in a very large search space. We have conducted a series of tests to determine the best settings for the algorithm. The latter has been run on several instances of the literature for which it shows its performance and robustness. Finally, we extend the modeling of the CHDSP to consider the collisions avoidance constraints between hoists that share the same traffic rail. Tests carried out on benchmarks show the effectiveness of our complete model.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon