Contribution à l'ordonnancement des ateliers de traitement de surface avec deux robots

Samah Chtourou Kharrat

N° d'ordre : 184

UNIVERSITE DE TECHNOLOGIE DE BELFORT-MONTBELIARD

ECOLE DOCTORALE SCIENCES PHYSIQUES POUR L'INGENIEUR ET MICROTECHNIQUES

UNIVERSITÉ DE SFAX

FACULTÉ DES SCIENCES ÉCONOMIQUES ET DE GESTION DE SFAX

THESE

présentée par :

Sameh CHTOUROU épouse KHARRAT

Pour l'obtention du grade de

Docteur de l'Université de Technologie de Belfort-Montbéliard en Automatique

CONTRIBUTION À L'ORDONNANCEMENT DES ATELIERS DE TRAITEMENT DE SURFACE AVEC DEUX ROBOTS

Soutenue le 13 Décembre 2012 devant le jury d'Examen composé de :

Mr. Mohamed Ayadi	Professeur - ISG - Tunis	**Président**
Mme. Taîcir Moalla Loukil	Professeur - FSEG - Sfax	**Directeur de Thèseen cotutelle**
Mme. Marie-Ange Manier	Maitre de Conférence -UTBM- France	**Directeur de thèse**
Mr. Hichem Kammoun	Professeur -FSEG- Sfax	**Rapporteur**
Mr. Phillipe Lacomme	Maître de Conférences -UBP- France	**Rapporteur**
Mr. Talel Ladhari	Maître de Conférences -ESSECT- Tunis	**Membre invité**

Année Universitaire : 2011-2012

Dédicaces

À mes parents.

À mon mari et mon fils.

À toute ma famille et mes amis.

Remerciements

Je conçois ces années de préparation de la thèse de doctorat comme une expérience enrichissante intellectuellement et humainement, avec tous les moments d'enthousiasme et de motivation mais aussi, de doute et de mise en question notamment lorsque le moral menace de chanceler. je les conçois comme une expérience personnelle certes, mais à laquelle de nombreuses personnes ont contribué, de près ou de loin.

A toutes ces personnes dont je garderai toujours une empreinte, concrétisée dans ce "document de ma vie", je tiens à dire MERCI. Ces quelques mots ne sauraient résumer ma profonde gratitude au Professeur Taicir LOUKIL pour avoir accepté d'encadrer cette thèse. Je souhaite la remercier pour ses directives, ses conseils, sa disponibilité et la confiance qu'elle m'a accordée durant ces années. Merci également à mon Co-directrice de thèse, Mme Marie-ange, Maître de conférences à l'université Belford, de m'avoir accueilli au sein de l'équipe SET (Système Et Transport), ainsi que pour son encadrement, sa compétence et tous les précieux conseils qu'elle a pu m'apporter, tout au long de ce travail doctoral. Je la aussi pour ses qualités humaines.

J'exprime ensuite toute ma reconnaissance aux honorables membres du jury :

Un chaleureux Merci également à mes chers amis des équipes LOGIQ et GIAD, pour leur aide précieuse. Je tiens à remercier tout particulièrement Monsieur Mahdi Khemakhem, Maître

assistant à pour nous avoir initiée à la programmation grâce à la formation enrichissante qu'il nous a fait, ce métier dont il m'a transmis la passion.

Pour toujours, toute ma gratitude, tout mon amour et affection à mon père et ma mère, à qui je dois tout, et sans qui cette thèse n'aurait jamais vu le jour. Merci d'avoir été là pour moi, d'avoir tellement donné, et tellement sacrifié, pour faire en sorte que ce projet de vie aboutisse. Que cette thèse fasse votre fierté avant la mienne.

Pour son amour, pour son soutien indéfectible, pour sa patience et pour sa contribution effective à ce travail, un tendre merci à mon mari, Mohamed Kharrat. Que la fin de cette étape promette à notre petite famille le début d'une ère nouvelle. Enfin, je dédie ce travail à mon petit Mahmoud, qui constitue pour moi une véritable source de tendresse, sans lequel je n'aurais sans doute jamais pu présenter ce travail.

Plusieurs personnes m'ont assisté, d'une façon ou d'une autre, lors de la réalisation de cette thèse. J'aimerais ici tous les remercier profondément.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Face à la concurrence de plus en plus accentuée entre les pays industrialisés, les entreprises devraient utiliser aux mieux leurs ressources, notamment optimiser le trio *qualité-coût-délai*. Cette optimisation peut s'étendre du contexte général avec les outils de la Gestion de la Chaîne Logistique (GCL) au contexte plus spécifique notamment au niveau des problèmes de production.

Ainsi, dans les ateliers, une bonne gestion des ressources permet de rentabiliser les lignes de production. Cette gestion dépend, généralement, de l'ordonnancement proposé quel que soit le type de l'atelier.

Dans ce contexte économique, le développement de nouvelles méthodes de résolution des problèmes d'ordonnancement contribue à améliorer la productivité et par conséquent à mieux positionner l'entreprise vis-à-vis de ses concurrents.

Parmi les ateliers de production existants, les ateliers de traitement de surface présentent des spécificités qui rendent leur pilotage plus difficile.

Le traitement de surface est une opération chimique intermédiaire dans le processus de production. Cette opération est, généralement, effectuée entre les opérations d'usinage et les opérations de montage. Le traitement de surface ou galvanoplastie consiste à appliquer un dépôt sur les pièces à traiter. Ce procédé permet de modifier en surface et avec une faible variation de poids, soit les caractéristiques mécaniques d'un produit (conductibilité, anticorrosion, protection inoxydable,...), soit ses propriétés esthétiques (dépôt d'argent, dorure,

chromage,...). Le traitement consiste à immerger, successivement et sans attente, les pièces ou produits dans des bains contenant des solutions chimiques ou neutres. Des robots sont utilisés pour le déplacement des produits entre les cuves.
A la différence des autres ateliers de production, les temps de traitement dans les cuves doivent se trouver dans un intervalle de temps bien défini. Généralement, lorsque les temps de traitement sont assez élevés, les cuves deviennent les ressources critiques. De même, lorsque le nombre des cuves et des produits à traiter est assez élevé, les robots peuvent devenir les ressources critiques. Ceci peut constituer un goulet d'étranglement au niveau du processus de fabrication général. Ainsi, tout gain de productivité au niveau des ateliers de traitement de surface peut entraîner une hausse globale de la quantité produite.

Le travail effectué dans cette thèse a pour objet de proposer des méthodes efficaces pour la résolution des problèmes de traitement de surface dans le cas où :

- les produits à traiter sont du même type. Ce type de problème est connu dans la littérature sous l'appellation CHSP (Cyclic Hoist Scheduling Problem) (Manier et Bloch, 2003).
- les produits à traiter ne sont pas du même type. Ce type de problème est connu dans la littérature sous l'appellation DHSP (Dynamic Hoist Scheduling Problem) (Manier et Bloch, 2003).
- deux lots de produits doivent être traité dans un cadre de fonctionnement mixte, c'est-à-dire traiter dans une même installation des produits différents (DHSP) et des rafales de produits identiques (CHSP).

Nous traitons aussi le cas où le nombre des robots présents sur la ligne est égal à deux, ce qui augmentera le nombre des contraintes du problème. Dans le cas mono robot, ce problème a été prouvé NP-Complet (Lei et Wang, 1989b).

Dans le chapitre 1, nous présenterons les problèmes d'ordonnancement dans un contexte général, dans une première partie. Dans la deuxième partie, nous décrirons les lignes de traitement de surface. Une classification des différents problèmes, issue des travaux de Manier et Bloch (2003), sera donnée dans la troisième partie. Finalement dans la quatrième partie, nous proposons une étude bibliographique pour les problèmes mono-robot.

Dans le chapitre 2, nous proposerons une étude bibliographique pour les problèmes **HSP Cycliques** avec plusieurs ressources de transport. Ensuite, nous proposerons une méthode de résolution approchée pour des problèmes avec deux robots et des cuves de capacité unitaire. L'heuristique développée sera testée sur plusieurs instances et les résultats trouvés permettront de prouver sa validité et sa pertinence.

Dans le chapitre 3, nous généraliserons cette méthode de résolution en tenant compte de l'hypothèse que la gamme opératoire des produits est différente de l'ordre de l'implantation des cuves. Cette méthode sera également testée sur des cas empruntés à la littérature afin de montrer sa performance.

Dans le chapitre 4, tout d'abord, nous développerons, dans un premier temps, un programme linéaire mixte pour le problème du HSP Dynamique mono-robot et, dans un deuxième temps, un autre programme pour ce même problème avec deux robots. Ensuite, nous aborderons le problème HSP qui traite des produits différents (lot 1) et des rafales de produits identiques (lot 2) dans une même installation. Dans ce contexte, nous développerons une approche qui optimise la phase transitoire qui allie le traitement du lot 1 à celui du lot 2. Enfin, cette approche sera testée sur des instances générées conformément à des travaux de la littérature.

Chapitre 1

Ordonnancement des lignes de traitement de surface

Sommaire

1.1 Introduction

La stabilité, la complexité, la diversité des marchés et l'hostilité, sont quatre dimensions qui caractérisent tout environnement économique (Mintzberg, 1982). La perpétuelle évolution de cet environnement incite les entreprises à réagir et à être plus compétitives. Face à cette situation, les entreprises ont bien évolué ces dernières années pour faire face à cet aspect concurrentiel du marché et à son dynamisme. Pour rester compétitives, les entreprises doivent adapter leur outil de production afin qu'il puisse être en adéquation avec l'évolution de son environnement (Hitomi, 1996). L'évolution des systèmes de production est étroitement liée au développement des technologies informatiques. En effet, la robotisation de la production a permis de remplacer l'homme pour les tâches les plus répétitives, les plus pénibles et les plus dangereuses, que ce soit pour les produits ou les services. En plus, elle permet une meilleure précision et plus de stabilité dans le niveau de qualité des produits ou services.

Vers les années 60, sont apparus *les systèmes de production flexibles* ou *flexible manufacturing systems (FMS)* pour répondre à des exigences de personnalisation des produits, compte tenu de la diversité de la demande du marché. Ces ateliers se caractérisent par un système automatisé composé de machines commandées par ordinateur où des ressources de transport chargent et déchargent les pièces destinées à ce système ou aux composants de ce système.

L'automatisation a touché plusieurs types d'atelier, comme les cellules robotisées ou encore les ateliers de traitement de surface. Dans ces deux cas, des robots sont utilisés pour assurer le transport des produits à traiter. Ces opérations de transport sont importantes et déterminantes pour la qualité de fonctionnement du système, d'autant plus lorsque les robots constituent une ressource critique de l'atelier. L'amélioration de la productivité est alors garantie par un ordonnancement efficace des opérations de déplacement des robots.

Les travaux de cette thèse s'inscrivent dans le cadre des problèmes d'ordonnancement qui feront l'objet de ce chapitre. Ce dernier présente, en premier lieu (**section 1.2**), les problèmes d'ordonnancement d'une manière générale. Il décrit, en second lieu (**section 1.3**), les lignes de traitement de surface qui représentent un nouveau type d'atelier de production automatisé. Une classification des problèmes d'ordonnancement des lignes de traitement de surface est abordée en troisième lieu (**section 1.4**). En dernier lieu (**section 1.5**), un état de l'art sur les problèmes du hoist scheduling cycliques et mono-robot est présenté.

1.2 Présentation des problèmes d'ordonnancement

"*Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début*" (Carlier *et al.*, 1993).

Le problème d'ordonnancement consiste à affecter un ensemble de tâches à des ressources en tenant compte d'un ensemble de contraintes de fabrication et en optimisant certains critères.
Les problèmes d'ordonnancement trouvent leurs fondements dans divers processus industriels, opérationnel, de service, etc. Nous nous intéressons dans le reste de notre thèse aux processus industriels notamment les ateliers, étant donné le rôle majeur que joue l'ordonnancement dans la productivité des ateliers.

1.2.1 Notions de base

Dans un problème d'ordonnancement, quatre notions fondamentales doivent être définies : les tâches, les ressources, les variables de décision et les contraintes,

et enfin les objectifs.

1.2.1.1 Les tâches

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début et/ou de fin, dont la réalisation nécessite un certain nombre d'unités de temps (sa durée de traitement) et d'unités de chaque ressource.
Dans certains problèmes, les tâches peuvent être interrompues au cours de leur exécution, on parle alors de problèmes préemptifs. Dans d'autres, au contraire, elles doivent être exécutées sans interruption et le problème est dit non préemptif.
Lorsque les tâches ne sont pas liées entre elles par des contraintes de cohérence technologique, elles sont dites indépendantes.
Une tâche requiert pour son exécution certaines ressources qu'il faut programmer de façon à optimiser un certain objectif.

1.2.1.2 Les ressources

Une ressource est un moyen technique ou humain utilisé pour la réalisation d'une tâche. Elle est disponible en quantité limitée. On distingue plusieurs types de ressources :

- les ressources renouvelables : après avoir été allouées à une ou plusieurs tâches, elles sont à nouveau disponibles en même quantité, par exemple : les hommes, les machines, l'équipement en général, etc.
- les ressources consommables : après avoir été allouées à une ou plusieurs tâches, elles ne seront plus disponibles, par exemple : les matières premières, le budget, etc.

On distingue par ailleurs principalement dans le cas de ressources renouvelables :

- les ressources disjonctives qui ne peuvent exécuter qu'une tâche à la fois comme par exemple : machine-outil, robot manipulateur, etc.
- les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité, par exemple : équipe d'ouvriers,

poste de travail, etc.

1.2.1.3 Les variables de décision et les contraintes

Deux types de variables peuvent être distingués, selon qu'elles portent sur des décisions relatives au temps ou aux ressources. Ces variables sont dites , respectivement, variables temporelles et variables d'affectation.

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision (Lopez et Esquirol, 2001). On distingue :

- les contraintes temporelles. On distingue les contraintes de temps alloué et les contraintes d'antériorité :
  - les contraintes de temps alloué : elles sont issues généralement d'impératifs de gestion et elles sont relatives aux dates limites des tâches ou à la durée totale d'un projet comme par exemple : date de début au plus tôt, date de fin au plus tard, etc.
  - les contraintes d'antériorité ou de cohérence technologique : elles décrivent des relations d'ordre relatif entre les différentes tâches, par exemple, contraintes de gamme dans le cas des problèmes d'atelier, etc.
- les contraintes de ressources. On distingue les contraintes disjonctives et les contraintes cumulatives :
  - les contrainte disjonctives : elles imposent la réalisation disjointe de deux tâches, par exemple dans le cas où une seule ressource est partagée par plus que deux tâches en même temps.
  - les contrainte cumulatives : elles imposent que, pour une ressource donnée, le nombre de tâches réalisées simultanément doit être inférieur à sa capacité.

1.2.1.4 Les objectifs de l'ordonnancement

Dans la résolution d'un problème d'ordonnancement, on peut choisir entre deux grands types de stratégies, visant respectivement à l'optimalité des solutions, ou plus simplement à leur admissibilité. Dans ce dernier cas, la fonction d'évaluation porte sur le nombre de contraintes non satisfaites dans l'ordonnancement.

L'objectif d'un problème d'ordonnancement est de minimiser ou maximiser une fonction déterminée à partir d'un ou plusieurs critères d'évaluation. Les critères d'évaluation les plus courants sont (Carlier *et al.*, 1993) :

- la durée totale de l'ordonnancement appelée également makespan et notée C_{max} ;
- l'écart par rapport au délai souhaité avec avance favorable et retard défavorable, appelé Lateness ;

D'autres critères d'évaluation peuvent exister pour les problèmes d'ordonnancement basés sur des sommes pondérées ou non, à savoir :

- somme des temps de réalisation ;
- somme des retards dans l'ordonnancement ;
- somme pondérée des temps de réalisation. Dans ce cas, un poids est associé à une tâche qui traduit son importance par rapport aux autres tâches.

Ainsi, un problème d'ordonnancement se compose de tâches à réaliser et de machines disponibles pour l'exécution de ces tâches. La résolution d'un problème d'ordonnancement consiste à déterminer :

- l'affectation des tâches aux machines ;
- l'ordre dans lequel chaque machine exécute les tâches ;
- le placement des tâches dans le temps, c'est-à-dire les instants de début d'exécution de chaque tâche sur chaque machine.

Les différentes notions : tâches, ressources, variables de décision, contraintes et objectif sont nécessaires pour la définition d'un problème d'ordonnancement en général, et du problème d'atelier, en particulier, qui sera présenté ci-après.

1.2.2 Ordonnancement d'ateliers

"Dans les problèmes d'ordonnancement d'atelier, les ressources sont des machines ne pouvant réaliser qu'une seule tâche ou opération à la fois"(Lopez et Esquirol, 2001).

Les problèmes d'ordonnancement se divisent en deux grandes catégories selon le nombre d'opérations nécessaires à la réalisation de chaque travail. La première catégorie regroupe les problèmes pour lesquels chaque travail nécessite une seule opération. On parle de problèmes à une machine. La deuxième regroupe ceux pour lesquels chaque travail requiert plusieurs opérations.
Les problèmes de la deuxième catégorie sont généralement spécifiés par la donnée de m machines et de n travaux composés chacun de m opérations ; chaque opération devant être exécutée par une machine différente. Nous décrivons dans la suite les différents problèmes d'atelier classiques à m machines.

1.2.2.1 Atelier sans flexibilité des ressources

Plusieurs types d'ateliers ont été définis selon l'ordre d'utilisation des machines pour la fabrication d'un produit (gamme de fabrication). Ainsi, seules les contraintes temporelles et les contraintes de capacité des ressources sont considérées. Les dates de début des opérations constituent les inconnues du problème et leur détermination définit l'ordonnancement.
On trouve les ateliers à cheminement unique où toutes les gammes sont identiques (flow-shop), les ateliers à cheminement multiples où chaque produit ou famille de produits possède (job-shop), ou ne possède pas (open-shop) une gamme linéaire spécifique.

– **Atelier à cheminement unique ou Flow-shop**

Dans les ateliers de type "Flow-Shop", la ligne de production est constituée de plusieurs machines en série. Tous les produits visitent les machines dans le même

ordre, avec des durées opératoires pouvant être différentes. Le flux des pièces est unidirectionnel sur l'atelier. Les machines ne peuvent exécuter qu'une seule opération à la fois, de même que chaque opération ne peut s'exécuter que sur une seule machine à la fois.

– **Atelier à cheminement multiple ou Job-Shop**

Dans les ateliers de type "job shop", chaque produit passe sur les machines dans un ordre fixé par sa gamme et variant d'un produit à l'autre.

– **Atelier à cheminement quelconque ou Open-Shop**

Dans les ateliers de type "Open shop", les gammes ne sont pas linéaires. En effet, toutes ou partie des opérations peuvent être traitées dans n'importe quel ordre.

1.2.2.2 Atelier avec flexibilité des ressources

Le besoin de réactivité et d'amélioration de la productivité ont conduit les entreprises à rechercher de nouvelles alternatives comme la duplication des machines pour effectuer une ou plusieurs opérations. On parle alors de systèmes avec flexibilité des ressources.

La présence de ressources en exemplaires multiples induit un problème supplémentaire qui est celui de l'affectation : la machine nécessaire pour exécuter une opération doit être sélectionnée parmi un ensemble donné. On distingue trois types différents de machines en fonction des vitesses de celles-ci :

- machines identiques : toutes les machines présentent la même vitesse d'exécution quel que soit le travail,
- machines uniformes : la vitesse d'une machine diffère d'une autre par un coefficient de proportionnalité,
- machines indépendantes : chaque machine présente une vitesse particulière pour chaque travail ; la durée opératoire d'un travail dépend donc du travail et de la machine qui l'exécute.

Les problèmes avec flexibilité de ressources les plus étudiés dans la littérature sont : les problèmes à machines parallèles, le flow shop hybride et le job shop

flexible.

– **Machines parallèles**

Plusieurs machines interviennent pour réaliser les travaux. Chaque travail se compose d'une seule opération et exige donc une seule machine pour sa réalisation. L'ordonnancement s'effectue en deux phases : la première phase consiste à affecter les travaux aux machines et la deuxième phase consiste à établir la séquence et les dates de réalisation sur chaque machine.

– **Flow-Shop hybride**

Le flow shop hybride est une extension du flow shop classique pour lequel chaque travail doit subir des opérations différentes, et chaque opération doit se faire sur une machine choisie parmi un ensemble, appelé étage, de machines parallèles dédiées à cette opération. Une machine ne peut appartenir qu'à un seul étage.

– **Job-Shop flexible**

Dans le job-shop flexible, une extension du modèle job-shop classique, plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations. Plus précisément, une opération est associée à un ensemble contenant toutes les machines pouvant effectuer cette opération.

1.2.2.3 Atelier avec ressources de transport

Dans les problèmes d'atelier précédents, les durées de transport sont négligeables par rapport aux durées d'exécution sur les machines et ne sont pas prises en compte dans la modélisation du problème. Généralement, dans les ateliers automatisés, plusieurs moyens de transport (robots, chariots filoguidés, etc.) seront utilisés pour le déplacement des produits entre les machines. Ce type d'atelier est défini comme atelier avec ressources de transport et qui prend en compte les durées de transport lors de la modélisation du problème.

Les systèmes avec ressources de transport les plus étudiés dans la littérature sont : les ateliers de traitement de surface (le problème d'ordonnancement associé étant appelé Hoist Scheduling Problem) et les problèmes des Systèmes Flexibles

de Production (SFP) ou FMS (Flexible Manufacturing System).

– **Hoist Scheduling Problem (HSP)**

Dans les ateliers de traitement de surface, les durées de trempe dans les cuves (machines) sont bornées. Chaque produit à traiter possède une gamme qui indique les numéro des cuves (machines) à visiter. Le déplacement entre les cuves est réalisé par des robots suspendus à un rail unidirectionnel. Dans le reste de la thèse, notre intérêt sera tout particulièrement accordé, à ce type d'atelier qui sera décrit dans la **section 1.3**.

– **Flexible Manufacturing System**

Dans les ateliers de type "FMS", les durées d'exécution des produits dans les machines ne sont pas bornées ou seulement par valeur inférieure. Généralement, les produits sont déplacés entre les machines par des chariots automatiquement guidés.

1.3 Présentation des lignes de traitement de surfaces

Depuis l'article fondateur de W.Phillips et S.Unger (1976), l'ordonnancement des ateliers de traitement de surface a fait l'objet de divers travaux dans la littérature. En effet, il existe des types de produits qui nécessitent lors de leurs fabrications une opération mécanique, physique , chimique ou électrochimique sur ce type de ligne. Ce traitement a pour but de changer l'aspect ou la forme de la surface des produits pour des besoins ou conditions d'utilisation. Par exemple, on peut trouver ce type de lignes de production dans l'industrie mécanique pour des opérations de trempe (bains électrolytiques, rinçage, etc.), dans l'industrie électronique pour la fabrication des circuits imprimés ou les opérations de dépôt d'or, d'argent, de cuivre ou d'autres métaux.

1.3.1 Description physique des lignes

Généralement, une ligne de traitement de surface est composée de plusieurs cuves contenant des bains de traitement chimiques et/ou électrolytiques. Les pièces (ou produits) sont traité(e)s suivant une gamme opératoire qui consiste en un enchaînement de traitements à réaliser dans les cuves correspondantes. Un ou plusieurs robot(s) est (sont) utilisé(s) pour effectuer le déplacement de porteurs sur lesquels sont chargées les pièces (ou produits) par lots. Ce robot est généralement installé sur un rail pour permettre les déplacements d'une cuve à une autre. Le robot est appelé aussi palan ou hoist en anglais. La Figure 1.1 donne une représentation schématique d'une ligne de traitement de surface, formée par :

- les cuves où les produits seront traités ;
- les porteurs dans lesquels les produits sont mis afin d'être transportés entre les cuves ;
- les robots qui déplacent les porteurs.

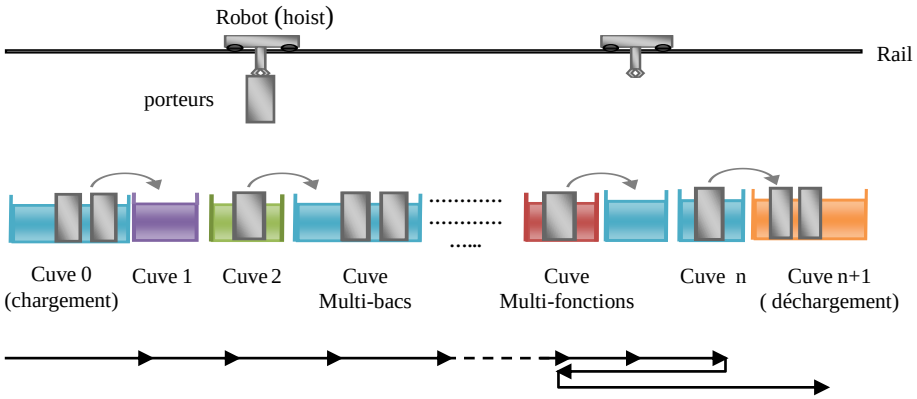

FIGURE 1.1 – Schéma d'une ligne de traitement de surface

1.3.1.1 Les cuves

Les cuves contiennent des solutions chimiques nécessaires aux traitements souhaités et dans lesquelles les porteurs sont immergés. Les durées de traitement dans chaque cuve sont bornées. Ainsi, un porteur ne doit pas être retiré d'une cuve

avant une durée minimale (borne inférieure) et ne doit pas dépasser une durée maximale (borne supérieure) pour éviter que les produits soient défectueux ou subissent une dégradation sur la qualité souhaitée. Ces durées sont généralement définies par des experts chimistes responsables de la fabrication.
Il existe plusieurs types de cuves :

- Cuve mono-bac : elle ne peut accueillir qu'un seul produit à la fois.
- Cuve multi-bacs : parfois la durée de traitement dans une cuve est nettement supérieure aux durées de traitement dans les autres cuves. Ceci peut entraîner un goulot d'étranglement (limitation de la capacité de production) sur la ligne. Pour éviter ce problème, il est préférable d'augmenter sa capacité en insérant plusieurs bacs contenant la même solution et ayant les mêmes caractéristiques que les cuves. Le nombre de cuves multi-bacs dans une ligne est assez faible. Par contre, le nombre de bacs est plus élevé et peut atteindre la dizaine. Un exemple de cuve multi-bacs est illustré dans la Figure 1.1,
- Cuve mono-fonction : elle apparaît une fois au plus dans la gamme de fabrication de chaque produit,
- Cuve multi-fonctions : un produit peut utiliser plusieurs fois la même cuve dans sa gamme de fabrication. Les traitements réalisés dans ces cuves diffèrent seulement au niveau des durées de trempe et non pas au niveau du traitement puisque la solution chimique est la même. Généralement, les traitements dans ces cuves sont de type rinçage et ont des durées opératoires petites. En plus, deux traitements appartenant à une même gamme réalisés dans une cuve multi-fonctions ne doivent pas être consécutifs.
- Cuves de chargement et de déchargement : Ce sont les postes d'entrée et de sortie des porteurs de la ligne. Ils sont considérés comme des cuves fictives et sont le plus souvent considérés de capacité infinie. Les opérations de chargement et déchargement des porteurs sont, en général, effectuées manuellement.

Plusieurs configurations des cuves de chargement et déchargement peuvent exister. Ainsi, elles peuvent être :

- dissociées : les deux cuves sont en général situées à chaque extrémité de la ligne.
- associées : c'est le cas si la place dans l'atelier est limitée ou bien si l'opérateur responsable des opérations de chargement et déchargement est le même. Ainsi, les deux cuves se situent à la même extrémité de la ligne et n'en forment qu'une seule.

– Cuves de transfert : dans le cas d'ateliers de grande taille avec des lignes en parallèle, on trouve des cuves qui permettent de transporter les porteurs d'une partie de la ligne à une autre ; ce sont généralement soit des cuves qui peuvent se déplacer sur un rail soit sur un tapis roulant. Ces cuves peuvent être soit vides soit de type rinçage.

1.3.1.2 Les porteurs

Les pièces à traiter sont généralement installées sur des porteurs afin d'être transportées par lots d'une cuve à une autre. Ceci est dû à plusieurs raisons : quantité trop élevée, dimension très petite des pièces, etc. Il existe plusieurs types de porteurs qui dépendent de la nature des pièces à transporter :

– Les tonneaux : ce sont des cylindres permettant de transporter les pièces de petites tailles de type vis, boulons, etc. Ils fonctionnent en rotation par rapport à leur axe pour que chaque pièce ait toute sa surface traitée.
– Les paniers : ce sont des réservoirs contenant des pièces et sont déposés dans les cuves. Les pièces mises dans les paniers sont généralement de petites tailles.
– Les cadres : ils sont utilisés pour les pièces de plus grande taille. Ces pièces sont généralement accrochées au cadre comme par exemple, les tiges d'amortisseurs de voiture, les cartes pour les circuits imprimés, etc.

1.3.1.3 Les robots

Les robots ou palans sont les moyens de transport des porteurs. Ils permettent le déplacement des porteurs d'une cuve à une autre tout en se déplaçant sur des rails. Deux types de déplacement sont effectués par un robot :

- Déplacement en charge : le robot déplace un porteur d'une cuve à une autre. Ce déplacement comporte :
  - Positionnement au dessus du porteur dès qu'il termine son traitement ;
  - Descente et saisie pour la récupération du porteur, ensuite élévation au dessus de la cuve et égouttage ;
  - Déplacement du porteur jusqu'à la cuve du traitement suivant dans la gamme opératoire ;
  - Stabilisation, descente puis dépôt du porteur dans la nouvelle cuve.
- Déplacement à vide : il s'effectue entre deux déplacements en charge. Ainsi, le robot doit se déplacer à vide de la cuve d'arrivée du premier transport vers la cuve de départ du deuxième transport.

La Figure 1.2 présente les différents mouvements possibles d'un robot pour les deux types de déplacements. Les flèches en pointillé représentent les mouvements à vide tandis que les flèches continues représentent les mouvements en charge (pouvant être moins rapides que ceux à vide).

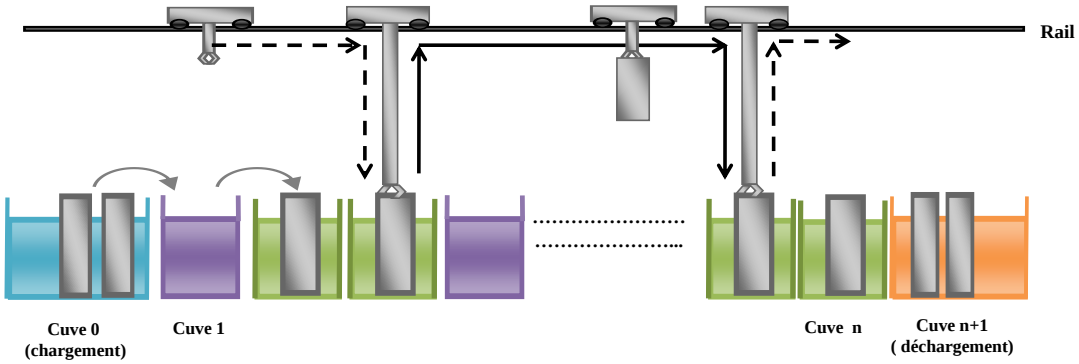

FIGURE 1.2 – Mouvements du robot pour le transport d'un porteur

Les durées de déplacement des robots entre les cuves sont connues ainsi que les vitesses. Un robot ne peut s'arrêter pendant un déplacement que pour l'un des cas suivants :

- pour effectuer l'égouttage du porteur au dessus d'une cuve ;
- pour se stabiliser au dessus d'une cuve ;
- lorsqu'il n'y a pas de transport à effectuer.

Une pause du robot en cours de transport en charge, et en dehors de ces cas, aura pour conséquence une dégradation possible des produits (pour cause d'oxydation par exemple).

1.3.1.4 Les lignes

Dans les ateliers de traitements de surface, il existe des lignes simples et des lignes complexes. Ceci dépend de plusieurs critères (le nombre des cuves, des robots, la taille de l'atelier, etc.).

Une ligne est dite simple si elle n'utilise qu'un seul robot et les cuves qui la constituent sont mono-bac et mono-fonction. Certaines lignes ont la particularité de comporter plusieurs robots sur un même rail dans le but de partager les tâches afin d'augmenter la productivité ; on parle alors de lignes complexes. Dans ce type de ligne, les robots ne doivent pas se croiser, donc il faut gérer le

risque de collision.

Parfois le nombre de cuves est très important de façon qu'il est impossible de les disposer sur une seule ligne. Plusieurs lignes sont alors disposées en parallèle, reliées par des systèmes de transport autres que les robots. Ces systèmes sont généralement des chariots de transfert ou bien des cuves de transfert qui transportent les porteurs entre plusieurs tronçons de ligne (souvent dits en "I" du fait de leur configuration linéaire). Parfois, une synchronisation entre le système de transfert et un robot est nécessaire pour l'échange du porteur. Le robot vient déposer celui-ci dans le chariot qui le transporte jusqu'au tronçon sur lequel il doit continuer son traitement, puis un autre vient s'en saisir. Souvent, ces chariots contiennent des produits neutres.

Trois implantations sont habituellement utilisées pour les lignes multi- tronçons : implantation en U, O ou H comme le montre la Figure 1.3 ci-dessous.

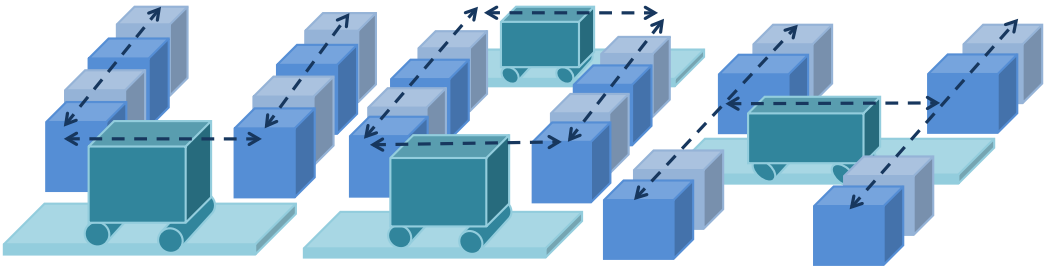

FIGURE 1.3 – Implantation en U, O ou H (Manier, 1994)

1.3.2 Hoist scheduling problem

Le traitement de surface inclut trois types de ressources : les cuves, les porteurs et les robots, et deux types d'opérations : opération de trempe (opération de traitement des produits dans les cuves) et opération de transport (déplacement des porteurs par les robots entre les cuves).

Le problème d'ordonnancement des ateliers de traitement de surface est connu dans la littérature sous l'appellation Hoist Scheduling Problem (HSP). Il consiste à ordonnancer les mouvements de transport du robot pour optimiser un ou plusieurs critères (Manier *et al.*, 2000), tout en respectant un ensemble de contraintes :

- Les contraintes liées aux ressources :
  - Contraintes de capacité des cuves : chaque cuve ne peut pas recevoir plus que sa capacité en même temps. Ainsi, une cuve peut recevoir un porteur (pour les cuves mono-bac) ou plusieurs porteurs (pour les cuves multi-bacs). Il faut donc s'assurer que la cuve mono-bac soit vide avant de recevoir un porteur ou qu'il reste un bac libre pour les cuves multi-bacs.
  - Contraintes de capacité des robots : chaque robot ne peut pas déplacer plus qu'un porteur en même temps. En plus, entre deux mouvements de transfert consécutifs, le robot doit avoir suffisamment de temps pour se déplacer à vide entre la cuve d'arrivée du premier transport et la cuve origine du deuxième transport.
  - Contraintes de collision entre les robots : dans le cas où la ligne comporte plus qu'un robot partageant un unique rail de déplacement, il faut éviter le risque de collision entre les robots.
- Les contraintes relatives à la gamme de production :
  - les durées opératoires relatives à chaque opération de traitement sont bornées (avec une durée minimale et maximale).
  - l'ordre d'exécution des opérations pour chaque porteur doit être respecté.
- Contraintes de non-préemption, non attente et non stock : aucune opération (trempe ou transport) ne peut être interrompue une fois commencée ;

un produit ne peut pas attendre dans la cuve une fois son traitement est terminé et un robot ne peut pas faire de pause en charge. Enfin, il n'y a pas de cuve de stockage intermédiaire entre deux cuves de traitement (en cas d'occupation de la cuve suivante).

Du fait de ces dernières contraintes, un porteur doit enchaîner sans attente et alternativement des opérations de trempe et de transport. Ceci implique une égalité entre la date de fin d'un transport et la date de début de l'opération de trempe suivante. Ceci explique que si toutes les tâches (trempe + transport) sont à ordonnancer, cela revient à ordonnancer les tâches de transport, d'où la définition que nous avons donnée pour le HSP au début de ce paragraphe.

Il existe deux types de problèmes de traitement de surface : les problèmes statiques et les problèmes dynamiques. Le choix de l'approche à adopter dépend du compromis à réaliser entre les trois objectifs qui sont la *productivité*, la *qualité* et la *flexibilité* (Baptiste *et al.*, 2001).

1.3.2.1 HSP statique

On distingue deux types de HSP statique : le problème de traitement de surface cyclique (CHSP) et le problème de traitement de surface prédictif (PHSP).

– **HSP cyclique (CHSP)**

Pour la production en grande série, les produits entrant dans la ligne sont identiques et la production est dite mono-produit. On cherche alors une séquence des mouvements du robot à répéter indéfiniment. Cette séquence est appelée cycle. Un cycle est dit de degré n ou n-périodique lorsque n pièces sont introduites dans la ligne au cours d'une période et n pièces quittent la ligne. Ce mode de fonctionnement cyclique a donné lieu à la définition d'un premier problème d'ordonnancement des lignes de traitement de surfaces connu sous le nom de Cyclic Hoist Scheduling Problem ou CHSP (Lei et Wang, 1989a). L'objectif cherché est la détermination du cycle de durée minimale (période T) pour un degré de cycle

n fixé, afin de maximiser la *productivité* de la ligne. La solution trouvée doit spécifier la séquence des mouvements ainsi que la date de début de chacun d'eux au cours d'une période. Cette séquence doit être réalisable, c'est-à-dire :

- les durées opératoires doivent être conformes aux bornes spécifiées,
- la séquence des mouvements doit être exécutable par les robots.

Le CHSP étant un problème fortement contraint et NP-complet (Lei et Wang, 1989b), la plupart des chercheurs qui s'y sont intéressés n'ont résolu que le problème le plus simple c'est-à-dire le problème 1-cyclique avec une ligne simple ne comportant que des cuves mono-bac et mono-fonction et desservies par un seul robot.

- **HSP prédictif (PHSP)**

Dans le cas d'une production multi-produits (ou multi-gammes) et en grandes séries, s'il n'existe que deux ou trois gammes et qui diffèrent très peu, il reste possible d'appliquer un mode de gestion cyclique. Sinon, un autre mode de gestion est envisageable et qui consiste à travailler par campagne. Dans ce cas on enchaîne des phases de production monogamme. Même si la solution la plus simple consiste à terminer la production d'une campagne avant d'entamer une autre, il est plus pertinent de déterminer un ordonnancement permettant de faire cohabiter les deux types de produits au cours d'une phase transitoire (Varnier, 1996). L'objectif est de minimiser la durée de cette phase.

Lorsque la ligne est capable de traiter des produits différents et par petites séries, le mode cyclique ou par campagne n'est plus envisageable. Dans ce cas, si la production est stable et prévue sur un horizon relativement petit, un autre mode de gestion est défini. Il consiste à chercher un ordonnancement pour les différents produits à traiter avant l'entrée en ligne (Caux *et al.*, 1995). Ces deux problèmes statiques mais non cycliques ont été regroupés sous le nom de "Predictive Hoist Scheduling Problem " ou *PHSP*.

1.3.2.2 HSP dynamique

Traiter une grande variété de produits par petites quantités et en temps limité est inévitable pour la plupart des sociétés, surtout celles travaillant en sous traitance. En effet, face à la diminution de la taille des séries et à la réduction des délais, les responsables de production doivent combiner les différents produits dans une même ligne et donc augmenter la flexibilité de leurs systèmes automatisés. La diminution des délais de production de plus en plus exigée par les clients entraîne la difficulté de faire une planification fiable sur plusieurs heures. Ainsi les responsables de fabrication doivent être capables d'agir vite face à une commande urgente. Pour la planification des mouvements des robots en fonction d'un compromis recherché entre les différents objectifs (*productivité*, la *qualité* et la *flexibilité*), deux approches peuvent être envisagées : le problème de traitement de surface dynamique (DHSP) et le problème de traitement de surface réactif (RHSP).

– **HSP dynamique (DHSP)**

Si le respect de la qualité est l'objectif primordial, toutes les opérations de transport sont affectées au moment où un nouveau porteur entre dans le système en tenant compte de l'état de la ligne, en particulier des porteurs en cours de traitement. Cet ordonnancement doit respecter toutes les contraintes et permettre au nouveau porteur d'entrer sur la ligne le plus tôt possible. Le critère à minimiser est généralement le makespan. Cette approche est adaptée pour les productions à forte valeur ajoutée et offre l'avantage de ne pas être sensible aux perturbations externes telles que l'arrivée d'une commande urgente ou le retard de certains produits. Le problème d'ordonnancement associé est nommé "Dynamic Hoist Scheduling Problem" ou DHSP. Dans sa thèse, Lamothe le décrit comme le pilotage réactif avec prévision (Lamothe, 1996).

– **HSP réactif (RHSP)**

Si la productivité ainsi que la flexibilité sont l'objectif de la société, alors après chaque opération de transport un algorithme détermine quel est le pro-

chain porteur à déplacer, tout en tenant compte de l'état de la ligne. Aucun ordonnancement prévisionnel n'est déterminé.

Cette affectation est envisagée en considérant la nature du bain dans lequel se trouve le porteur à déplacer. Parfois, le respect des temps de traitement n'est pas assuré, surtout au niveau de la borne supérieure. Cette approche est adaptée pour les productions à faible valeur ajoutée. Le problème d'ordonnancement associé est nommé "Reactive Hoist Scheduling Problem" ou RHSP. Dans sa thèse, Lamothe le décrit comme le pilotage réactif sans prévision (Lamothe, 1996).

Dans la suite de ce chapitre, nous nous intéresserons au HSP cyclique mono-robot, c'est-à-dire, celui qui utilise un seul robot comme ressource de transport.

1.4 Classification des problèmes d'ordonnancement des lignes de traitement de surface

Manier et Bloch (2003) ont proposé une classification pour les problèmes d'ordonnancement des lignes de traitement de surface. Les auteurs considèrent les principaux paramètres physiques et logiques des problèmes HSP déterminés dans la littérature.

Elles ont défini quatre domaines : le type du HSP (α), les paramètres physiques (β), les paramètres logiques (δ) et les critères d'optimisation (γ).

Chaque domaine se compose de plusieurs paramètres :

- **Type de HSP** (α=***XHSP***) : la variable X dans $XHSP$ décrit le type de HSP considéré ($X \in C, P, D, R$). Le Hoist Scheduling Problem peut être statique (cyclique ($CHSP$) ou predictif ($PHSP$)), ou dynamique (dynamique ($DHSP$) ou réactif ($RHSP$)) ;
- **Paramètres physiques** ($\beta = \beta_1, \beta_2, \beta_3, \beta_4/\beta_5, \beta_6, \beta_7, \beta_8/\beta_9$) : ce domaine inclut respectivement :
  - β_1=nl : le nombre de lignes ;

- β_2=*ntransfer* : le nombre des moyens de transfert reliant les lignes ;
- $\beta_3 \in \{\phi, synchro\}$: le besoin de synchronisation entre les robots et les moyens de transfert. ϕ indique qu'il n'existe pas de synchronisation ;
- $\beta_4 = (\beta_{41}, \beta_{42}, \beta_{43})$: indique, respectivement, pour chaque ligne installée, le nombre de robots (mh), le nombre de cuves (mt) et la capacité maximale des cuves (ct). Si pour chaque ligne le nombre de robots est égal à un ainsi que la capacité des cuves, alors $(\beta_{41}, \beta_{42}, \beta_{43}) = (\phi, mt, \phi)$;
- β_5=nc : le nombre de porteurs (par défaut $\beta_5 = \phi$ 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 entre les porteurs a été introduite par Shapiro et Nuttle (1988) et signifie qu'un porteur déchargé pendant un cycle doit être chargé pour le cycle suivant ;
- $\beta_7 \in \{\phi, ret\}$: ret indique qu'il existe un autre système qui assure le transport des porteurs vides de la cuve de déchargement vers la cuve de chargement, (ϕ) indique que soit le robot assure cette fonction ou que les deux cuves sont associées ;
- $\beta_8 \in \{\phi, empty\}$: est associée à la nécessité de gérer les porteurs vides qui ne peuvent être évacués de la ligne. β_8= empty, si les porteurs vides ne peuvent pas être stockés hors ligne ou au poste de chargement/ déchargement, sinon $\beta_8 = \phi$;
- $\beta_9 \in \{\phi, ass, diss\}$: elle décrit une configuration donnée pour les cuves de chargement et déchargement (quelconque, postes associés ou dissociés) ;

– **Paramètres logiques** ($\delta = \delta_1/\delta_2, \delta_3, \delta_4, \delta_5$) : ils décrivent l'environnement de production pris en considération qui inclut :

- δ_1=*nparts* : la quantité de pièces (ou lots) à traiter dans la ligne. Par défaut, (ϕ) indique qu'il existe une infinité de pièces ;
- δ_2=*nps* : le nombre de types de produits à traiter. Si toutes les pièces sont identiques alors $\delta_2 = \phi$;
- δ_3=*nop* : le nombre maximal d'opérations dans les gammes des produits à traiter ;
- $\delta_4 \in \{\phi, clean\}$: elle indique si les porteurs doivent être nettoyés ou non après le déchargement. (ϕ) indique que le déchargement est la dernière opération effectuée ;
- $\delta_5 \in \{\phi, recrc\}$: il indique la notion de recirculation introduite par Pinedo (1995) et qui concerne les cuves multi-fonction. (ϕ) indique que toutes les cuves sont mono-fonction ;

– **Les critères d'optimisation** (criteria)(γ) : l'objectif pour le HSP peut être :

- la minimisation de la durée du cycle pour un CHSP ($Tmin$) ;
- la minimisation du makespan ($Cmax$) ;
- la maximisation de la productivité ($ThroughputMax$) ;
- la minimisation du nombre de robots ($mhmin$) ;
- la minimisation du nombre de pièces défectueuses ;
- etc.

Ainsi une notation complète est généralisée par :
XHSP|nl,ntransfer,synchro,(mh, mt, ct)$_{i=1nl}$/nc,circ,ret,empty/load-unload| nparts/nps,nop,clean,recrc|criteria.

1.5 Le HSP cyclique mono-robot

(CHSP|1,mt,ct/nc,circ,ret,empty/load-unload|nparts/nps,nop,clean,recrc|criteria)

Lorsque les produits à traiter suivent la même gamme opératoire, l'ordonnancement est généralement réalisé d'une manière séquentielle. Cette séquence est appelée un cycle. Ainsi, le ou les robots sont programmés pour effectuer, successivement et un grand nombre de fois, les séquences fixées des opérations de transport. Le temps requis pour exécuter une séquence de mouvements est défini par la durée du cycle ou la période. Le degré du cycle est défini par le nombre r ($r \geq 1$) de porteurs introduits pendant une période et est dit cycle r-périodique ou r-cycle. Pendant une période, r porteurs entrent et r porteurs sortent de la ligne.

La propriété découlant d'un cycle est que l'état du système reste le même à tout instant m modulo la période T, c'est-à-dire le nombre ainsi que la position des porteurs, la position des robots et les durées de traitement dans les cuves sont les mêmes. Pour un cycle de degré 1, le nombre de porteurs se trouvant dans la ligne à tout instant représente le nombre de cycles nécessaires pour qu'un produit termine son traitement. La gamme opératoire contient, généralement, les opérations de traitement ainsi que les deux opérations de chargement et déchargement.

Le CHSP consiste à ordonnancer les mouvements des robots, c'est-à-dire déterminer une séquence des mouvements des robots ainsi que les dates de début d'exécution de ces mouvements. L'objectif est de minimiser la durée d'un cycle. Le problème le plus simple à traiter est le problème cyclique mono-robot ayant des cuves mono-bac et mono-fonction. Ce problème doit respecter, généralement, les quatre contraintes principales :

1. La durée opératoire dans chaque cuve doit être comprise entre une borne minimale et une borne maximale fixées ;
2. Toute cuve ne doit pas recevoir plus qu'un porteur en même temps ;
3. Le robot ne peut pas effectuer deux mouvements de transport en même temps ;

4. Le robot doit avoir suffisamment de temps pour se déplacer à vide entre deux opérations de transport (depuis la cuve recevant un porteur faisant l'objet du dernier mouvement du robot jusqu'à la cuve d'où il doit retirer un autre porteur).

Le CHSP est un problème dont la résolution est très complexe. En effet, Lei et Wang (1989b) ont montré que la résolution du problème cyclique mono-produit et mono-robot fait partie de la classe des problèmes NP-complets. Manier et Baptiste (1994) ensuite Bloch *et al.* (2008) ont effectué un état de l'art sur le HSP où ils ont exposé les différents travaux et méthodes de résolution réalisés pour ce type de problèmes ainsi que les avantages et inconvénients pour chaque méthode. Le Tableau 1.1 présente quelques références sur les principales publications (Bloch *et al.*, 2008) auquel nous avons ajouté quelques références plus récentes sur le problème du Hoist Scheduling Problem.

TABLE 1.1 – Publications relatives aux différentes classes du HSP

CHSP	**Mono-robot, cuves mono-bac et mono-produit :** W.Phillips et S.Unger (1976), Bracker et Chapman (1985), Lei (1993), Song *et al.* (1993), Armstrong *et al.* (1994), Song *et al.* (1995), Baptiste *et al.* (1996), Kats et Levner (1997), Lim (1997), Ng et Leung (1997), Levner *et al.* (1997), Chen *et al.* (1998), Liu *et al.* (2002)
	Mono-robot, cuves mono-bac et multi-produits : Varnier et Jeunehomme (2000), Mateo *et al.* (2000), Mateo et Companys (2006), Mateo et Companys (2007), El-Amraoui *et al.* (2008), El-Amraoui *et al.* (2009), El-Amraoui *et al.* (2012)
	Mono-robot, cuves multi-bacs et mono-produit : Shapiro et Nuttle (1988), Lei et Wang (1989a), Hanen et Munier (1993), Ng (1995), Liu *et al.* (2002), Zhou et Li (2003), Che et Chu (2005)
	Multi-robots, cuves mono-bac et mono-produit : Lei et Wang (1991), Lei *et al.* (1993), Manier *et al.* (2000), Yang *et al.* (2001), Leung et Zhang (2003), Leung *et al.* (2004), Manier et Lamrous (2008), Zhou et Liu (2008), Zhou et Li (2009),
	Multi-robots, cuves multi-bacs et mono-produit : Hanen (1994), Manier (1994), Armstrong *et al.* (1996), Varnier *et al.* (1997), Manier *et al.* (2000), Riera et Yorke-Smith (2002), Zhou et Li (2009)
PHSP	**Ordonnancement de phases transitoires :** Varnier (1996)
	Ordonnancement des travaux en entrée de ligne : Caux *et al.* (1995), Fleury (1995), Caux et Pierreval (1997) Lacomme (1998), Bloch *et al.* (1999)
DHSP	**Mono-robot et mono-bac :** Yih (1994), Ge et Yih (1995), Cheng et Smith (1995), Rosse-Bloch (1999), Fargier et Lamothe (2001), Hindi et Fleszar (2004), Paul *et al.* (2007)
	Mono-robot et multi-bacs : Spacek *et al.* (1999)
	Multi-robots : (Lamothe, 1996)
RHSP	**Mono-robot et cuves mono-bac :** (Yih *et al.*, 1993)
	Multi-robots : (Thesen et Lei, 1990), (Jegou *et al.*, 2006)

1.5.1 Notations

Nous présentons dans ce qui suit une notation commune pour tous les travaux réalisés sur le HSP.

Nous désignons par *move* i, l'opération effectuée par le robot et qui consiste à transporter un porteur de la cuve i vers la cuve $i+1$ $(i = 0, ..., n)$. Un porteur débute son traitement lorsqu'il quitte la cuve de chargement et finit son

traitement lorsque le robot le dépose au poste de déchargement, ce qui explique que les temps de chargement et déchargement ne soient pas pris en compte dans la notation.

Les paramètres du problème sont :

n le nombre de cuves de traitement ; la cuve de chargement est assimilée à une cuve fictive appelée cuve 0, et la cuve de déchargement à la cuve $n+1$;

d_i le temps de déplacement du robot en charge entre les cuves i et $i+1$ de la ligne. Ce temps peut comprendre les temps de levée et descente du porteur par le robot et le temps d'égouttage s'il y en a $(i = 0, ..., n)$;

$c_{i,j}$ le temps de déplacement du robot à vide entre deux cuves quelconques i et j de la ligne, avec $d_{i,j} = d_{j,i}$ pour tout i et j, $(j, i = 0, ..., n+1)$;

L_i la durée minimale de trempe dans la cuve i $(i = 1, ..., n)$;

U_i la durée maximale de trempe dans la cuve i $(i = 1, ..., n)$;

M une valeur très grande et positive ;

δ mesure de sécurité entre deux robots adjacents partageant le même rail pour éviter une collision. Dans un but de simplification, cette mesure est exprimée en secondes. Généralement, elle est égale au rapport entre une distance de sécurité entre les deux robots et la vitesse du robot. Cette dernière est généralement la même pour les deux robots ;

Les variables du problème sont :

T la durée d'un cycle ;

t_i la date de retrait d'un porteur de la cuve i au cours d'un cycle $(i = 0, ..., n)$. L'ensemble des t_i définit donc les dates de début de tous les transports effectués au cours d'un cycle donné ;

$y_{i,j}$ une variable binaire définie par :

$y_{i,j}$ =1 si *move* i est effectué avant *move* j, pour $i, j = 1, ..., n$, $i \neq j$

=0 sinon

Nous désignons aussi par T_l et T_u, respectivement, une borne inférieure et

une borne supérieure pour T.

1.5.2 Approches de résolution pour le CHSP mono-robot

Plusieurs méthodes ont été appliquées depuis l'article de W.Phillips et S.Unger (1976) pour résoudre les problèmes d'ordonnancement des ateliers de traitement de surface, notamment le CHSP mono-robot. Parmi les méthodes dédiées à la recherche d'ordonnancement 1-cyclique, nous citons :

- la programmation linéaire (W.Phillips et S.Unger, 1976; Song *et al.*, 1993, et, Liu *et al.* 2002) ;
- la programmation logique sous contraintes (Manier, 1994) ;
- les méthodes de type séparation-évaluation (Shapiro et Nuttle, 1988; Lei et Wang, 1989a, et, Chen *et al.* 1998) ;
- les heuristiques (Lei, 1993; Levner *et al.*, 1997) ;
- les métaheuristiques (Lim, 1997) ;

Nous décrirons dans ce qui suit, ces approches :

1.5.2.1 La Programmation Linéaire

W.Phillips et S.Unger (1976) des laboratoires Bell, en collaboration avec la Western Company de Richmond, ont été les premiers chercheurs à s'intéresser à la modélisation de ce problème. Leurs travaux ont servi de référence pour les recherches effectuées par la suite. Ils présentent un modèle pour le problème à un seul robot, avec des cuves mono-bac numérotées de 1 à c, plus une cuve fictive 0 représentant le poste de chargement et de déchargement. L'entrée d'un porteur sur la ligne correspond à la fin de son traitement dans la cuve 0 donc à la fin de son chargement. Sa date de sortie est le moment où le robot le dépose dans la cuve 0 donc le début de son déchargement. La solution optimale du problème est cherchée sous forme d'un ensemble ordonné de dates d'exécution des opérations de transport effectuées au cours d'un cycle.

Phillips et Unger ont construit un modèle du système comportant $(n+1)$ $(n+2)/2$ variables et $(n+1)$ contraintes (sous forme d'inéquations). la fonction objectif permet de minimiser la durée d'un cycle T.
Ils ont défini trois types de contraintes :

- le premier fixe les valeurs de t_{max} et des z_i. Ainsi, le numéro de la cuve où a lieu la dernière opération d'un cycle est identifié ;
- le deuxième assure que le robot ne peut exécuter qu'une seule opération de transport à la fois. Ces contraintes définissent alors les $y_{i,j}$;
- le troisième assure que le temps de traitement dans chaque cuve ne dépasse pas les durées minimale et maximale de trempe.

Pour la résolution de ce problème, les auteurs ont adopté le MPSX-MIP (Mathematical Programming System Extended-Mixed Integer Programming) qui utilise une approche arborescente de type séparation-évaluation. Ils ont testé leur modèle sur un benchmark correspondant à un cas industriel comportant une douzaine de cuves. Toutefois, ils fixent a priori le nombre de porteurs sur la ligne pendant un cycle, ce qui les empêche de trouver la solution optimale pour cette instance. Enfin, ils ont proposé une extension du problème au cas des lignes comportant des cuves multi-fonctions.

Song *et al.* (1993) ont développé un Programme Linéaire Mixte (PLM) pour décrire une ligne de traitement de surface avec des ressources restreintes. Le modèle proposé prend en considération le nombre de tâches (jobs) présentes dans la ligne. Ils ont proposé une heuristique pour la résolution de ce modèle. Cette heuristique permet de déterminer la date de début de traitement d'une tâche le plus tôt possible (x_j : la date de début de traitement de la tâche j). Le calcul des x_j dépend des dates de début des tâches antérieures (x_{j-1}) et de l'inexistence de conflits de robot c'est-à-dire la non satisfaction des contraintes de capacités des robots. Si un conflit existe, alors un incrément est déterminé par une procédure afin de régler la valeur de x_j. Cette procédure est appliquée

jusqu'à l'élimination de tout conflit.

Liu *et al.* (2002) ont développé un PLM pour le problème simple du CHSP mais avec possibilité d'attente d'un porteur au dessus d'une cuve. Dans leur formulation, ils ont introduit une nouvelle variable w_i pour $i = 0, ..., n$ qui représente la différence entre le temps réel et le temps minimum d_i pour le déplacement du robot entre la cuve i et la cuve $i + 1$. Ils ont montré que l'introduction des variables w_i dans le programme linéaire permet d'améliorer la durée du cycle T par rapport au modèle de W.Phillips et S.Unger (1976). Ensuite, ils ont proposé deux extensions : une première extension pour le problème avec des cuves multi-bacs et une deuxième extension pour les cuves multi-fonction.
Pour le cas du CHSP simple, le programme linéaire de Liu *et al.* (2002) est présenté comme suit :

$$\text{Minimiser} \quad T \tag{1.1}$$

S/c :

$$L_0 \leq T - t_n - d_n - w_n \leq U_0, \tag{1.2}$$

$$d_i + L_{i+1} - M(1 - y_{i,i+1}) \leq t_{i+1} - t_i - w_i \leq d_i + U_{i+1} \tag{1.3}$$

$$+M(1 - y_{i,i+1}), \quad i = 0, 1, ..., n - 1,$$

$$d_i + L_{i+1} - My_{i,i+1} \leq t_{i+1} + T - t_i - w_i \leq d_i + U_{i+1} \tag{1.4}$$
$$+My_{i,i+1}, \quad i = 1, ..., n-1,$$
$$t_i \geq d_0 + w_0 + c_{q_1,q_i}, \quad i = 1, ..., n, \tag{1.5}$$
$$t_j - t_i \geq d_i + w_i + c_{q_{i+1},q_j} - (1 - y_{i,j})M, \quad i, j = 2, ..., n; \quad i < j, \tag{1.6}$$
$$t_i - t_j \geq d_j + w_j + c_{q_{j+1},q_i} - y_{i,j}M, \quad i, j = 2, ..., n; \quad i < j, \tag{1.7}$$
$$T - t_i \geq d_i + w_i + c_{q_{i+1},q_0}, \quad i = 1, ..., n, \tag{1.8}$$
$$T \geq 0, \tag{1.9}$$
$$t_i \geq 0, \quad w_i \geq 0, \quad i = 1, ..., n, \tag{1.10}$$
$$y_{i,j} \in \{0, 1\}, \quad i, j = 1, 2, ..., n, \quad i < j. \tag{1.11}$$

Où q_i : désigne le numéro de la cuve utilisée pour exécuter le traitement i ; Dans ce modèle, la fonction objectif est la minimisation de la durée du cycle. Les contraintes (1.2)-(1.4) garantissent que les temps de traitement dans chaque cuve i, $(i = 0, ..., n)$ sont bornés par les durées minimales (L_i) et maximales (U_i). Les contraintes (1.5)-(1.8) assurent que le robot a suffisamment de temps pour se déplacer à vide entre deux mouvement en charge. Les contraintes (1.9)-(1.11) sont les contraintes de non-négativité et les contraintes binaires.

1.5.2.2 Programmation Logique sous contraintes

Dans sa thèse, Manier (1994) a modélisé le problème du HSP simple en se basant sur le modèle proposé par Shapiro et Nuttle (1988). Elle a proposé une nouvelle approche basée sur la programmation Logique avec contraintes pour la recherche d'une solution optimale. Dans ses travaux, elle a défini un ensemble de disjonctions concernant les contraintes qui portent sur le disponibilité du robot. Elle a défini des intervalles de temps $(I_{i,k})$ associés à un déplacement $(move\ i)$ d'un porteur k. Une autre condition a été ajoutée à ces intervalles. Elle impose que deux opérations de transfert doivent êtres séparées par le temps nécessaire au robot pour se déplacer à vide entre la cuve de dépôt du porteur sur lequel porte la

première opération et la cuve de retrait du porteur subissant le second transport. Des contraintes disjonctives relatives à ces intervalles de temps ont été définies selon les relations de précédence possibles entre deux opérations *move* i et *move* j. La méthode de résolution proposée consiste à chercher des ordonnancements réalisables en utilisant une procédure arborescente de type séparation-évaluation. Ensuite, Manier (1994) a donné des extensions de son modèle initial pour traiter les cas des lignes avec cuves multi-bacs, multi-fonctions, multi-robots, avec diverses configurations de gammes et de postes de chargement et déchargement. Enfin, elle a proposé une extension au cas n-cyclique.

1.5.2.3 Branch and Bound

Shapiro et Nuttle (1988) ont pris le même exemple que celui de Phillips et Unger exception faite pour les postes de chargement et déchargement qui sont dissociés. Ils ont proposé un modèle mathématique dans lequel ils ont pris en compte les temps réels de trempe o_i dans chaque cuve. Ainsi, ils utilisent n variables pour les n cuves de traitement (en ne considérant pas les cuves de chargement et déchargement). Ils forment ainsi un vecteur $O = (o_1, ..., o_n)$ à partir duquel sont déduites les dates $t(Re_i)$ de retrait des cuves i $(i = 1, ..., n)$ du premier porteur entrant dans la ligne et par conséquent le temps de traitement total d'un porteur noté T_t : $t(Re_i) = \sum_{j=1}^{i}(d_{j-1} + o_i), \quad pour \quad i = 1, ..., n$

$avec \quad t(Re_0) = 0 \quad et \quad t(Re_{n+1}) = \infty$

$T_t = t(Re_n) + d_n + C_{n+1,0}$

Cette dernière relation inclut le retour du porteur de la cuve de déchargement vers la cuve de chargement pour transporter un nouveau porteur. Il introduisent des vecteurs $V_j = v_j(0), ..., v_j(n-1)$ déterminant l'ordre d'exécution des opérations de retrait des cuves pour les porteurs 1 et $j + 1$. Ainsi, la notation $v_j(i) = k$ signifie que le $j + 1^{ème}$ porteur subit l'opération Re_i entre la $k^{ème}$ et la

$k+1^{ème}$ opération du porteur 1.

Le programme mathématique proposé comporte $n+1$ variables qui sont la durée du cycle T et les durées réelles de traitement dans chaque cuve o_i pour $i = 1, ..., n$. Il comporte aussi $(2p+1).n$ contraintes où p est le nombre de porteurs présents sur la ligne.

Ensuite, les auteurs ont proposé un algorithme de résolution basé sur une méthode par séparation-évaluation (Branch & Bound). Le $j^{ème}$ niveau de l'arbre est constitué de tous les ordres partiels possibles entre les opérations des j premiers porteurs. Si le système des contraintes est vérifié pour un ordre partiel donné représenté par $V_1, ..., V_j$, alors un fils de ce noeud, appelé **extension** est noté $V_1, ..., V_j, V_{j+1}$ pour être testé à son tour. Une feuille de l'arbre est caractérisée par la relation : $V_j(0) = n$, ce qui signifie que le premier porteur sort de la ligne avant que le $j+1^{ème}$ porteur n'y entre.

Lei et Wang (1989a) proposent une méthode utilisant la procédure par séparation et évaluation pour la résolution des problèmes d'ordonnancement cycliques des lignes de traitement de surface cyclique. Cette méthode permet de trouver des cycles n-périodiques pour des lignes simples. Pour trouver un ordonnancement optimal, la fonction objectif proposée est la minimisation de la durée du cycle T sujet à un ensemble de contraintes portant sur l'ordre d'exécution des opérations de transport et les dates d'exécution de ces opérations.
Dans la modélisation de leur modèle, les auteurs prennent en considération le nombre des tâches J présentes sur la ligne. Ils définissent une fonction d'ordonnancement $\Phi(j,i) = k\,(1 \leq k \leq J(n+1))$ qui affecte un numéro k à l'opération de transfert depuis la cuve i $(0 \leq i \leq n)$ d'un porteur j $(1 \leq j \leq J)$. Ils définissent aussi deux autres fonctions $\Phi_1(k) = j$ et $\Phi_2(k) = i$ qui renvoient le numéro du porteur et la cuve d'origine de l'opération de transport ayant pour

ordre k.

Chen *et al.* (1998) ont proposé un modèle ainsi qu'un algorithme de résolution pour trouver un ordonnancement cyclique optimal pour les mouvements du robot. Ils ont proposé deux procédures de type séparation et évaluation. Une première procédure nommée "procédure A" énumère toutes les distributions possibles des cartes de circuits imprimées (PCB) dans un cycle, à savoir, si la cuve contient une carte au début du cycle ou non. Ensuite, une deuxième procédure nommée "procédure B" énumère implicitement les séquences des mouvements du robot pour chaque distribution initiale potentiellement réalisable.

1.5.2.4 Heuristiques

Lei (1993) a développé une méthode qui permet de déterminer les dates de début des opérations de transport pour un ordonnancement cyclique donné. Pour cela, l'auteur a défini un programme linéaire en nombres entiers présenté ci-après :

$$\text{Minimiser} \quad T = t_{[n+1]} \tag{1.12}$$

S/c :

$$L_{[k]} \leq t_{[k]} - (L_{p[k]} + d_{p[k]}) \qquad \text{si} \quad R_{p[k]} \rightarrow R_{[k]} \tag{1.13}$$

$$t_{[k]} - (L_{p[k]} + d_{p[k]}) \leq U_{[k]} \qquad \text{si} \quad R_{p[k]} \rightarrow R_{[k]} \tag{1.14}$$

$$L_{[k]} \leq T - (L_{p[k]} + d_{p[k]}) + t_{[k]} \qquad \text{si} \quad R_{p[k]} \rightarrow || \rightarrow R_{[k]} \tag{1.15}$$

$$T - (L_{p[k]} + d_{p[k]}) + t_{[k]} \leq U_{[k]} \qquad \text{si} \quad R_{p[k]} \rightarrow || \rightarrow R_{[k]} \tag{1.16}$$

$$t_{[i]} + d_{[i]} + c[i] + 1, [i+1] \leq t_{[i+1]} \qquad \text{pour tout} \quad i = 1, ..., n \tag{1.17}$$

$$T, t_{[1]}, ..., t_{[n]} \in \aleph$$

où

$[k]$: la $k^{ème}$ opération effectuée par le robot. Par exemple : $R_{[k]} = R_i$ indique que R_i (transport d'un porteur de la cuve i vers la cuve $i+1$) est la $k^{ème}$ opération effectuée par le robot dans un cycle ;
$p[k]$: indique l'opération précédent k ;
$R_{p[k]} \rightarrow R_{[k]}$: les deux opérations sont effectuées au cours d'un même cycle ;
$R_{p[k]} \rightarrow || \rightarrow R_{[k]}$: les deux opérations ne sont pas effectuées au cours d'un même cycle.

Les contraintes (1.13)-(1.14) assurent que les temps de traitement dans une cuve à laquelle les opérations de retrait et de dépôt d'un porteur s'effectuent dans le même cycle, sont bornés par les durées de traitement minimales et maximales. Les contraintes (1.15)-(1.16) assurent que les temps de traitement dans une cuve à laquelle les opérations de retrait et de dépôt d'un porteur s'effectuent dans deux cycles consécutifs, sont bornés par les durées de traitements minimales et maximales. Les contraintes (1.17) garantissent que le robot doit avoir suffisamment de temps pour se déplacer à vide entre deux mouvements $R_{[i]}$ et $R_{[i+1]}$.

Pour résoudre ce problème, Lei (1993) a introduit une procédure algébrique en utilisant la recherche dichotomique (binary search). L'idée de cette procédure est de chercher une valeur pour T qui vérifie les contraintes précédentes. Cette procédure consiste à déterminer une borne inférieure T_l et une borne supérieure T_u pour T. Soit T_1 une première valeur générée comme suit : $T_1 = (T_u - T_l)/2$. Si la valeur de T_1 satisfait les contraintes précédentes alors $T_u = T_1$ Sinon $T_l = T_1$. Ainsi, les différentes valeurs T_2, T_3,... seront définies comme suit :

$$T_i = \left[\left(T_u^{(i-1)} - T_l^{(i-1)} \right) /2 \right]$$

où $T_l^{(i-1)}$ et $T_u^{(i-1)}$ sont, respectivement, les bornes inférieures et supérieures des durées de cycle obtenues dans l'étape (i-1). Le but de la méthode est de définir les différentes valeurs pour T_i afin de trouver la valeur minimale qui satisfait les

contraintes précédentes.

Levner *et al.* (1997) ont résolu le problème cyclique mono-robot par un algorithme exact. Cet algorithme permet la résolution de ce problème en un temps polynomial O(n^3 log n). Ils ont développé un programme mathématique comportant des contraintes linéaires et quadratiques. Ils ont proposé une relation entre les variables m_i, t_i et T définie comme suit : $m_i = t_i + s_i T$ où m_i est la date de début de déplacement du premier porteur depuis la cuve i. Ainsi l'ordre des opérations du robot dépend de T. Ce programme ne peut pas être résolu en un temps polynomial. Pour cela, les auteurs l'ont reformulé en éliminant les contraintes quadratiques et la dépendance des indices d'ordres (s_i) avec T. Ils ont utilisé une approche par intervalle pour la détermination de T en un temps polynomial. L'algorithme consiste à chercher une valeur de T telle que les valeurs T_1, T_2, T_3,...T_n n'appartiennent pas aux intervalles suivants $I_k = \left\{]\frac{m_i - m_j - d_j - c_{i,j}}{k}, \frac{m_i - m_j + d_j + c_{i,j}}{k}[\right\}$ avec $i > j$ et $i, j, k = 0, ..., n$. Une fois la durée du cycle T trouvée, les valeurs de t_i sont calculées en fonction de la relation $t_i = m_i \bmod T$ et l'ordonnancement des opérations du robot est défini, par la suite, en triant dans l'ordre croissant les valeurs de t_i. Cet algorithme a été testé sur un exemple de ligne dans l'industrie électronique composée de 7 cuves y compris les cuves de chargement et déchargement.

Che et Chu (2005) ont aussi développé un algorithme polynomial pour résoudre les problèmes du CHSP simple. Ils ont proposé aussi la résolution du problème lorsque les cuves sont multi-bacs et multi-fonctions.

1.5.2.5 Métaheuristiques

Lim (1997) a proposé un algorithme génétique qui permet de trouver un ordonnancement proche de l'optimum. Partant d'une population initiale composée d'un ensemble de solutions (chromosomes), les parents sont sélectionnés sur la

base de la force (fitness) de chaque individu. La sélection se fait en utilisant le principe de la roulette qui attribue, pour chaque individu, une probabilité proportionnelle à sa force. L'auteur utilise l'algorithme de Lei (1993) pour calculer la durée du cycle qui représente la valeur de la force de chaque individu. Les descendants sont obtenus avec l'opérateur LOX (Line Order Crossover). Les nouveaux descendants remplacent les mauvaises solutions se trouvant dans la génération actuelle. Et, ainsi, une nouvelle génération de solutions est obtenue. Lim a testé son algorithme sur l'exemple de W.Phillips et S.Unger (1976). Après expérimentation, il a trouvé que la solution générée par son algorithme converge rapidement vers la solution optimale quand la taille de la population est 30. Toutefois, Lim (1997) ne précise pas la manière dont sont gérés les inévitables et nombreux chromosomes correspondant à des solutions non faisables. Le choix d'opérateurs non dédiés et l'absence de procédures de réparation rendent peu probable une convergence vers l'optimum, d'autant plus sur le benchmark considéré, particulièrement contraint.

1.6 Conclusion

Nous avons présenté, dans le premier chapitre, le problème d'ordonnancement en général en définissant ses différentes notions de base, et le problème d'ordonnancement d'atelier, en particulier. Plusieurs types d'atelier ont été étudiés dans la littérature dont les lignes automatisées de traitement de surface. Ce type d'atelier traite le changement d'aspect ou de forme de la surface des produits pour des besoins ou conditions d'utilisation. Le problème d'ordonnancement associés est connu dans la littérature sous l'appellation Hoist Scheduling Problem (HSP). Nous avons également donné dans ce chapitre une revue des différentes approches de résolution pour le HSP mono-robot, où on cherche un cycle de degré 1.
Les différents types de HSP sont souvent issus d'applications industrielles. C'est le cas des problèmes cycliques à plusieurs robots. L'augmentation du nombre

des robots partageant un même rail amplifie la difficulté du problème. Plusieurs auteurs se sont déjà intéressés à la résolution de ce genre de problème. Mais, ce dernier reste ouvert en terme d'amélioration soit au niveau de l'objectif des industriels à savoir l'augmentation de la productivité (minimisation de la durée du cycle), soit au niveau du temps de résolution (CPU). Le deuxième chapitre sera consacré aux approches de résolution du CHSP multi-robots et détaillera notre contribution dans ce domaine.

Chapitre 2

Le HSP Cyclique multi-robots : Etat de l'art et nouvelle méthode de résolution

Sommaire

2.1 Introduction

Lorsque le nombre de cuves devient très important dans une ligne de traitement de surface, les distances à parcourir s'en trouvent augmentées. Le robot peut alors devenir une ressource critique de l'atelier. En conséquence, l'amélioration de la productivité passe par un accroissement du nombre des robots pour assurer le transfert des porteurs. Le nombre des robots peut être déterminé empiriquement (Leung et Levner, 2006). Les auteurs ont proposé un algorithme polynomial pour la détermination du nombre minimal des robots pour les problèmes avec des temps de traitement fixes. Pour une durée de cycle T donnée, un ensemble partiellement ordonné (P_T) de mouvements conflictuels est défini. Deux mouvements sont conflictuels s'ils ne peuvent pas être effectués avec le même robot. Ils ont défini trois types de points de transitions. Un point de transition C est défini lorsque l'ensemble ordonné P_C diffère de l'ensemble ordonné $P_{C'}$ pour $C' = C + \varepsilon$ (ε une valeur très petite). Pour tout point de transition défini dans l'intervalle $[T_l, T_u]$, un ensemble ordonné est déterminé et la plus longue chaîne des ensembles ordonnés P_c correspond au nombre minimal de robots.

L'augmentation du nombre des robots dans une ligne unidirectionnelle génère des contraintes additionnelles pour assurer la bonne gestion des mouvements des robots (Manier, 1994). Il s'agit :

- d'assurer la meilleure répartition possible des opérations de transport en charge, en les affectant aux robots ;
- de gérer le risque de collision entre les différents robots partageant le même rail. La collision entre deux robots adjacents peut se produire soit :
  - entre deux robots en déplacement que ce soit en charge ou à vide ;
  - entre un robot en déplacement et un autre en position d'arrêt.

Le nombre de situations conflictuelles rend le problème plus difficile à

résoudre. Plusieurs auteurs se sont intéressés à la résolution de ce problème, en s'inspirant d'abord des travaux de Shapiro (1985), qui dans sa thèse, a adopté une stratégie de partition appelée ***zoning approach***, développée par la suite par Lei et Wang (1991). Celle-ci consiste à diviser l'ensemble des cuves en un nombre de sous-ensembles de cuves adjacentes, égale au nombre de robots. Chaque robot est alors affecté à un sous ensemble. Ainsi, entre deux sous-ensembles adjacents se trouve une cuve dans laquelle un robot vient placer un porteur et un autre le retire. Ces cuves constituent des ***points de transfert*** (Manier, 1994), appelées aussi cuves ***pivot*** par Lei et Wang (1991). Ce type de problème sera défini par la suite comme le ***problème sans zone de chevauchement***.
Dans sa thèse, Manier (1994) a traité le cas général où les robots peuvent partager des zones communes non réduites aux cuves pivot. Chaque robot est affecté non pas à un ensemble de cuves mais à un ensemble ***d'opérations de transfert*** des porteurs. Chaque opération est affectée à un seul robot. Ce type de problème sera défini par la suite comme le ***problème avec zone de chevauchement***.
Plusieurs méthodes ont été développées pour la résolution de ces deux types de problèmes.

Dans la suite de ce chapitre nous présenterons un état de l'art sur les méthodes de résolution de ces deux types de problèmes (**section 2.2**). Dans la **section 2.3**, nous proposons une approche hybride pour la résolution des problèmes **CHSP** avec deux robots. Cette heuristique est validée par des tests réalisés sur plusieurs instances, les résultats obtenus étant discutés dans la **section 2.4**.

2.2 Etat de l'art sur le CHSP multi-robots

Lei et Wang (1991) ont étudié le problème d'une ligne comportant deux robots. Ils ont proposé un algorithme appelé l'algorithme du plus petit cycle

commun (Minimum Common Cycle), de complexité $O(n.(n-1)!)$.
Le principe de l'algorithme consiste à trouver une partition de la ligne en deux zones, et d'affecter chaque robot à une zone. Ainsi, la première zone comporte les $V+1$ premières cuves et la deuxième zone les $n-V$ dernières cuves. De ce fait, la cuve $V+1$ est considérée comme la cuve de déchargement pour la première zone (zone 1) et comme la cuve de chargement pour la deuxième zone (zone 2). A chaque zone est associé un problème mono-robot qui est résolu indépendamment de l'autre par un algorithme d'optimisation. Deux durées de cycles sont déterminées, pour les deux sous-problèmes. Si les deux durées ne sont pas égales, alors une résolution est recommencée pour la zone ayant la petite durée de cycle. Et ainsi, la durée la plus grande sera considérée comme une borne inférieure pour le sous problème à résoudre. Une fois que les deux ordonnancements obtenus ont un temps de cycle commun, il suffit de synchroniser les mouvements des deux robots au niveau de la cuve $V+1$ afin que le temps de traitement dans celle-ci soit respecté.
Pour chaque valeur de V ($V = 0, ..., n-1$), $T(V)$ est définie comme étant la plus petite durée du cycle commun aux deux sous-problèmes. Ainsi, la partition optimale sera définie pour la plus petite valeur de $T(V)$.

Manier *et al.* (2000) ont proposé un modèle de résolution pour les problèmes de HSP cyclique multi-robots basé sur la programmation logique sous contraintes. Ils ont proposé une nouvelle partition, non pas au niveau des cuves de traitement, mais plutôt portant sur les opérations de transfert affectées aux robots. Le modèle proposé est moins restrictif que celui de Lei et Wang (1991) du fait qu'il permet de résoudre des problèmes plus généraux, c'est-à-dire dans lesquels les zones de déplacement communes des robots ne se limitent plus à une seule cuve. Cela autorise donc des configurations avec des cuves multi-fonctions, et où les produits ne se déplacent pas forcement de manière unidirectionnelle sur la ligne. L'objectif de ce travail étant de proposer une procédure de résolution

pour une partition déjà donnée. Les auteurs ont identifié un ensemble de disjonctions associées à toutes collisions potentielles entre deux opérations de transport effectuées par deux robots adjacents. Ils définissent des règles pour toutes les situations possibles, permettant d'arbitrer ces disjonctions. Ce modèle a été implémenté en utilisant la programmation logique sous contraintes et résolu avec Prolog IV.

Un PLM pour la résolution du CHSP avec plusieurs robots partageant le même rail et avec zone de chevauchement a été proposé par Leung et Zhang (2003). Ils ont défini un ensemble de contraintes qui permettent la non collision entre les robots en se référant aux travaux de Heinrichs et Moll (1997).

Ultérieurement, Leung *et al.* (2004) ont proposé un autre PLM pour le problème CHSP multi-robots et avec zone de chevauchement, dans le cas où la gamme opératoire suit l'implantation des cuves. Ensuite, ils ont utilisé la relaxation continue pour ce PLM, tout en ajoutant les inégalités triangulaires proposées par Grötschel *et al.* (1985) qui s'expriment par la relation suivante :

$$0 \leq y_{i,j} + y_{j,k} - y_{i,k} \leq 1 \qquad \text{pour tout} \quad i, j, k = 1, ..., n \tag{2.1}$$

Les auteurs ont montré que la relaxation du programme linéaire ainsi que l'ajout des inégalités valides permettent de réduire le nombre des nœuds branchés lors de la résolution du programme ainsi que les temps de résolution (CPU). Ils ont montré aussi, par des expérimentations, que le choix de la borne supérieure utilisée comme paramètre (M) dans le programme influe aussi sur le nombre des nœuds branchés et les temps de résolution (CPU). Ainsi, deux bornes supérieures ont été définies comme suit :

- $UB_1 = d_0 + \sum_{i=1}^{n}(d_i + L_i)$
- la deuxième borne UB_2 est obtenue en déterminant la durée du cycle par

application de l'algorithme de Levner *et al.* (1997) mono-robot et en utilisant comme durées de traitement fixes les bornes inférieures des durées de traitement L_i.

Les expérimentations ont prouvé que la meilleure valeur M est la borne supérieure UB_1, borne par ailleurs assez classiquement utilisée.

Liu et Jiang (2005) ont proposé un algorithme polynomial pour la résolution du CHSP à deux robots et avec des temps de traitement fixes. Sous cette hypothèse nous pouvons dire que le problème réellement considéré est connexe au HSP. Cet algorithme permet la résolution du problème en un temps de calcul limité à $O(n^4 \log n)$ et permet de fournir deux ordonnancements pour les deux robots avec zone de chevauchement. Ils ont défini des intervalles de temps pour limiter le domaine de recherche des solutions réalisables. Pour la détermination de ces intervalles de temps, ils se sont basés sur les travaux de Levner *et al.* (1997). A partir de ces intervalles, ils définissent différentes valeurs de T, et pour chaque valeur de T, les dates t_i seront déterminées. Ensuite, ils proposent une procédure d'affectation qui permet d'assigner chaque mouvement à l'un des deux robots. Si la faisabilité de la solution est vérifiée alors la solution optimale est trouvée.

Zhou et Liu (2008), partant du même principe que l'algorithme présenté par Liu et Jiang (2005), proposent une heuristique pour la résolution du problème du CHSP à deux robots. Ils définissent dans une première étape les différentes séquences, pas nécessairement faisables, pour le problème à un robot. Ensuite, pour chaque séquence, une procédure d'affectation est appliquée pour attribuer chaque opération à l'un des deux robots. Un PLM est développé afin de tester la faisabilité des deux ordonnancements mono-robot trouvés et d'ajuster les valeurs des variables. Le PLM proposé contient l'ensemble des contraintes du CHSP à savoir les contraintes sur les temps de traitement, les contraintes sur

la disponibilité des robots et les contraintes sur la non collision entre les deux robots. Ces dernières sont définies cas par cas selon les positions des cuves pour les deux mouvements de transport à effectuer. Vingt différents cas ont été illustrés par les auteurs. Selon les deux ordonnancements obtenus par la procédure d'affectation, uniquement une partie de ces contraintes est considérée dans le MILP.

Che et Chu (2004) ont proposé une méthode de résolution basée sur une approche par séparation et évaluation pour le problème du HSP multi-robots, où les postes de chargement et déchargement sont dissociés. Dans leur article, ils ont développé tout d'abord un modèle mathématique du problème considéré dans les deux cas où :

- deux robots adjacents partagent des zones de chevauchement (problème avec zones de chevauchement) ;
- deux robots adjacents se partagent la même cuve (problème sans zone de chevauchement).

Dans le premier cas, les auteurs définissent un ensemble de disjonctions sur la base desquelles les contraintes de non-collision ont été développées. En effet, en considérant deux mouvements de transport *move* i et *move* j tels que $t_i < t_j$, deux cas de figure peuvent se présenter :

- $i > j$ et $R_{r_i} < R_{r_j}$ c'est-à-dire le robot R_{r_i} se trouve à gauche du robot R_{r_j} (voir Figure 2.1(a)) ;
- $i < j$ et $R_{r_i} > R_{r_j}$ c'est-à-dire le robot R_{r_i} se trouve à droite du robot R_{r_j} (voir Figure 2.1(b)).

Avec R_{r_i} : robot numéro r_i, et r_i : numéro du robot ayant effectué *move* i sachant que les robots sont indexés dans le même ordre que l'indexation des cuves.

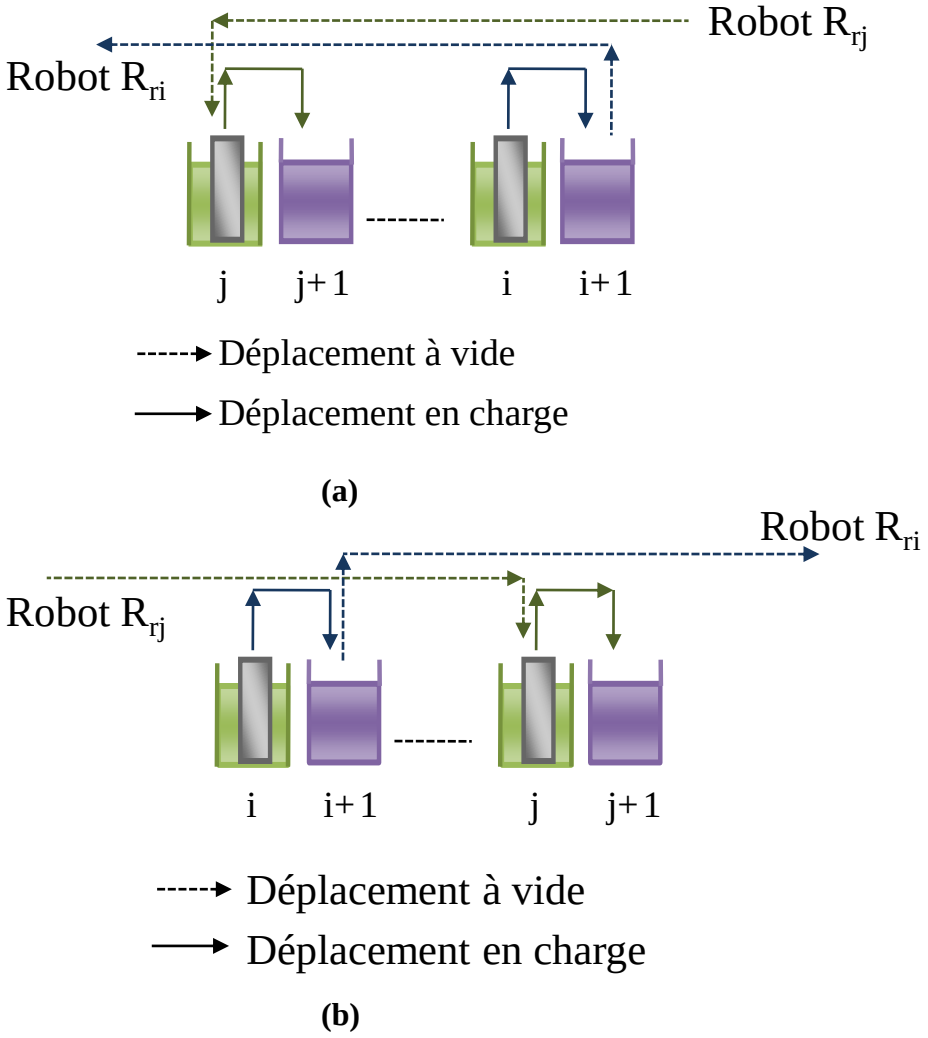

FIGURE 2.1 – Partage de zone entre deux robots adjacents

Pour éviter toute collision, il faut que le robot r_i, après avoir effectué *move* i, arrive au dessus de la cuve j avant le robot r_j. Ainsi, les auteurs ont défini les deux contraintes suivantes :

$$t_i + d_i + c_{i+1,j} + |R_{r_i} - R_{r_j}|\delta \leq t_j, \quad \text{pour tout } t_i < t_j$$

$$t_j + d_j + c_{j+1,i} + |R_{r_i} - R_{r_j}|\delta \leq T + t_i, \quad \text{pour tout } t_i < t_j$$

δ représente la mesure de sécurité entre deux robots définie dans le chapitre précédent dans la **section 1.5.1**.

Les deux auteurs ont montré que ces deux contraintes sont nécessaires mais pas suffisantes pour garantir la non-collision entre les deux robots. Pour cela, ils ont ajouté deux propriétés comme règles de vérification liées aux mouvements des deux robots. Ensuite, les auteurs ont réalisé une relaxation sur ces deux propriétés, car leur traitement lors de la conception de l'algorithme de résolution proposé est très difficile. Cet algorithme est basé sur deux approches de type séparation-évaluation (*branch and bound*), nommées procédure **A** et procédure **B**. La première consiste à énumérer toutes les configurations possibles pour les cuves. Chaque nœud est étiqueté par la notation $Z_i = (0, R_{r_i})$ ou $Z_i = (1, R_{r_i})$. La première indique que la cuve i est vide au début du cycle et que *move* i est effectué par le robot r_i, tandis que la deuxième indique que la cuve i est chargée au début du cycle et que *move* i est effectué par le robot r_i. Le branchement d'un nœud correspondant à la cuve i consiste à tester toutes les configurations possibles pour la cuve $i + 1$. Les auteurs définissent quelques règles de priorité pour éliminer les nœuds infaisables. Le $n^{ème}$ niveau de l'arbre correspond à une distribution partielle notée Z_n des cuves (ordonnancement partiel des mouvements). Arrivant au $n^{ème}$ niveau de l'arbre, la procédure **B** est activée, dans laquelle les risques de collision sont gérés. Cette dernière consiste à énumérer les différentes relations de précédence entre deux mouvements de robots. Le branchement au niveau de chaque nœud dépend de la solution du programme linéaire proposé par les auteurs. S'il existe une solution mais qui dépasse la valeur d'une borne supérieure ou bien s'il n'y a pas de solution, alors le nœud est éliminé. Sinon, le branchement est effectué au niveau de ce nœud.
Pour la détermination des bornes inférieures et la détection des infaisabilités des solutions au niveau des procédures A et B, les auteurs ont formulé deux programmes linéaires. Le premier relaxé pour la procédure A et le deuxième

pour la procédure B. Pour la résolution de ces deux programmes, les auteurs ont utilisé l'algorithme polynomial proposé par Chen *et al.* (1998).

Plusieurs auteurs ont proposé des métaheuristiques pour la résolution du CHSP multi-robots. Parmi ces travaux, Yang *et al.* (2001) ont proposé une méthode de résolution basée sur le recuit simulé pour le problème sans zone de chevauchement. Les cuves de chargement et déchargement sont dissociées. Ce travail consiste à décomposer, arbitrairement, la ligne en m partitions ($P = \Omega_1, ..., \Omega_m$). Les auteurs ont proposé, ensuite, un programme linéaire $SchP_m$ pour chaque partition comportant des contraintes sur les temps de traitement, les temps de transfert du robot et l'affectation des opérations au robot R_k avec $k = 1, ..., K$ (K est le nombre des robots). Ainsi, pour chaque partition P, $SchP_m$ est un problème mono-robot avec $m = 1, ..., K$. Ils ont appliqué l'algorithme de Armstrong *et al.* (1994) pour la résolution du sous-problème $SchP_m$. Ensuite, dans une deuxième étape, ils ont proposé un deuxième algorithme basé sur le recuit simulé pour la résolution du problème multi-robots. Cet algorithme consiste à :

1. Sélectionner une partition initiale P_0 soit :
   - en décomposant l'ensemble des cuves en K zones tel que le nombre de cuves dans chaque zone soit le même ;
   - en sélectionnant le nombre de cuves dans chaque zone au hasard.
2. Définir une fonction qui permet de générer d'autres partitions que P_0 (des voisins de P_0). Les auteurs ont utilisé quatre méthodes pour générer les voisins de chaque partition :
   - Changer la zone de goulot d'étranglement (ayant une durée de cycle très grande), et la zone ayant une durée du cycle faible ;
   - Changer la zone de partition P_0 aléatoirement, cela revient à changer les cuves limitant chaque zone ;
   - Changer aléatoirement les cuves limitant les zones de goulot d'étranglement ;

– Chercher aléatoirement une nouvelle partition de P.

Manier et Lamrous (2008) ont proposé un algorithme évolutionnaire pour la résolution des problèmes cycliques multi-robots avec zones de chevauchement. Les cuves de chargement et déchargement sont associées. Le codage des solutions est fait sur la base des mouvements du robot à vide contrairement aux codages habituels qui représentent les mouvements du robot en charge. Une procédure constructive est définie pour la construction d'un chromosome. La population initiale est générée aléatoirement, composée d'un ensemble de chromosomes de plusieurs tailles. Les auteurs proposent un PLM pour l'évaluation de la force (*fitness*) de chaque chromosome. Ainsi, la force de chaque solution est mesurée par le produit du temps de cycle et le nombre de robots ($nh.T$). Un sous ensemble de solutions est sélectionné en adoptant la méthode élitiste et la sélection universelle stochastique. La descendance est obtenue en adoptant un croisement à 2-points et une mutation qui consiste à permuter deux gènes sélectionnés aléatoirement. Après mutation, ce nouvel individu remplace le chromosome parent dans la prochaine population si la solution produite est réalisable. Sinon, les auteurs utilisent deux procédures de réparation. Mais, cet algorithme présente l'inconvénient qu'il ne gère pas les collisions entre les robots.

Dans la suite, nous présenterons le modèle développé par Leung *et al.* (2004). C'est un PLM qui contient l'ensemble des contraintes relatives aux problèmes CHSP multi-robots et qui ont été définies dans la **section 1.3.2**.

Leung *et al.* (2004) ont proposé un PLM pour le problème cyclique multi-robots. La gamme de production est la même que l'ordre des cuves c'est-à-dire chaque produit doit passer par la cuve numéro 1 ensuite doit être transporté à la cuve numéro 2 puis la cuve numéro 3, ... jusqu'à la cuve de déchargement. K robots seront numérotés consécutivement par R_k $(k = 1, ..., K)$ et celui le plus proche de la cuve 0 sera le robot R_1.
Notre intérêt a été tout particulièrement accordé à cette formulation étant

donnée l'exhaustivité de ses contraintes. Lorsque le nombre des robots sur une ligne est égale à 2, le modèle proposé comporte $11.n^2 - n + 3$ contraintes et $n^2 + 4.n + 3$ variables dont $n^2 + 3.n + 1$ variables binaires.

Les variables utilisées dans le modèle sont définies par :

T La durée d'un cycle ;

t_i la date de retrait d'un porteur de la cuve i au cours d'un cycle ($i = 0, ..., n$) ;

$$Z_i^k = \begin{cases} 1 & \text{si } move\ i \text{ est effectué par le robot } k\ (R_k), \text{ pour } i = 1, ..., n,\ k \in K \\ 0 & \text{sinon} \end{cases}$$

$$F_i = \begin{cases} 1 & \text{si } move\ i \text{ est le dernier mouvement pour le premier robot } (R_1), \\ & \text{pour } i = 0, ..., n \\ 0 & \text{sinon} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si } move\ i \text{ est effectué avant move } j, \text{ pour } i, j = 1, ..., n,\ i \neq j \\ 0 & \text{sinon} \end{cases}$$

$$s_i = \begin{cases} 1 & \text{si la cuve } i \text{ contient un porteur au début du cycle, pour } i = 1, ..., n \\ 0 & \text{sinon} \end{cases}$$

Le modèle proposé par Leung *et al.* (2004) est présenté ci-dessous :

$$\text{Minimiser} \quad T \tag{2.2}$$

S/c :

$$\sum_{k=1}^{K} Z_i^k = 1 \qquad \text{pour tout} \quad i = 1, ..., n \tag{2.3}$$

$$\sum_{j=0}^{n} F_j = 1 \tag{2.4}$$

$$L_0 + Z_i^1 \leq 1 \qquad \text{pour tout} \quad i = 1, ..., n \tag{2.5}$$

$$F_i \leq Z_i^1 \qquad \text{pour tout} \quad i = 1, ..., n \tag{2.6}$$

$$F_j + Z_i^1 - y_{ij} \leq 1 \qquad \text{pour tout} \quad i, j = 1, ..., n \tag{2.7}$$

$$t_i + d_i + c_{i+1,0} F_i \leq T \qquad \text{pour tout} \quad i = 0, ..., n \tag{2.8}$$

$$t_j - (d_0 + c_{1,j}) Z_j^1 \geq 0 \qquad \text{pour tout} \quad j = 1, ..., n \tag{2.9}$$

$$t_0 = 0 \tag{2.10}$$

$$t_i - (t_{i-1} + d_{i-1}) \leq U_i \qquad \text{pour tout} \quad i = 1, ..., n \tag{2.11}$$

$$t_i - (t_{i-1} + d_{i-1}) + M s_i \geq L_i \qquad \text{pour tout} \quad i = 1, ..., n \tag{2.12}$$

$$t_i + T - (t_{i-1} + d_{i-1}) - M(1 - s_i) \leq U_i \qquad \text{pour tout} \quad i = 1, ..., n \tag{2.13}$$

$$t_i + T - (t_{i-1} + d_{i-1}) \geq L_i \qquad \text{pour tout} \quad i = 1, ..., n \tag{2.14}$$

$$t_i - t_{i-1} - d_{i-1} + \delta - (U_i + \delta)(1 - s_i) \leq 0 \qquad \text{pour tout} \quad i = 1, ..., n \tag{2.15}$$

$$t_j - t_i \leq M y_{ij} \qquad \text{pour tout} \quad i, j = 1, ..., n, i \neq j \tag{2.16}$$

$$y_{ij} + y_{ji} = 1 \qquad \text{pour tout} \quad i, j = 1, ..., n, i \neq j \tag{2.17}$$

$$t_i + d_i + c_{i+1,j} - t_j \leq M \left(3 - y_{ij} - Z_i^k - \sum_{h=k}^{K} Z_j^h \right) \qquad \text{pour tout} \quad i, j = 1, ..., n, j < i, k \in K \tag{2.18}$$

$$t_j + d_j + c_{j+1,i} - t_i \leq M \left(3 - y_{ji} - Z_i^k - \sum_{h=k}^{K} Z_j^h \right) \qquad \text{pour tout} \quad i, j = 1, ..., n, j < i, k \in K \tag{2.19}$$

$$t_j + d_j + c_{j+1,i} - t_i \leq M \left(3 - y_{ji} - Z_i^k - \sum_{h=1}^{k} Z_j^h \right) \qquad \text{pour tout} \quad i, j = 1, ..., n, i < j, k \in K \tag{2.20}$$

$$t_i + d_i + c_{i+1,j} - t_j \leq M \left(3 - y_{ij} - Z_i^k - \sum_{h=1}^{k} Z_j^h \right) \qquad \text{pour tout} \quad i, j = 1, ..., n, i < j, k \in K \tag{2.21}$$

$$t_j + d_j + c_{j+1,i} - (T + t_i) \leq M \left(2 - Z_i^k - \sum_{h=k}^{K} Z_j^h \right) \qquad \text{pour tout} \quad i, j = 1, ..., n, j < i, k \in K \tag{2.22}$$

$$t_i + d_i + c_{i+1,j} - (T + t_j) \leq M \left(2 - Z_i^k - \sum_{h=k}^{K} Z_j^h\right)$$
$$\text{pour tout} \quad i, j = 1, ..., n, j < i, k \in K \tag{2.23}$$
$$t_j + d_j + c_{j+1,i} - (T + t_i) \leq M \left(2 - Z_i^k - \sum_{h=k}^{K} Z_j^h\right)$$
$$\text{pour tout} \quad i, j = 1, ..., n, i < j, k \in K \tag{2.24}$$
$$t_i + d_i + c_{i+1,j} - (T + t_j) \leq M \left(2 - Z_i^k - \sum_{h=k}^{K} Z_j^h\right)$$
$$\text{pour tout} \quad i, j = 1, ..., n, i < j, k \in K \tag{2.25}$$

Les auteurs ont défini quatre groupes de contraintes lors de la formulation de ce programme :

- Contraintes d'affectation des robots et définition de la durée du cycle : (contraintes (2.3) à (2.10))

  Les contraintes (2.3) assurent que chaque mouvement est affecté à un seul robot. La contrainte (2.4) définit le numéro de la cuve d'où part le dernier mouvement pour R_1. Si *move* 0 est le dernier mouvement pour R_1 (contraintes (2.5)) alors tous les mouvements (1,...,n) seront affectés aux robots situés au dessus de R_1 (pour $K = 2$ les mouvements seront affectés à R_2). Les contraintes (2.6) et (2.7) définissent la relation entre les variables binaires L_i, Z_i^k et y_{ij}. Les contraintes (2.8) et (2.9) assurent que la durée du cycle est assez longue pour permettre au robot R_1 de revenir vers la cuve 0 pour le début du cycle suivant.
- Contraintes sur les temps de traitement : (contraintes (2.11) à (2.12))

  Ces contraintes permettent d'assurer le respect des bornes inférieures et supérieures des temps de traitement dans chaque cuve. En effet, si $s_i = 0$, alors le traitement dans une cuve se fait au cours d'un même cycle et la durée de traitement est égale à $[t_i - (t_{i-1} + d_{i-1})]$; les contraintes (2.11) et (2.12) vérifient alors que la durée de traitement est dans les limites. Par contre, si $s_i = 1$ alors le traitement d'un porteur dans la cuve i se fait dans deux cycles successifs et la durée de traitement est égale à $[T - (t_{i-1} +$

$d_{i-1}) + t_i]$; les contraintes (2.13) et (2.14) assurent le respect des bornes des traitements de la cuve i.

Les contraintes (2.15) permettent d'assurer un temps de sécurité δ entre l'opération de levage et descente d'un porteur lorsque ces deux opérations coïncident dans le temps.

– Contraintes d'ordre : (contraintes (2.16) à (2.17))

Ces contraintes permettent de fixer les valeurs de y_{ij} en tenant compte des dates de début des mouvements t_i.

– Les contraintes (2.18) à (2.25) permettent d'éviter la collision entre les robots : considérons deux mouvements *move* i effectué par le robot R_k et *move* j effectué par le robot R_h.

Si R_h est situé à gauche de R_k ($k < h$) et $j < i$ alors les contraintes (2.18) et (2.19) assurent la non collision entre les deux robots. En effet, si move i est effectué avant *move* j c'est-à-dire $y_{ij} = 1$ alors les deux robots R_h et R_k doivent passer par la cuve j de façon que la date t_j doit être supérieure à la date $t_i + d_i + c_{i+1,j}$ comme le montre la Figure 2.2.

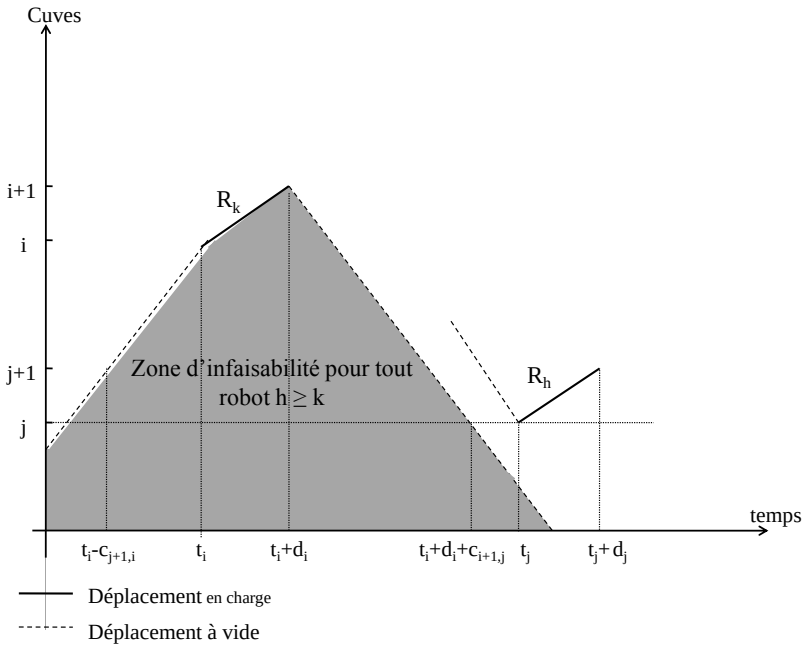

FIGURE 2.2 – Un diagramme de Gantt schématisant la contrainte (2.18)

Si move i est effectué après *move* j c'est-à-dire $y_{ji} = 1$ alors les contraintes (2.19) assurent que la date de début de *move* i (t_i) doit être supérieure à la date de fin de *move* j ($t_j + d_j$) d'au moins une durée égale à $c_{j+1,i}$.
De même, Si R_k est situé à gauche de R_h ($k > h$) et $i < j$ alors les contraintes (2.20) et (2.21) assurent la non collision entre les deux robots (voir Figure 2.3).

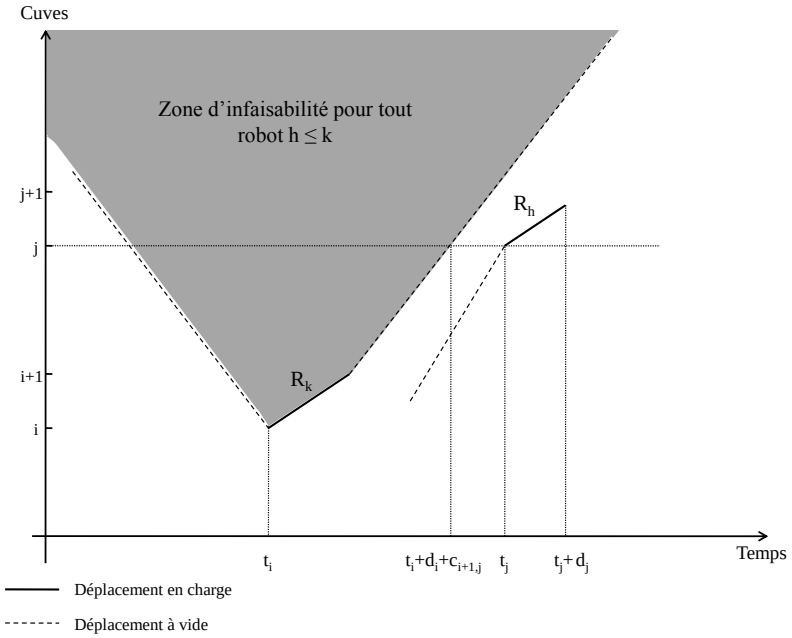

FIGURE 2.3 – Un diagramme de Gantt schématisant la contrainte (2.21)

Les contraintes (2.22) à (2.25) assurent que chaque robot a suffisamment de temps pour se déplacer sans se croiser avec les autres robots pour deux mouvements s'effectuant dans deux cycles consécutifs. Cette situation se produit lorsque un porteur commence son traitement dans une cuve i au cours d'un cycle et finit au cycle suivant (Figure 2.4)

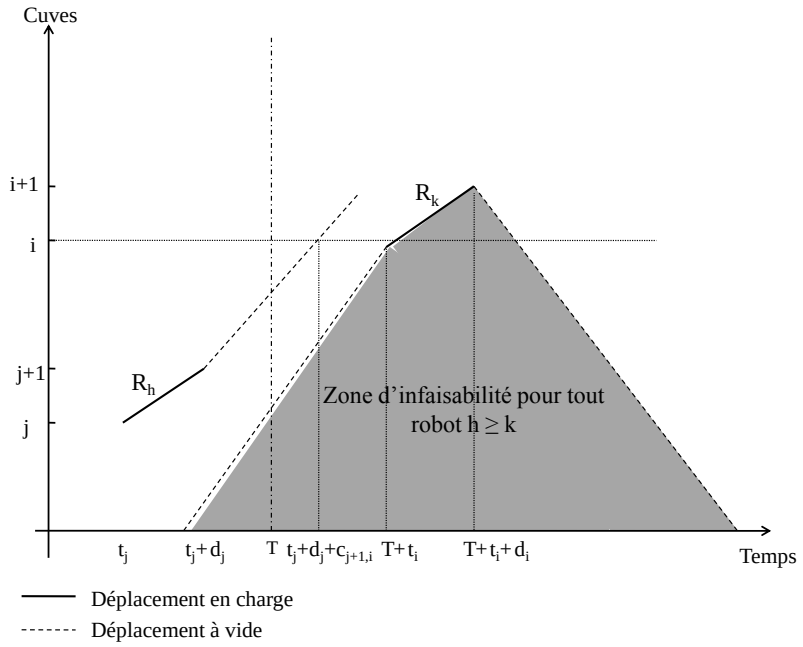

FIGURE 2.4 – Un diagramme de Gantt schématisant la contrainte 2.22

2.3 Une approche hybride pour la résolution du CHSP avec deux robots

Comme il a été mentionné précédemment, un problème de traitement de surface se caractérise par une ligne composée de plusieurs cuves dans lesquelles sont placés des porteurs. Le déplacement se fait à l'aide d'un ou plusieurs robot(s). Dans le cas de plusieurs robots, une collision entre eux peut se produire lorsqu'ils se partagent une même zone. Ciblant un ordonnancement optimal des mouvements de transport de ces robots, plusieurs travaux ont été réalisés dans la littérature pour la résolution des problèmes multi-robots avec zones de chevauchements. Jusqu'à présent, à mes connaissance, tous ces travaux se basent sur des méthodes exactes en développant des programmes linéaires (Leung et Zhang, 2003; Leung *et al.*, 2004), ou bien sûr des heuristiques de résolution. Ces méthodes et heuristiques ont résolu le problème du CHSP multi-robots dans les cas où :

- les temps de traitement sont fixes (Che et Chu, 2008; Liu et Jiang, 2005) ;
- le risque de collision entre les robots n'est pas géré lors de la résolution (Manier et Lamrous, 2008) ;
- on part d'une distribution complète des cuves pour l'affectation des robots (Che et Chu, 2004; Manier, 1994) ;
- on part d'une partition déjà connue (Manier *et al.*, 2000; Zhou et Liu, 2008) ;

Le Tableau 2.1 permet de résumer ces travaux et de s'y positionner. Ainsi, face aux limites des méthodes proposées pour la résolution des problèmes de type CHSP multi-robots avec zones de chevauchement, nous proposons une méthode de résolution de ce problème. Elle permet de gérer les risques de collision, même pour les cas de chevauchement des zones de déplacement des robots. De plus, elle ne fixe pas a priori une affectation des tâches aux robots. Notre approche consiste en une hybridation de deux heuristiques avec un PLM. Nous la détaillons dans les paragraphes suivants.

TABLE 2.1 – Positionnement de notre travail par rapport aux travaux antérieurs

	sans zone de chevauchement	avec zone de chevauchement		
		temps de traitement fixe	temps de traitement bornée	
			partition connue	partition non connue
méthode exacte	Zhou et Li (2009)		Manier *et al.* (2000) Che et Chu (2004)	Leung et Zhang (2003) Leung *et al.* (2004)
heuristique	Lei et Wang (1991)	Che et Chu (2008)	Zhou et Liu (2008)	**algorithmes proposés**
métaheuristique	Yang *et al.* (2001)			Manier et Lamrous (2008)

2.3.1 Hypothèses et notations

Nous considérons une ligne comportant des cuves mono-bac et mono-fonction. Les cuves sont implantées linéairement en forme de **I**. Nous considérons les opérations de chargement et déchargement comme étant, respectivement, le premier traitement et le dernier traitement de la gamme opératoire. De même, les cuves

de chargement et déchargement sont supposées dissociées. Nous supposons également les hypothèses suivantes :

- L'entrée d'un porteur sur la ligne correspond au début de son chargement.
- La sortie d'un porteur par le poste de déchargement correspond à la fin de son déchargement.
- La gamme opératoire des produits à traiter suit l'indexation des cuves de traitement, c'est-à-dire, la $i^{ème}$ opération de la gamme est effectuée dans la cuve i, $i = 1, 2, 3, ..., n$.
- Le premier mouvement exécuté par un robot est le déplacement d'un porteur depuis la cuve de chargement vers la cuve n°1. Cette opération est définie comme le début d'un cycle.
- Au cours d'un cycle, chaque robot est, soit en déplacement en charge ou à vide soit en attente.

Dans la suite, nous présentons de nouvelles notations qui viennent compléter celles déjà définies dans la section 1.5.1 :

n_1	le nombre des mouvements affectés au premier robot R_1 dans un cycle ;
n_2	le nombre des mouvements affectés au deuxième robot R_2 dans un cycle, avec $n_2 = n + 1 - n_1$;
m_i	les dates de début des déplacements du même porteur de chaque cuve, $i_1 = 0, ..., n_1 - 1$;
$move\ i$	opération de déplacement d'un porteur de la cuve i vers la cuve $i + 1$;
p_i	les durées de trempe dans chaque cuve, $i_1 = 0, ..., n_1 - 1$;
$ITER$	le nombre d'itérations définies pour chaque instance ;
$S + 1$	le nombre de séquences générées pour chaque itération ;
$O(s)$	la séquence numéro s des n mouvements au cours d'une itération, $s = 0, ..., S$;
O_1	la séquence des n_1 mouvements pour le robot 1 ; $O_1[i_1] = i$ indique que $move\ i$ est le $i_1^{ème}$ mouvement pour R_1, $i_1 = 0, ..., n_1 - 1$ et $i \in 0, ..., n$;

$O_1[0] = 0$ (le premier mouvement de R_1 est *move* 0) ;

O_2 la séquence des n_2 mouvements pour le robot 2 ;
$O_2[i_2] = j$ indique que *move* j est le $i_2^{ème}$ mouvement pour R_2, $i_2 = 0, ..., n_2 - 1$ et $j \in \{0, ..., n\}$;

EV_r vecteur de temps de dimension n définissant les dates de fin des mouvements à vide du robot r, $r = 1, 2$;

ET_r vecteur de temps de dimension $n + 1$ définissant les date de fin des déplacements en charge du robot r, $r = 1, 2$;

LM_1 vecteur de temps de dimension n_1 définissant les dates de fin des déplacements en charge du robot 1 ;

LM_2 vecteur de temps de dimension $n_2 + 1$ définissant les date de fin des déplacements en charge du robot 2 ;

Les dates, éléments du vecteur EV_r, sont calculées afin de chercher lequel des deux robots arrive le premier à la cuve origine du transport qu'on souhaite affecter. Pour cela, le calcul s'effectue au début de l'algorithme tout en supposant que le mouvement à affecter est assigné aux deux robots.
Les dates, éléments du vecteur ET_r, sont calculées au fur et à mesure qu'un mouvement est affecté à un robot.

2.3.2 Présentation de l'algorithme proposé

Quatre étapes composent l'algorithme proposé (voir Figure 2.5) :

$1^{ère}$ étape : générer un ensemble de séquences de mouvements dans le cas mono-robot en se basant sur la méthode proposée par Zhou et Liu (2008). Cette méthode sera décrite dans la **section 2.3.2.1**.

$2^{ème}$ étape : pour chaque séquence de mouvements générée dans la $1^{ère}$ étape, nous proposons une heuristique qui permet d'affecter les différents mouve-

ments aux deux robots R_1 et R_2. En effet, l'affectation de chaque mouvement sera faite en tenant compte de la position de chaque robot sur le rail et la minimisation de la durée du cycle, tout en gérant le risque de collision entre les robots sachant que R_1 doit toujours se trouver à gauche de R_2. Deux sous-séquences seront définies à la fin de l'heuristique, respectivement, pour R_1 et R_2. Cette heuristique sera présentée dans la **section 2.3.2.2**.

$3^{ème}$ étape : dans cette étape, nous développons un PLM pour vérifier la faisabilité des deux sous-séquences de mouvements. Ce programme permet d'ajuster les valeurs des variables de décision.

A la différence des travaux de Zhou et Liu (2008), le programme linéaire proposé est relaxé par l'élimination des contraintes de non-collision entre les deux robots. Ainsi, le risque de collision entre les deux robots est gérée mais non éliminé dans l'heuristique d'affectation. Le programme linéaire sera présenté dans la **section 2.3.2.3**.

$4^{ème}$ étape : en se basant sur les deux séquences de mouvements définies dans la deuxième étape ainsi que les valeurs de t_i fournis par le PLM, nous proposons une heuristique qui permet de tester s'il existe un risque de collision. Si une collision existe alors $collision = oui$ sinon $collision = non$, avec $collision$ une variable booléen indiquant si une collision existe pour les deux séquences O_1 et O_2. Cette heuristique sera présentée dans la **section 2.3.2.4**.

Comme nous venons de le dire, à la différence des travaux de Zhou et Liu (2008), le programme linéaire proposé est relaxé par l'élimination des contraintes de non-collision entre les deux robots. Nous avons minimisé le risque de collision au niveau de l'heuristique d'affectation. A la fin de l'algorithme, une heuristique permet de tester s'il reste des collisions entre les mouvements des deux robots. Le but est de proposer un algorithme rapide au niveau du temps d'exécution.

Au début de cet algorithme, nous initialisons *iter* à 0.

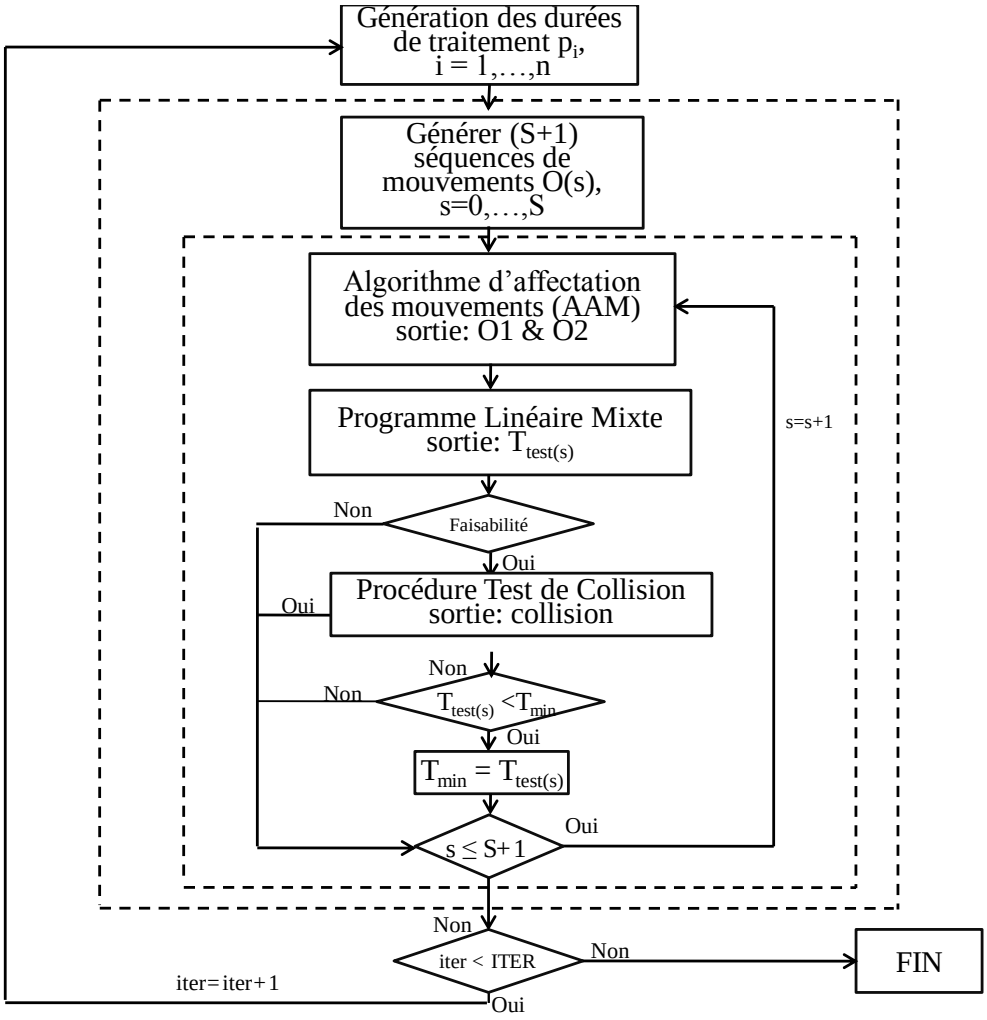

FIGURE 2.5 – Algorithme proposé pour la résolution du CHSP avec deux robots

2.3.2.1 Génération des séquences de mouvements

En se basant sur l'algorithme développé, en premier, par Kats et Levner (1997) et proposé, ensuite, par Zhou et Liu (2008), une heuristique a été définie pour générer un ensemble de séquences des mouvements de transfert des porteurs. Elle consiste à :

– générer aléatoirement un ensemble de durées de traitement p_i à l'aide de l'expression suivante :

$$p_i = L_i + \alpha_i(U_i - L_i), \quad i = 1, ..., n;$$

avec α_i : un nombre aléatoire suivant une distribution uniforme dans [0,1], i=1,...,n ;

– calculer pour chaque ensemble de p_i les dates de début de déplacement m_i d'un porteur entre les cuves de traitement :

$$m_i = m_{i-1} + d_{i-1} + p_i, \quad i = 1, 2, ..., n, \text{avec } m_0 = 0;$$

Chaque déplacement depuis chaque cuve est effectué exactement une fois dans un cycle, vu que chaque porteur est introduit au début de chaque cycle. Ainsi, au cours d'un même cycle de durée T, les dates de début des mouvements de transfert sont définies comme suit :

$$t_i = m_i \mod T, \quad i = 0, ..., n; \tag{2.26}$$

La relation entre m_i et t_i est illustrée par la Figure 2.6 pour un exemple de problème cyclique mono-robot à quatre cuves.

Kats et Levner (1997) ont développé le théorème suivant :

Théorème : lorsque la valeur de T change dans l'intervalle de temps $[T_l, T_u]$, l'ordre des mouvements du robot peut changer aux points $T = T_{ij}^k$.

Avec :

T_l : une borne inférieure pour T ;

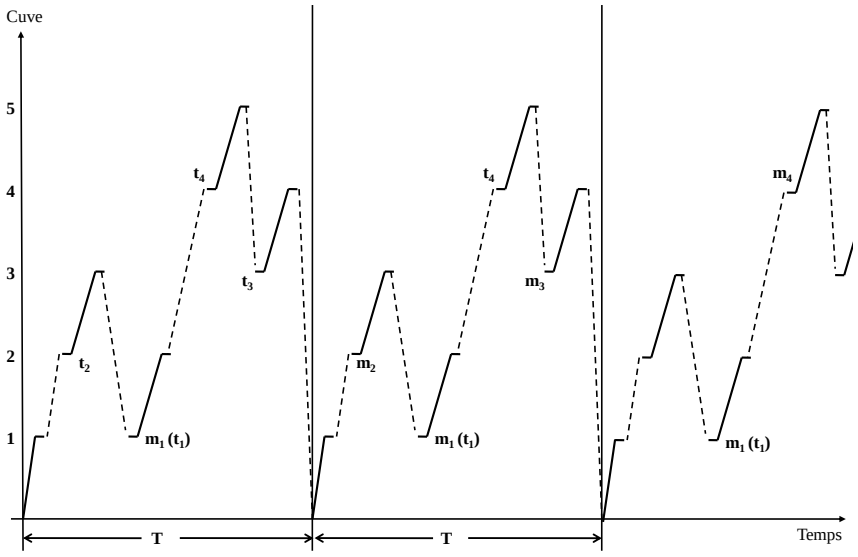

FIGURE 2.6 – Diagramme de Gantt pour un exemple de problème cyclique mono-robot

T_u : une borne supérieure pour T ;

$$T_{ij}^k = (m_j - m_i)/k \quad \text{pour} \quad i = 0, 1, ..., n-1, \quad j = i+1, ..., n \quad \text{et} \quad k = 1, ..., n. \tag{2.27}$$

Zhou et Liu (2008) ont défini une borne inférieure ainsi qu'une borne supérieure de T.

Borne inférieure : la durée d'un cycle ne peut pas être inférieure à la plus grande durée de traitement dans une cuve plus la durée de sécurité δ. Rappelons que cette dernière est mesurée par le rapport entre une distance de sécurité entre les deux robots et la vitesse des robots sachant qu'elle est la même pour les deux robots. Ainsi, une borne inférieure de T est définie comme suit :

$$T_l = \max\{p_i + \delta, \quad i = 0, 1, ..., n\} \tag{2.28}$$

Cette borne est fonction des temps de traitement p_i qui sont générés aléatoirement. Une autre borne pour la durée du cycle T peut être obtenue en remplaçant p_i par les durées minimales L_i. La nouvelle borne minimale sera définie comme

suit :

$$T_l^* = \max\{L_i + \delta, \quad i = 0, 1, ..., n\} \tag{2.29}$$

Borne supérieure : Si un porteur entre dans la ligne après que le précédent termine son traitement et sort du système, alors une borne supérieure de T est obtenue comme suit :

$$T_u = d_0 + \sum_{i=1}^{n} (p_i + d_i) \tag{2.30}$$

– Les différentes valeurs de T, définies par la relation 2.27, seront triées dans un ordre croissant et limitées par les deux bornes inférieure et supérieure telles que $T_l < T_1 < T_2 < ... < T_N < T_u$.

2.3.2.2 Affectation des mouvements aux robots

Dans cette section, nous proposons une heuristique d'affectation des mouvements aux différents robots. Partant d'un vecteur O représentant une séquence de $n + 1$ mouvements, cette heuristique permet d'affecter chaque mouvement au premier robot disponible tout en considérant le risque de collision entre les deux robots.

Le robot situé à l'extrémité gauche de la ligne sera noté R_1, le deuxième sera noté R_2. Comme conditions initiales, nous avons supposé que le premier mouvement noté *move* 0, correspondant au déplacement du robot de la cuve 0 à la cuve 1, est affecté au robot R_1. Le deuxième robot R_2 est censé être au-dessus de la cuve n.

Tout mouvement est affecté à l'un des deux robots en se référant à leurs positions sur la ligne à un instant donné. Ces positions sont représentées sur un diagramme de type Gantt, qui permet de schématiser chaque mouvement pour chaque robot.

Soit une séquence de mouvement $O(s)$ à une itération donnée. Nous supposons que deux mouvements appelés *move* i et *move* j sont déjà affectés, respective-

ment, à R_1 et R_2. Les deux mouvements *move* i et *move* j sont les derniers mouvements effectués par les deux robots à un instant t donné. Le prochain mouvement à affecter est *move* k. Trois cas sont alors envisagés (voir Figures 2.7 et 2.8) :

1^{er} cas. Si $k \geq j$, alors *move* k est affecté à R_2 (voir Figure 2.7) ;

$2^{ème}$ cas. Si $k \leq i$, alors *move* k est affecté à R_1 (voir Figure 2.7) ;

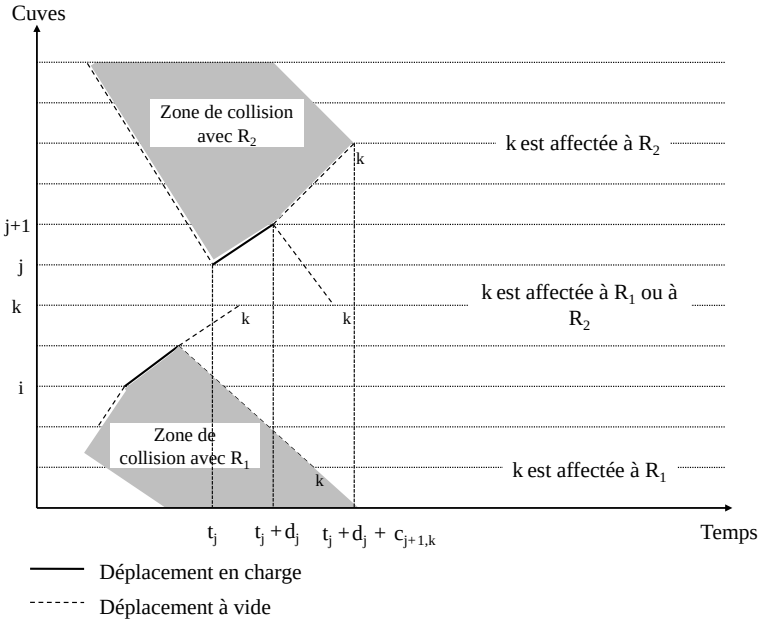

FIGURE 2.7 – Affectation de *move* k pour le premier et le deuxième cas

$3^{ème}$ cas. Si $i < k < j$, alors *move* k peut être affecté soit à R_1 soit à R_2. Dans ce cas, on calcule :

- la date à laquelle le robot R_1 arriverait vide à la cuve k, notée D_1, telle que $D_1 \in EV_1$.
- la date à laquelle le robot R_2 arriverait vide à la cuve k, notée D_2, telle que $D_2 \in EV_2$.

Le détail sur le calcul de ces dates sera déterminé par l'équation (2.31) définie ci-dessous.

Si $D_1 \leq D_2$, R_1 peut arriver à la cuve k avant R_2 (voir Figure 2.8). On cherche alors s'il existe un transport *move* c effectué par R_2 et antérieur à *move* j qui cause un conflit avec *move* k si ce dernier est effectué par le robot R_1, comme le montre la Figure 2.9. S'il existe un conflit, alors *move* k est affecté à R_2, sinon, il est affecté à R_1.

Autrement, si $D_1 \geq D_2$, alors on vérifie s'il existe un mouvement antérieur à *move* i qui cause un conflit avec *move* k sachant que ce dernier serait effectué par R_2. S'il existe un conflit, alors *move* k est affecté à R_1. Sinon, il est affecté à R_2.

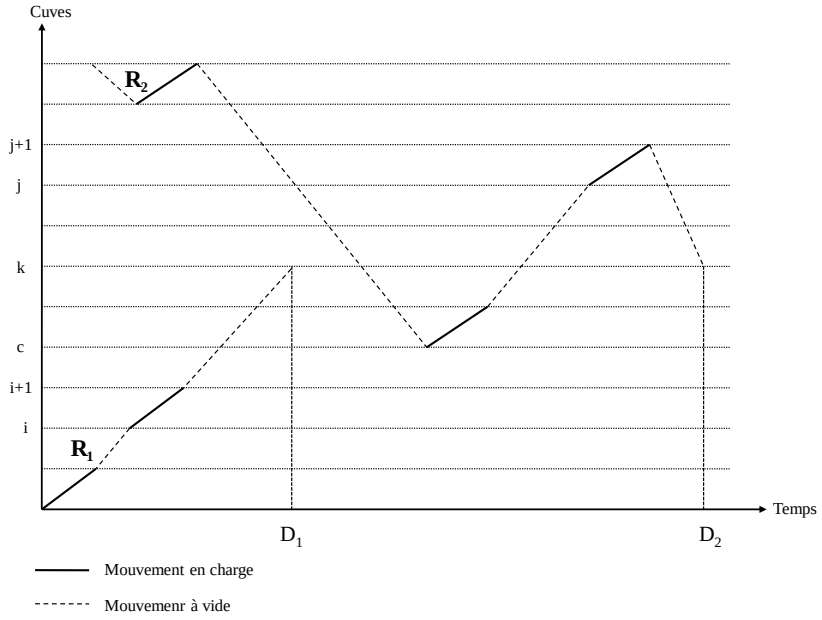

FIGURE 2.8 – Affectation de *move* k pour le troisième cas

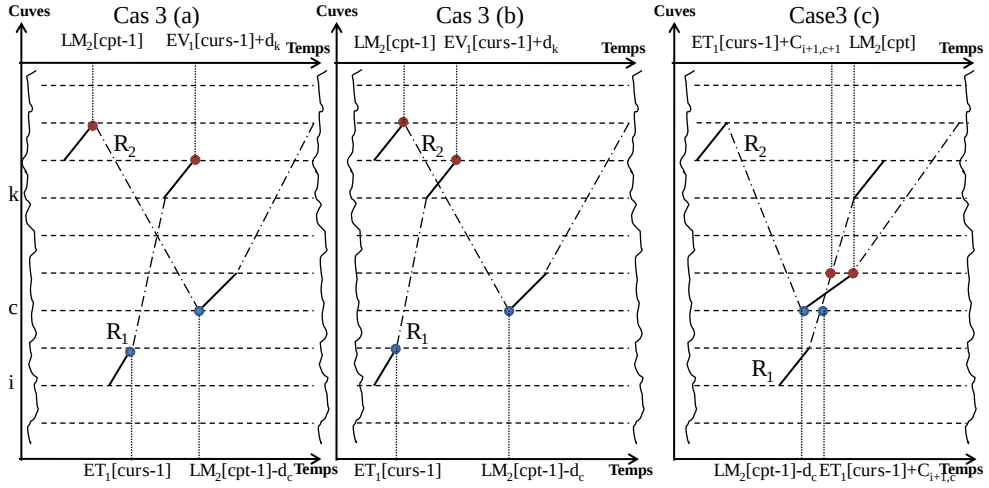

FIGURE 2.9 – Les Types de conflits

Dans ce qui suit, nous allons remplacer $O(s)$ par O pour alléger l'écriture. Le pseudo code de l'algorithme proposé pour l'affectation des différents mouvements aux deux robots est décrit ci-après (Algorithme 1), avec la phase d'initialisation suivante :

Initialisation

- pour le robot R_1

  $ET_1[0] = d_0$: puisque le premier mouvement *move* 0 est affecté automatiquement à R_1 ; $LM_1[0] = d_0$;
- pour le robot R_2

  $ET_2[0] = 0$: aucun mouvement n'est affecté à R_2 et il est supposé au dessus de la cuve n ; $LM_2[0] = 0$.

Supposons qu'à une étape $curs$ ($curs = 1, ..., n$) de l'algorithme, $l_1 - 1$ et $l_2 - 1$ mouvements sont déjà affectés, respectivement, à R_1 et R_2 et le mouvement à affecter est *move* $O[l]$. Le calcul des dates $ET_r[curs]$ et $EV_r[curs - 1]$ pour le robot R_r ($r = 1, 2$) est effectué comme suit :

$$EV_1[curs - 1] = ET_1[curs - 1] + c_{O_1[l_1-1]+1,O[curs]} \tag{2.31}$$

$$EV_2[curs - 1] = ET_2[curs - 1] + c_{O'_2[l_2-1]+1,O[curs]} \tag{2.32}$$

$$ET_r[curs] = EV_r[curs - 1] + d_{O[curs]} \quad r = 1, 2 \tag{2.33}$$

Au cours de cet algorithme, nous définissons deux compteurs $curs_2$ et $curs$ qui parcourent, respectivement, les vecteurs O'_2 et O. $curs$ est connu à priori puisque la taille du vecteur O est connue et est égale à $n + 1$, par contre $curs_2$ sera défini par l'Algorithme 1. Ainsi, la taille du vecteur O'_2 est égale à $curs_2$ c'est-à-dire $n_2 + 1$.

Le premier élément du vecteur O'_2 est $n-1$ ($O'_2[0] = n-1$) vue que nous affectons un mouvement fictif à R_2 pour le localiser à la cuve n ($O'_2[curs_2-1]+1 = n$) au début de l'algorithme. Donc, ce mouvement n'est pas considéré dans l'ordonnancement du robot 2 c'est-à-dire dans O_2. Ainsi, le deuxième élément du vecteur O'_2 correspond au premier élément du vecteur O_2 c'est-à-dire le premier mouvement que le robot R_2 doit effectuer.

Algorithme 1 : Algorithme d'Affectation des Mouvements

Entrée : n, O : une séquence donnée.
Sortie : n_1, n_2, O_1,O_2.
Initialisation : $curs_2 = 1$, $O_1[0] = 0$, $O'_2[0] = n-1$, $n_1 = 1$, $n_2 = 0$, $ET_1[0] = d_0$, $ET_2[0] = 0$, $LM_1[0] = d_0$, $LM_2[0] = 0$.
pour ($curs \leftarrow 1$ à n) **faire**
  $k \leftarrow O[curs]$;
  $EV_1[curs-1] = ET_1[curs-1] + C_{O_1[n_1-1]+1,k}$; */la date à laquelle R_1 arrive vide à la cuve k/*
  $EV_2[curs-1] = ET_2[curs-1] + C_{O'_2[curs_2-1]+1,k}$; */la date à laquelle R_2 arrive vide de la cuve n à la cuve k/*
  si ($k \geq O'_2[curs_2-1]$) **alors**
    /Cas 1/
    $n_2 = n_2 + 1$;
    $O'_2[curs_2] = k$; */on affecte $move$ k à R_2/*
    $ET_2[curs] = EV_2[curs-1] + d_k$; */la date à laquelle R_2 arrive à la cuve $k+1$ après avoir effectué $move$ k/*
    $LM_2[curs_2] = ET_2[curs]$
    $ET_1[curs] = ET_1[curs-1]$;
    $curs_2 = curs_2 + 1$;
  sinon
    si ($k > O_1[n_1-1]$ et $k < O'_2[curs_2-1]$) **alors**
      /Cas 3/
      si ($EV_1[curs-1] \leq EV_2[curs-1]$) **alors**
        /si R_1 arrive vide à la cuve k avant R_2/
        $indice = 0$;
        si ($curs_2 \geq 2$) **alors**
          pour ($cpt \leftarrow (curs_2-1)$ à 1) **faire**
            si ($O'_2[cpt] < k$) **alors**
              si ($LM_2[cpt-1] < EV_1[curs-1] + d_k$ et $ET_1[curs-1] < LM_2[cpt] - d_{O'_2[cpt]}$) ou ($ET_1[curs-1] + C_{O_1[n_1]+1,O'_2[cpt]} > LM_2[cpt] - d_{O'_2[cpt]}$ et $ET_1[curs-1] + C_{O_1[n_1]+1,O'_2[cpt]+1} < LM_2[cpt]$) **alors**
                $indice = indice + 1$; */chercher s'il existe un risque de collisions entre les 2 robots/*
        si ($indice > 0$) **alors**
          $n_2 = n_2 + 1$;
          $O'_2[curs_2] = k$; */on affecte $move$ k à R_2/*
          $ET_2[curs] = EV_2[curs-1] + d_k$;
          $LM_2[curs_2] = ET_2[curs]$;
          $ET_1[curs] = ET_1[curs-1]$;
          $curs_2 = curs_2 + 1$;
        sinon
          $O_1[n_1] = k$; */on affecte $move$ k à R_1/*
          $ET_1[curs] = EV_1[curs-1] + d_k$;
          $LM_1[n_1] = ET_1[curs]$;
          $ET_2[curs] = ET_2[curs-1]$;
          $n_1 = n_1 + 1$;
      sinon
        /si R_2 arrive vide à la cuve k avant R_1/
        $indice = 0$;
        si ($n_1 \geq 2$) **alors**
          pour ($cpt \leftarrow (n_1-1)$ à 1) **faire**
            si ($O_1[cpt] > k$) **alors**
              si ($ET_2[curs-1] < LM_1[cpt]$ et $LM_1[cpt-1] < EV_2[curs-1]$) ou ($LM_1[cpt] - d_{O_1[cpt]} < ET_2[curs-1] + C_{O'_2[curs_2-1]+1,O_1[cpt]}$ et $ET_2[curs-1] + C_{O'_2[curs_2-1]+1,O_1[cpt]+1} < LM_1[cpt]$) **alors**
                $indice = indice + 1$; */chercher s'il existe un risque de collisions entre les 2 robots/*
        si ($indice > 0$) **alors**
          $O_1[n_1] = k$; */on affecte $move$ k à R_1/*
          $ET_1[curs] = EV_1[curs-1] + d_k$;
          $LM_1[n_1] = ET_1[curs]$;
          $ET_2[curs] = ET_2[curs-1]$;
          $n_1 = n_1 + 1$;
        sinon
          $n_2 = n_2 + 1$;
          $O'_2[curs_2] = k$; */on affecte $move$ k à R_2/*
          $ET_2[curs] = EV_2[curs-1] + d_k$;
          $LM_2[curs_2] = ET_2[curs]$;
          $ET_1[curs] = ET_1[curs-1]$;
          $curs_2 = curs_2 + 1$;
      sinon
        /Cas 2/
        $O_1[n_1] = k$; */on affecte $move$ k à R_1/*
        $ET_1[curs] = EV_1[curs-1] + d_k$;
        $LM_1[n_1] = ET_1[curs]$;
        $ET_2[curs] = ET_2[curs-1]$;
        $n_1 = n_1 + 1$;
pour $j \leftarrow 0$ à $n_2 - 1$ **faire**
  $O_2[j] = O'_2[j+1]$

Exemple Illustrative :

Soit l'exemple d'une ligne ("ligne 1") comportant 12 cuves de traitements. Les durées de déplacements à vide et en charge sont définies dans l'Annexe B.

Soit une séquence de mouvements O déterminée par l'heuristique définie précédemment tel que $O = (0, 5, 9, 1, 6, 10, 2, 7, 11, 8, 3, 12, 4)$. En suivant les étapes de l'algorithme d'affectation (Algorithme 1), nous aurons les différentes valeurs définies par les Tableaux 2.2 et 2.3.

TABLE 2.2 – Les étapes de l'Algorithm 1 pour l'exemple illustrative

curs	k	case	$n_r(r = 1, 2)$	O_1 or O'_2	$curs_2$
1	5	3	$n_2 = 1$	$O'_2[1] = 5$	2
2	9	1	$n_2 = 2$	$O'_2[2] = 9$	3
3	1	3	$n_1 = 2$	$O_1[1] = 1$	–
4	6	3	$n_1 = 3$	$O_1[2] = 6$	–
5	10	1	$n_2 = 3$	$O'_2[3] = 10$	4
6	2	2	$n_1 = 4$	$O_1[3] = 2$	–
7	7	3	$n_2 = 4$	$O'_2[4] = 7$	5
8	11	1	$n_2 = 5$	$O'_2[5] = 11$	6
9	8	3	$n_1 = 5$	$O_1[4] = 8$	–
10	3	2	$n_1 = 6$	$O_1[5] = 3$	–
11	12	1	$n_2 = 6$	$O'_2[6] = 12$	7
12	4	3	$n_1 = 7$	$O_1[6] = 4$	–

TABLE 2.3 – Les valeurs de ET_r, EV_r, LM_r $(r = 1, 2)$

	0	1	2	3	4	5	6	7	8	9	10	11	12
ET_1	23	23	23	52	82	82	108	108	108	135	167	167	186
ET_2	0	27	56	56	56	80	80	110	135	135	135	165	165
EV_1	30	33	23	58	87	89	116	118	115	143	176	167	-
EV_2	8	32	66	62	56	90	86	116	142	145	135	175	-
LM_1	23	52	82	108	135	167	186	-	-	-	-	-	-
LM_2	0	27	56	80	110	135	165	-	-	-	-	-	-

A partir du Tableau 2.2, nous obtenons les deux séquences O_1 et O'_2 pour R_1 et R_2, respectivement et par conséquent la séquence O_2 à partir de O'_2, Ainsi que les valeurs de n_1 et n_2. Sept mouvements $(n_1 = 7)$ sont affectés à R_1 et six mouvements $(n_2 = 6)$ à R_2. Les séquences de mouvements sont définies comme

suit :

$$O_1 = (0, 1, 6, 2, 8, 3, 4)$$

$$O_2' = (11, 5, 9, 10, 7, 11, 12)$$

$$O_2 = (5, 9, 10, 7, 11, 12)$$

2.3.2.3 Modèle de Programmation Linéaire Mixte

A ce stade, nous avons obtenu une affectation des opérations de transport, et une séquence associée pour chacun des deux robots. Nous utilisons alors un programme linéaire afin de vérifier la faisabilité de chacune des deux séquences proposées. Il permet aussi d'ajuster les valeurs des variables de décisions. Ce programme linéaire est formulé en tenant compte des deux vecteurs O_1 et O_2. Toutes les contraintes décrites dans la **section 1.5** sont prises en considération dans la formulation. Par contre, les contraintes de non-collision ne sont pas considérées dans ce programme, puisque la gestion du risque de collision entre les deux robots est effectuée par l'algorithme d'affectation et par un test de collision que nous définirons plus tard dans la **section 2.3.2.4**.

Dans notre programme linéaire, les variables de décision sont :

T La durée d'un cycle ;

t_i la date de retrait d'un porteur de la cuve i au cours d'un cycle, $(i = 0, ..., n)$;

s_i Une variable binaire relative à l'état initial de la ligne, c'est à dire à l'occupation des cuves en début de cycle ; $(i = 1, ..., n)$

$$s_i = \begin{cases} 1 & \text{si un porteur est dans la cuve } i \text{ au début du cycle} \\ 0 & \text{sinon} \end{cases}$$

Il est important de noter que les s_i ne sont pas toujours des inconnus. En effet, une fois une séquence des mouvements pour chaque robot est fixée par

l'algorithme d'affectation (Algorithme 1) et avant de résoudre le modèle que nous présenterons ci-dessous, nous pouvons déduire la valeur de s_{i+1} dans le cas où *move* i et *move* $i+1$ **sont réalisés par le même robot** R_r $(r = 1, 2)$. Ainsi, seule une partie des s_i resterait des variables, les autres seraient déduits des données d'entrée du modèle comme suit :

- si *move* i s'effectue avant *move* $i+1$ alors la variable s_{i+1} est égale à 0 $(s_{i+1} = 0)$;
- si *move* i s'effectue après *move* $i+1$ alors la variable s_{i+1} est égale à 1 $(s_{i+1} = 1)$;

Dans le cas où *move* i et *move* $i+1$ **ne sont pas réalisés par le même robot** R_r $(r = 1, 2)$, les valeurs des s_i seront alors défini par le programme linéaire.

Le modèle que nous avons proposé et que nous décrivons ci-après comporte $9 * n + 4$ contraintes et $2 * n + 2$ variables.

$$\text{Minimiser} \quad T \tag{2.34}$$

$$t_0 = 0 \tag{2.35}$$

$$t_{i+1} - (t_i + d_i) \leq U_{i+1}, \quad i = 0, ..., n-1 \tag{2.36}$$

$$T + t_{i+1} - (t_i + d_i) \leq U_{i+1} + M.(1 - s_{i+1}), \quad i = 0, ..., n-1 \tag{2.37}$$

$$t_{i+1} - (t_i + d_i) \geq L_{i+1} - M.s_{i+1}, \quad i = 0, ..., n-1 \tag{2.38}$$

$$T + t_{i+1} - (t_i + d_i) \geq L_{i+1}, \quad i = 0, ..., n-1 \tag{2.39}$$

$$t_{i+1} - (t_i + d_i) + \delta - (U_{i+1} + \delta)(1 - s_{i+1}) \leq 0, \quad i = 0, ..., n-1 \tag{2.40}$$

$$t_{O_1[i_1+1]} - t_{O_1[i_1]} \geq d_{O_1[i_1]} + c_{O_1[i_1]+1, O_1[i_1+1]}, \quad i_1 = 0, ..., n_1 - 2 \tag{2.41}$$

$$t_{O_2[i_2+1]} - t_{O_2[i_2]} \geq d_{O_2[i_2]} + c_{O_2[i_2]+1, O_2[i_2+1]}, \quad i_2 = 0, ..., n_2 - 2 \tag{2.42}$$

$$T + t_{O_1[0]} - t_{O_1[n_1-1]} \geq d_{O_1[n_1-1]} + c_{O_1[n_1-1]+1, O_1[0]} \tag{2.43}$$

$$T + t_{O_2[0]} - t_{O_2[n_2-1]} \geq d_{O_2[n_2-1]} + c_{O_2[n_2-1]+1, O_2[0]} \tag{2.44}$$

$$t_i \leq T, \quad i = 0, ..., n \tag{2.45}$$

$$t_i \geq 0, \quad i = 0, ..., n \tag{2.46}$$

$$s_i \in \{0, 1\}, i = 1, ..., n \tag{2.47}$$

avec $M = T_u$.

La valeur de M a été définie en se référant aux expérimentations établies par Leung *et al.* (2004) qui ont prouvé que la meilleure valeur de M est la borne supérieure de T.

Nous avons pris la même valeur que celle fixée par Leung *et al.* (2004) pour δ. Cette valeur est égale à 1 seconde. Ce choix était basé sur le fait qu'une partie de nos expérimentations a été comparée a celle de Leung *et al.* (2004).

Interprétation des contraintes :

La fonction objectif pour ce problème (2.34) est la minimisation de la durée du cycle T pour une séquence donnée.

La contrainte (2.35) définit une date de début pour un cycle. Nous avons supposé qu'un cycle commence dès la fin de l'opération de chargement d'un porteur.

Les contraintes (2.36) à (2.39) assurent que le temps de traitement dans chaque cuve i $(i = 1, ..., n)$ doit respecter les bornes inférieure et supérieure.

Les contraintes (2.40) garantissent qu'un temps de sécurité δ entre l'opération de levage et descente d'un porteur au niveau d'une même cuve lorsque ces deux opérations coïncident dans le temps. En effet, cette sécurité est assurée par le temps de traitement qui ne doit pas dépasser la borne supérieure lorsque $s_{i+1} = 0$. Par contre, lorsque $s_{i+1} = 1$, cette sécurité assure la non collision entre deux robots différents au moment où l'un effectue la levée d'un porteur et l'autre effectue la dépose d'un autre dans la cuve $i + 1$.

Les contraintes (2.41) et (2.42) assurent que chaque robot a suffisamment de temps pour effectuer un déplacement à vide entre deux transports successifs qui lui sont assignés.

Les contraintes (2.43) et (2.44) assurent la transition entre 2 cycles successifs : chaque robot doit avoir suffisamment de temps pour reprendre le premier mouvement qui lui est affecté au cycle suivant. Ainsi, le robot R_1 doit avoir le temps nécessaire pour se déplacer de la dernière cuve visitée dans le cycle courant jus-

qu'à la cuve 0. Le robot R_2 doit avoir le temps nécessaire pour se déplacer de la dernière cuve visitée dans le cycle courant jusqu'à la première cuve qui lui est affectée au cycle suivant.

Les contraintes (2.45) assurent que les dates de retrait d'un porteur de chaque cuve t_i $(i = 0, ..., n)$ soient inclus dans un cycle.

Les contraintes (2.46) à (2.47) sont les contraintes de non-négativité du modèle.

2.3.2.4 Procédure Test de Collision

Au niveau de l'algorithme d'affectation, nous avons essayé de chercher une bonne affectation des mouvements aux deux robots tout en minimisant le risque de collision. La disponibilité des robots est prise en compte lors de cette affectation par le calcul des différentes valeurs éléments des vecteurs EV_r et ET_r. Par contre, nous n'avons pas tenu compte des temps de traitement dans chaque cuve. Ensuite, sur la base des deux séquences O_1 et O_2 définies par l'algorithme d'affectation, le PLM fournit les différentes valeurs de t_i tout en respectant les contraintes sur les temps de traitement ainsi que les contraintes sur la disponibilité des robots. Nous avons éliminé les contraintes qui gèrent la non collision afin d'alléger le PLM. De ce fait, chaque solution ne garantit pas l'absence de collision entre les deux robots. Par conséquent, nous proposons, dans cette section, une heuristique qui permet de tester s'il existe ou non un risque de collision entre les deux robots.

Ce test est établi en prenant en considération les deux séquences O_1 et O_2, respectivement, pour R_1 et R_2 ainsi que les valeurs de t_i $(i = 0, ..., n)$ fournies par le PLM.

Nous supposons deux mouvements appelés *move* i_1 et *move* i_2 et effectués, respectivement, par R_1 et R_2 tels que $(i_1 > i_2)$. Aussi, soit le mouvement *move* i'_1 effectué par R_1 juste après *move* i_1 et le mouvement *move* i'_2 effectué par R_2 juste après *move* i_2. Tenant compte des valeurs de t_i $(i = 0, ..., n)$, deux cas sont envisagés :

Cas 1. *move* i_1 est effectué après *move* i_2 et *move* i'_2 après *move* i_1 ($t_{i'_2} > t_{i_1} > t_{i_2}$). Deux autres situations peuvent se présenter selon la position de la cuve i'_2 :

- si ($i'_2 > i_1$) alors pour que les deux robots n'entrent pas en collision, R_2 doit arriver à la cuve i_1 avant R_1. Autrement dit, cette inéquation doit être respectée (voir Figure 2.10) :

$$t_{i_2} + d_{i_2} + c_{i_2+1,i_1} + \delta \leq t_{i_1}$$

- si ($i'_2 < i_1$) alors les risques de collisions augmentent. Pour les éviter, nous risquons de rendre les séquences O_1 et/ou O_2 infaisables du fait du non respect probable des deux conditions suivantes (voir Figure 2.10) :

$$t_{i_1} + d_{i_1} + c_{i_1+1,i'_2} + \delta \leq t_{i'_2}$$

$$t_{i2} + d_{i2} + c_{i2+1,i1+2} + c_{i1+2,i1+1} \geq t_{i_1} + d_{i_1} + \delta$$

La première relation garantit que R_1 doit arriver à i'_2 vide avant que le robot R_2 effectue le *move* i'_2. La deuxième relation garantit que si R_2 doit se pousser entre deux transports pour laisser passer R_1, alors il doit avoir le temps de faire tous les déplacements à vide nécessaire avant $t_{i'2}$, tout en supposant que R_2 s'est déplacé jusqu'à la cuve située juste après la destination de R_1 ($i_1 + 2$).

Ainsi, nous pouvons estimer qu'il y a de grandes chances que R_2 n'ait pas la marge nécessaire pour effectuer tous les mouvements à vide pour éviter la collision. Pour cette raison, nous estimons qu'effectivement il y a collision tout en étant conscientes que quelques solutions faisables risquent d'être éliminées.

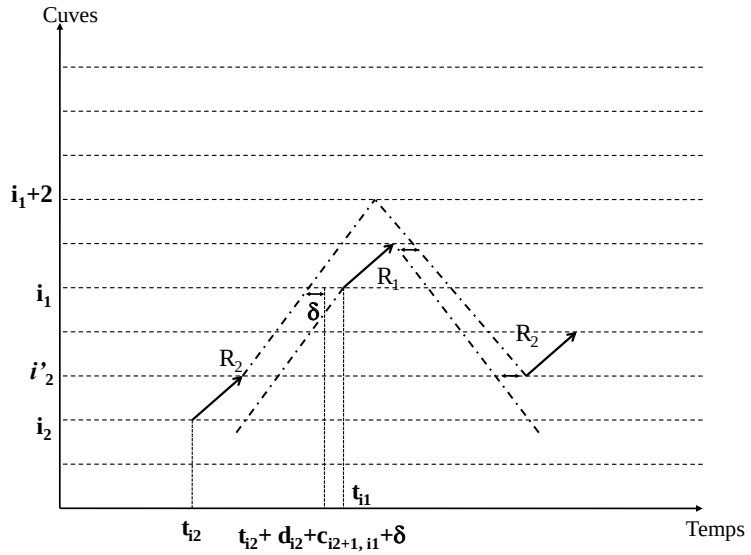

FIGURE 2.10 – Test de collision : cas 1

Cas 2. *move* i_1 est effectué avant *move* i_2 et *move* i'_1 après *move* i_2 ($t_{i_1} < t_{i_2} < t_{i'_1}$). Deux autres situations peuvent se présenter selon la position de la cuve i'_1 :

- si ($i'_1 < i_2$) alors pour que les deux robots n'entrent pas en collision, R_1 doit arriver à la cuve i_2 avant R_2. Autrement dit, cette inéquation doit être respectée (voir Figure 2.11) :

$$t_{i_1} + d_{i_1} + c_{i_1+1,i_2} + \delta \leq t_{i_2}$$

- si ($i'_1 > i_2$) alors les risques de collisions augmentent. Pour les éviter, nous risquons de rendre les séquences O_1 et/ou O_2 infaisables du fait du non respect probable des deux conditions suivantes (voir Figure 2.11) :

$$t_{i_2} + d_{i_2} + c_{i_2+1,i'_1} + \delta \leq t_{i'_1}$$

$$t_{i1} + d_{i1} + c_{i1+1,i2-1} + c_{i2-1,i2+1} \geq t_{i_2} + d_{i_2} + \delta$$

La première relation garantit que R_2 doit arriver à i'_1 vide avant que le

robot R_1 effectue le *move* i'_1. La deuxième relation garantit que si R_1 doit se pousser entre deux transports pour laisser passer R_2, alors il doit avoir le temps de faire tous les déplacements à vide nécessaire avant $t_{i'1}$, tout en supposant que R_1 s'est déplacé jusqu'à la cuve située juste avant la cuve i_2 $(i_2 - 1)$.

Ainsi, nous pouvons estimer qu'il y a de grandes chances que R_1 n'ait pas la marge nécessaire pour effectuer tous les mouvements à vide pour éviter la collision. Pour cette raison, nous estimons qu'effectivement il y a collision tout en étant conscientes que quelques solutions faisables risquent d'être éliminées.

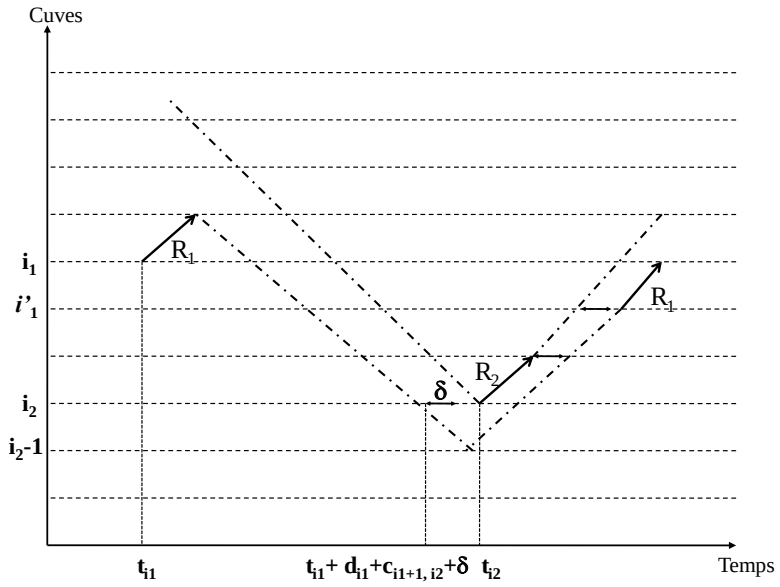

FIGURE 2.11 – Test de collision : cas 2

Une zone de chevauchement est définie selon les deux ordonnancements déterminés, respectivement, pour R_1 et R_2. Cette heuristique permet de tester si une collision existe dans cette zone. Elle est appliquée pour tout $i_1 > i_2$ avec $i_1 \in O_1$ et $i_2 \in O_2$. Au cours de cette heuristique dès qu'un fort risque de collision est détecté ($collision = true$), elle est arrêtée. Elle est définie ci-dessous (Algorithme 2) :

Algorithme 2 : Test de Collision (TC)

Entrée : n, $n_1 - 1$, $n_2 - 1$, O_1, O_2, t.
Sortie : *collision* : variable booléenne indiquant si une collision existe pour les deux séquences O_1 et O_2.

```
pour (curs1 ← 0 à n1 − 1) faire
  pour (curs2 ← 0 à n2 − 1) faire
    si (O1[curs1] > O2[curs2] ) alors
      si (t_{O1[curs1]} < t_{O2[curs2]}) et (t_{O2[curs2]} < t_{O1[curs1+1]}) alors
        si O1[curs1 + 1] < O2[curs2] alors
          si (t_{O2[curs2]} − t_{O1[curs1]} ≤ d_{O1[curs1]} + C_{O1[curs1]+1,O2[curs2]} + δ) alors
            */Cas 2 n'est pas vérifié/*
            collision = true ;   */il existe une collision/*
          sinon
            */Cas 2 est vérifié/*
            collision = false ;
        sinon
          collision = true ;   */il existe une collision/*
      sinon
        si (t_{O2[curs2+1]} < t_{O1[curs1]}) et (t_{O1[curs1]} < t_{O2[curs2]}) alors
          si O2[curs2 + 1] > O1[curs1] alors
            si (t_{O1[curs1]} − t_{O2[curs2]} ≤ d_{O2[curs2]} + C_{O2[curs2]+1,O1[curs1]} + δ) alors
              */Cas 1 n'est pas vérifié/*
              collision = true ;   */il existe une collision/*
            sinon
              */Cas 1 est vérifié/*
              collision = false ;
          sinon
            collision = true ;   */il existe une collision/*
```

Nous avons proposé une méthode qui permet de résoudre les problèmes du CHSP avec deux robots. Partant d'une séquence de mouvements, nous avons proposé un algorithme qui cherche la meilleure affectation pour chaque mouvement ($move\ i$),

- de sorte que le robot affecté est celui qui arrive le premier pour effectuer $move\ i$,
- en vérifiant si ce mouvement cause un conflit avec d'autres mouvements déjà effectués par l'autre robot,

Donc, avec cette heuristique, nous obtenons deux séquences de mouvements qui satisfont les contraintes sur les robots (disponibilité et bonne gestion de la non collision). Mais, nous n'avons pas pris en considération les contraintes sur les temps de traitements. De ce fait, ces deux séquences peuvent être non réalisables.

Pour ceci, nous avons proposé un PLM qui vérifie si cette affectation est réalisable et détermine dans ce cas les valeurs de t_i, et par suite de p_i, en tenant compte de la disponibilité des robots et des cuves (temps de traitement respecté). Toutefois, cette solution ne garantit pas l'élimination complète du risque de collision entre les deux robots. Pour cela, nous avons proposé une heuristique qui teste s'il subsiste encore des mouvements qui entrent en collision pour une solution donnée (définie par les deux séquences et les valeurs de t_i).

2.4 Résultats expérimentaux

Les expérimentations ont été réalisées sur un processeur Intel(R) Core(TM)2 Duo cadencé à 2,6 GHz et possédant un 1 Gbyte de RAM. Le système d'exploitation est Windows 7, le langage orienté objet est le C++, compilé avec Microsoft Visual C++ 6.0. Le solveur utilisé est CPLEX 10.0. Nous avons utilisé la librairie d'ILOG CONCERT version 1.1 qui sert à la modélisation des programmes linéaires et fait office d'interface avec HYBRID version 1.1 d'ILOG pour l'optimisation de programmes linéaires en nombres entiers, mixtes ou continus et qui utilise le solveur CPLEX version 10.0.

Nous avons implémenté, tout d'abord, le modèle linéaire proposé par Leung *et al.* (2004) et qui permet de résoudre les problèmes CHSP multi-robots. Ceci permet d'avoir une solution optimale pour pouvoir tester la performance de notre algorithme.
Dans une première étape, nous avons généré un ensemble d'instances qui ont été proposées par Mateo et Companys (2007) et Zhou et Liu (2008) (**section 2.4.1**). Ensuite, nous avons testé notre algorithme sur ces instances et interprété les résultats (**section 2.4.2**). Dans la dernière étape, nous avons testé cet algorithme sur d'autres exemples pris de la littérature (**section 2.4.3**).

2.4.1 Description des instances

Les expérimentations seront effectuées sur deux types d'instances. Des instances proposées par Mateo et Companys (2007) et des instances proposées par Zhou et Liu (2008).

Les instances proposées par Mateo et Companys (2007) sont générées aléatoirement en fonction de la largeur des fenêtres de temps et de la vitesse des robots. Les fenêtres de temps seront classées en fonction de la différence entre la borne minimale et la borne maximale $(U_i - L_i)$ pour chaque cuve i. Le temps de traitement minimal pour chaque cuve est généré aléatoirement selon une distribution uniforme (Ng, 1996) : $L_i = U[20; 80]$. Ainsi, les fenêtres de temps pour chaque cuves peuvent être classées en trois groupes :

- W_1 (fenêtre de temps étroite) pour $U_i - L_i = U[1, 2L_i; 2L_i]$;
- W_2 (fenêtre de temps moyenne) pour $U_i - L_i = U[2L_i; 3L_i]$;
- W_3 (fenêtre de temps large) pour $U_i - L_i = U[3L_i; 10L_i]$;

La deuxième classification tiendra compte des vitesses des robots en considérant le quotient entre la durée de déplacement du robot en charge entre les deux cuves i et $i+1$ (d_i) et sa durée à vide $(c_{i,i+1})$. Sur le même principe, trois groupes seront définis (Ng, 1996). Ainsi, la durée $(c_{i,i+1})$ sera générée selon la distribution uniforme : $c_{i,i+1} = U[5; 10]$. Trois groupes seront définis pour calculer les différentes valeurs de d_i :

- H_1 (robot rapide) pour $d_i/c_{i,i+1} < 2$;
- H_2 (robot à vitesse moyenne) pour $d_i/c_{i,i+1} = 2$;
- H_3 (robot lent) pour $d_i/c_{i,i+1} > 2$;

Par conséquent, neuf groupes d'instances sont générés en tenant compte des fenêtres de temps et de la vitesse des robots $((W_1H_1)$, (W_1H_2), (W_1H_3), (W_2H_1), (W_2H_2),(W_2H_3), (W_3H_1), (W_3H_2), $(W_3H_3))$. Pour chaque valeur de n $(n = 11, 12, 13, 15, 20, 25)$ et pour chaque groupe, dix instances seront générées. En tout, 540 instances ont été testées. Nous avons utilisé les données de Mateo et Companys (2007) pour générer ces instances, mais nous ne pouvons

pas nous comparer aux solutions de ces chercheurs, car ils n'ont pas traité le même problème que nous. En effet, Mateo et Companys (2007) ont résolu des problèmes CHSP 2-périodiques mono-robot, sur des lignes de 5 à 10 cuves, avec deux types de produits entrant alternativement sur l'atelier et ayant des fenêtres de temps différentes dans les cuves. Cela explique que, sur la base de ces jeux de données, nous nous comparons plutôt aux solutions fournies par la méthode de Leung *et al.* (2004).

Les instances proposées par Zhou et Liu (2008) sont des instances aléatoires générées comme suit : les auteurs ont généré un vaste ensemble de 200 instances ayant deux tailles différentes ($n = 18$ et $n = 20$). Pour chaque taille, deux groupes de fenêtres de temps sont définis : fenêtre de temps étroite et fenêtre de temps large. Quatre groupes d'instances sont alors générées. Pour chaque groupe, ils ont généré 50 instances aléatoires. Pour le groupe d'instances avec fenêtre de temps large, les bornes inférieures (L_i) ont été générées comme suit :

$$L_i = 30 + 100.e^{\alpha_i}$$

avec α_i : un nombre aléatoire suivant une distribution uniforme dans [0,1], i=0,1,...,n ;
Les bornes supérieures sont générées aléatoirement et calculées comme suit :

$$U_i - L_i = U[0, 2L_i; 0, 6L_i]$$

Les temps de déplacement en charge des robots d_i sont fixés à 13 et les temps de déplacement à vide sont générés comme suit :

$$c_{i,j} = 2|i - j|$$

Pour le groupe d'instances avec fenêtre de temps étroite, seules les bornes supérieures sont changées et elles sont calculées comme suit :

$$U_i' = L_i + (U_i - L_i)/2$$

2.4.2 Interprétation des résultats

Dans une première partie, l'efficacité de l'algorithme proposé a été évaluée selon le pourcentage de déviation ($\%dév$) par rapport à la solution optimale. Pour cela, nous avons implémenté le PLM proposé par Leung *et al.* (2004) afin de calculer les solutions optimales pour chaque instance.

Nous avons d'abord effectué des tests sur les instances issues de Mateo et Companys (2007). Le Tableau 2.4 présente les temps d'exécution moyens des neufs groupes d'instances ((W_1H_1), (W_1H_2),...) pour l'algorithme proposé ainsi que les temps d'exécution moyens pour le programme linéaire proposé par Leung *et al.* (2004). Il présente, aussi, les pourcentages de déviation moyens obtenus par rapport à la solution optimale ainsi que les pourcentages d'amélioration moyens obtenus par rapport à la borne supérieure. Ces derniers présentent un intérêt surtout lorsque aucune solution optimale n'est disponible (taille d'instance importante).

Le pourcentage de déviation par rapport à la solution optimale est calculé selon l'équation 2.48 :

$$\%dév = \frac{T_H - T^*}{T^*} * 100 \tag{2.48}$$

Le pourcentage d'amélioration est calculé comme suit :

$$\%amél = \frac{T_u - T_H}{T_u} * 100 \tag{2.49}$$

Avec :

– T_H : la solution obtenue par l'algorithme proposé pour une instance ;

– T^* : la solution optimale pour une instance ;

– T_u : borne supérieure définie précédemment (équation 2.30).

TABLE 2.4 – Présentation des résultats pour les instances générées (Mateo et Companys, 2007)

n	CPU_H^{moy}	CPU_{milp}^{moy}	$\%dév(moy)$	$\%amél(moy)$
11	21,928	355,049	2,58	75,52
12	29,829	7598,584	2,95	73,56
13	14,951	18129,598	9,58	73,72
15	120,506	***	***	70,90
20	115,047	***	***	66,49
25	33,946	***	***	57,86

Avec :

– n : la taille de l'instance (nombre de cuves) ;

– CPU_H^{moy} : le temps d'exécution moyen des instances pour les 9 groupes de taille n défini par notre algorithme ;

– CPU_{milp}^{moy} : le temps d'exécution moyen des instances pour les 9 groupes de taille n défini par le PLM proposé par Leung *et al.* (2004) ;

– $\%dév(moy)$: le pourcentage de déviation moyen de toutes les instances pour les 9 groupes de taille n ;

– $\%amél(moy)$: le pourcentage d'amélioration moyen de toutes les instances pour les 9 groupes de taille n.

D'après les résultats présentés dans le Tableau 2.4, nous remarquons que notre algorithme est performant du point de vue temps d'exécution. En outre, le pourcentage de déviation par rapport à la solution optimale n'excède pas les 9%. Pour chaque groupe d'instances, au moins 50% des solutions optimales sont trouvées par l'algorithme proposé, ce qui prouve sa bonne performance.

A partir de $n = 15$, la résolution exacte requiert un temps de calcul supérieur à un temps limite que nous avons fixé à 30000 secondes, alors qu'il ne dépasse

pas 121 secondes en moyenne par l'algorithme proposé.

Nous avons calculé les pourcentages d'amélioration par rapport à une borne supérieure pour les instances de différentes tailles ($n = 11, 13, 13, 15, 20, 25$). D'après le Tableau 2.4, nous remarquons que ces pourcentages varient entre 73,72% et 57,86%.

La Figure 2.12 montre la variation du pourcentage d'amélioration pour les différentes tailles du problème selon la taille des fenêtres du temps. Nous remarquons que lorsque n augmente, le taux d'amélioration diminue. Jusqu'à $n = 13$, la largeur des fenêtres de temps n'a pas d'influence sur la solution. En fait, les taux d'amélioration sont pratiquement équivalents. Pour des fenêtres de temps étroites, la qualité de la solution se dégrade. Ainsi, le % d'amélioration décroît pour les fenêtres de temps de type W_1 plus que celles de type W_3 (une différence de l'ordre de 18%).

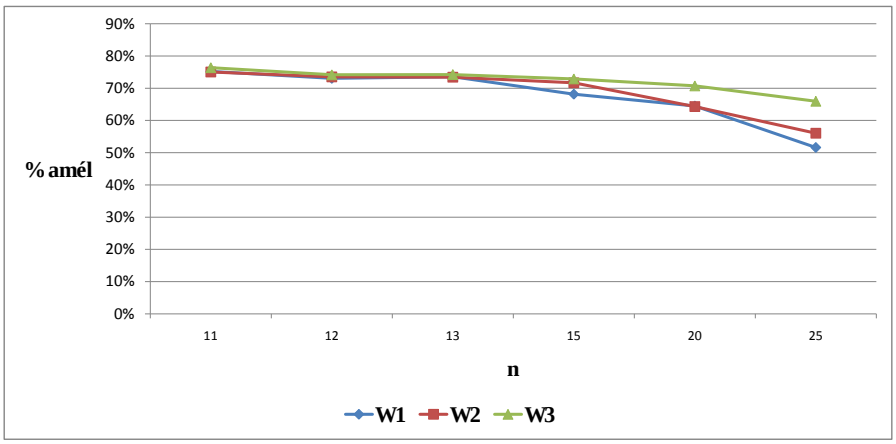

FIGURE 2.12 – Variation du %amél selon la taille du problème et la largeur des fenêtres de temps

La Figure 2.13 montre la variation du pourcentage d'amélioration pour les différentes tailles du problème selon la vitesse des robots. Nous remarquons que lorsque n augmente, la vitesse du robot n'influe pas beaucoup sur la qualité de la solution.

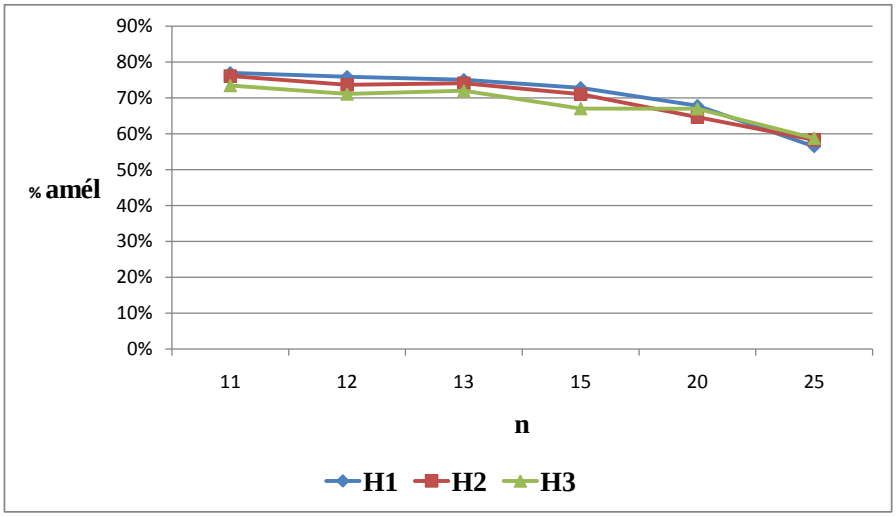

FIGURE 2.13 – Variation du %amél selon la taille du problème et la vitesse des robots

Le Tableau 2.5 présente les temps d'exécution moyens ainsi que les pourcentages de déviations pour les instances générées selon Zhou et Liu (2008). Les pourcentages de déviation par rapport à la solution optimale (fournie par l'algorithme de Leung *et al.* (2004)) sont assez faibles et ne dépassent pas 4%.

Nous avons utilisé l'algorithme de Leung *et al.* (2004) pour calculer les solutions optimales des instances générées selon les données de Mateo et Companys (2007) et de Zhou et Liu (2008). Cet algorithme nous a fourni des solutions en un temps d'exécution (CPU) court pour les instances de Zhou et Liu (2008) c'est-à-dire pour $n = 18$ et 20. Par contre, ce même algorithme n'arrive pas à

fournir des solutions optimales pour les instances de Mateo et Companys (2007) pour $n \geq 15$ malgré que ces dernières sont de taille moins grande que celle de Zhou et Liu (2008). Ceci tient principalement aux ressources du problème. En effet, les bornes inférieures et supérieures des durées de traitement dans les cuves ainsi que les durées de déplacements des robots (en charge ou à vide) influent considérablement sur le temps d'exécution lors de la résolution du problème. Effectivement, les bornes inférieures des durées de traitement pour les instances de Zhou et Liu (2008) sont assez élevées par rapport à celles des instances de Mateo et Companys (2007) ainsi que la vitesse des robots. En effet, les robots dans les instances de Zhou et Liu (2008) sont plus rapides que ceux des instances de Mateo et Companys (2007). Cela permet d'augmenter la disponibilité des robots et par conséquent, l'espace de recherche relatif à ces instances est étendu et facile à explorer.

Nous avons comparé, aussi, nos résultats avec ceux de Zhou et Liu (2008) (voir Tableau 2.6). Au niveau du temps d'exécution, l'algorithme que nous avons proposé est plus rapide à trouver une bonne solution que celui proposé par Zhou et Liu (2008). Aussi, les pourcentage de déviation par rapport à la borne inférieure sont faibles et nettement inférieurs à ceux de Zhou et Liu (2008), ce qui signifie que nous trouvons de meilleures solutions.
L'efficacité de notre algorithme par rapport à celui de Zhou et Liu (2008) peut être due à :

- notre algorithme d'affectation qui donne plus de flexibilité aux deux robots c'est-à-dire que la zone de chevauchement que nous proposons n'est pas limitée et varie d'une solution à une autre. Par contre, Zhou et Liu (2008) ont limité la zone de chevauchement avant même l'affectation des mouvements aux robots. Ainsi, au début de l'algorithme ils fixent trois zones connues à priori : une première pour le robot 1, une deuxième pour les deux robots et une troisième pour le robot 2 ;

– l'élimination des contraintes de non collision entre les deux robots, ce qui explique la rapidité de l'algorithme que nous avons proposé.

La borne inférieure a été calculée selon l'équation 2.50 et présentée comme suit :

$$BI = max\left\{max\left\{L_i + \delta\right\} i = 0, ..., n, \quad \left(\sum_{i=0}^{n} d_i\right)/2\right\}, \tag{2.50}$$

Le premier terme signifie que si deux robots chargent et déchargent, respectivement, la même cuve pendant un cycle , alors un des robots ne peut pas arriver au-dessus de cette cuve avec un porteur avant que l'autre robot :

– ait pris le porteur hors de la cuve de traitement (cette mesure ne peut intervenir qu'après que ce porteur a passé au moins son temps de traitement minimal L_i dans la cuve) ;

– et l'ait déplacé assez loin de cette position, c'est à dire au moins à une distance de sécurité afin éviter une collision. Delta (δ) correspond au temps de déplacement nécessaire pour atteindre cette distance de sécurité.

Le second terme considère une affectation optimisée des mouvements en supposant un équilibrage parfait entre les deux robots pour les déplacements en charge.

TABLE 2.5 – Présentation des résultats pour les instances générées (Zhou et Liu, 2008)

Instances	***n***	CPU_H^{moy}	CPU_{milp}^{moy}	$\%dév(moy)$
Fenêtre de temps large	**18**	5,775	8,725	2,25
	20	13,52	36,047	3,58
Fenêtre de temps étroite	**18**	4,20	29,309	3,04
	20	7,5	154,5	3,87

TABLE 2.6 – Comparaison des résultats avec ceux de (Zhou et Liu, 2008)

Temps d'exécution moyen (CPU) en secondes				
Algorithme de Zhou et Liu (2008)			Algorithme proposé	
n	F.T.L.	F.T.E	F.T.L.	F.T.E
18	381	680	**5,775**	**4,20**
20	671,9	1280,2	**13,52**	**7,5**
Pourcentage de déviation moyen par rapport à la borne inférieure (BI)				
Algorithme de Zhou et Liu (2008)			Algorithme proposé	
n	F.T.L.	F.T.E	F.T.L.	F.T.E
18	10,32	18,93	**4,75**	**5,55**
20	15,52	23,83	**6,09**	**7,14**

Avec :

F.T.L : fenêtre de temps large ;

F.T.E : fenêtre de temps étroite.

Afin de montrer que notre algorithme permet bien d'éliminer toutes les collisions et de fournir de meilleures solutions, nous présentons la Figure 2.14 un diagramme de Gantt pour notre solution d'une des instances de Zhou et Liu (2008). La solution présentée ci-dessous est une solution optimale pour $n = 20$. Le détail de cette solution est présenté en Annexe A.

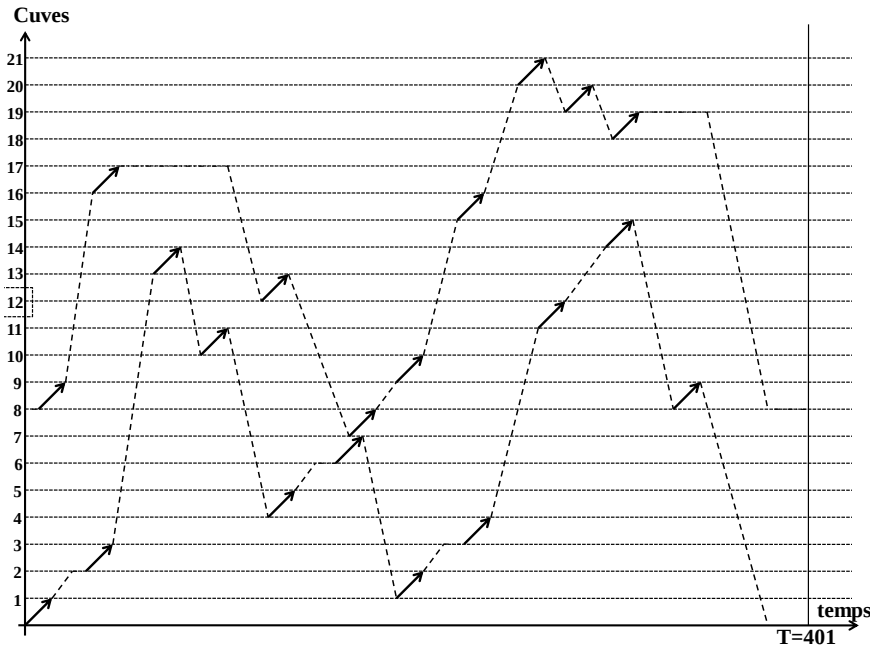

FIGURE 2.14 – Diagramme de Gantt pour l'instance de Zhou et Liu (2008)

D'après la Figure 2.14, les deux robots partagent une zone commençant de la cuve 7 à la cuve 15. Nous pouvons ainsi voir l'utilité du paramètre δ au niveau de la cuve 7 où R_1 retire un porteur exactement δ unités de temps avant que R_2 y place un autre porteur. Ainsi, un risque de collision a été éliminé.

2.4.3 Expérimentation sur des exemples de la littérature

L'algorithme proposé est testé sur des exemples dont les données sont directement trouvées dans la littérature. Le premier exemple est celui proposé par W.Phillips et S.Unger (1976) noté "P&U" et qui est devenu par la suite l'exemple de référence de tous les travaux sur le CHSP. Ensuite, il est testé sur deux exemples industriels proposés dans la thèse de Manier (1994) et libellés , respectivement, "ligne 1" et "ligne 2". Dans le quatrième exemple, nous allons utiliser les données proposées par W.Phillips et S.Unger (1976) mais modifiées par Manier (1994) en considérant un nouvel aménagement des cuves selon la matrice des durées de transport à vide. Les durées de transport des robots ainsi

que les bornes inférieures et supérieures des temps de traitement dans chaque cuve pour les quatre exemples sont présentées dans l'Annexe B.

2.4.3.1 Résultat pour l'exemple "P&U"

Le Tableau 2.7 présente la solution trouvée ainsi que l'affectation des mouvements aux deux robots pour cet exemple. Cette solution a été trouvée après 35 itérations avec un $CPU = 9,3$ secondes. La durée du cycle obtenue est égale à 251 secondes. Huit autres solutions réalisables ont été trouvées, le détail de ces solutions est présenté dans le Tableau 2.8.

TABLE 2.7 – Résultat pour l'exemple de "P&U"($T = 251$)

Cuve	0	1	2	3	4	5	6	7	8	9	10	11	12
t_i	0	191	73	215	97	156	41	127	194	95	20	153	235
Robot	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_2	R_2	R_2	R_2	R_2

TABLE 2.8 – Les solutions générées par notre algorithme pour l'exemple de "P&U"

T	**nombre d'itérations**	**CPU**
379	1	0.218
365	1	0.234
332	4	1.484
331	6	2.281
281	7	2.5
278	7	2.515
270	17	5.046
258	18	5.421
251	35	9.718

Remarque : avec cet algorithme, nous pouvons obtenir plusieurs solutions réalisables à une itération où plusieurs séquences sont générées. Nous pouvons voir ceci dans le Tableau 2.8 où deux solutions réalisables ont été trouvées à la première itération.

2.4.3.2 Résultat pour l'exemple "ligne 1"

Le Tableau 2.9 présente la solution trouvée ainsi que l'affectation des mouvements aux deux robots pour cet exemple. Cette solution a été trouvée après 5 itérations avec un $CPU = 1,593$ secondes. La durée du cycle obtenue est égale à 301 secondes. Quatre autres solutions réalisables ont été trouvées, le détail de ces solutions est présenté dans le Tableau 2.10.

TABLE 2.9 – Résultat pour l'exemple "ligne 1" ($T = 301$)

Cuve	0	1	2	3	4	5	6	7	8	9	10	11	12
t_i	0	129	188	95	0	70	135	260	43	62	206	26	165
Robot	R_1	R_1	R_1	R_1	R_2	R_1	R_2	R_1	R_1	R_2	R_2	R_2	R_2

TABLE 2.10 – Les solutions générées par notre algorithme pour l'exemple "ligne 1"

T	**nombre d'itérations**	**CPU**
347	1	0.062
314	3	0.906
306	3	0.921
302	4	1.25
301	5	1.593

2.4.3.3 Résultat pour l'exemple "ligne 2"

Le Tableau 2.11 présente la solution trouvée ainsi que l'affectation des mouvements aux deux robots pour cet exemple. Cette solution a été trouvée après une itération avec un $CPU = 0,5$ secondes. La durée du cycle obtenue est égale à 661 secondes. Une autre solution réalisable a été trouvée, le détail de ces solutions est présenté dans le Tableau 2.12.

TABLE 2.11 – Résultat pour l'exemple "ligne 2" ($T = 661$)

Cuve	0	1	2	3	4	5	6	7	8	9	10	11	12	13
t_i	0	195	279	374	579	7	59	199	343	543	35	138	308	332
Robot	R_1	R_1	R_1	R_1	R_2	R_2	R_1	R_2	R_1	R_2	R_2	R_2	R_1	R_2

TABLE 2.12 – Les solutions générées par notre algorithme pour l'exemple "ligne 2"

T	**nombre d'itérations**	**CPU**
722	1	0.468
661	1	0.5

2.4.3.4 Résultat pour l'exemple 4

Le Tableau 2.13 présente la solution trouvée ainsi que l'affectation des mouvements aux deux robots pour cet exemple. Cette solution a été trouvée après 39 itérations avec un $CPU = 14$ secondes. La durée du cycle obtenue est égale à 277 secondes. Quatre autres solutions réalisable ont été trouvées, le détail de ces solutions est présenté dans le Tableau 2.14.

TABLE 2.13 – Résultat pour l'exemple 4 ($T = 277$)

Cuve	0	1	2	3	4	5	6	7	8	9	10	11	12
t_i	0	242	110	180	133	245	80	269	49	157	22	119	46
Robot	R_1	R_1	R_1	R_1	R_1	R_2	R_1	R_2	R_1	R_2	R_2	R_2	R_2

TABLE 2.14 – Les solutions générées par notre algorithme pour l'exemple 4

T	**nombre d'itérations**	**CPU**
379	1	0.234
365	2	0.843
340	2	0.844
290	3	1.109
277	39	14

2.4.3.5 Discussion

Pour tester l'efficacité de notre algorithme, nous avons comparé les résultats obtenus par ceux trouvés dans la littérature. Aussi, nous avons calculé une borne inférieure pour les différents exemples et comparé ces bornes avec les solutions trouvées. Ces résultats sont présentés dans le Tableau 2.15.
Dans ce tableau, T_{best} est la meilleure solution trouvée dans la littérature. Celle ci est marquée par $(*)$ lorsqu'elle est optimale. BI est la borne inférieure.
Nous ajoutons les notations suivantes :

- T : la durée du cycle trouvée par notre algorithme ;
- $NBITER$: le nombre d'itérations passées pour trouver la meilleure solution par notre algorithme ;
- CPU_{best} :meilleur temps d'exécution trouvé dans la littérature.

TABLE 2.15 – Comparaison des résultats

Exemple	n	BI	T_{best}	T	$CPU(s)$	CPU_{best}	$NBITER$
"P&U"	12	168.5	251*	251	9.7183	17.8	35
"ligne 1"	12	301	361	301	1.593	-	5
"ligne 2"	13	661	661*	661	0.5	-	1
"exemple 4"	12	168.5	290	277	14	191	39

Le Tableau 2.15 montre que notre algorithme améliore la durée de cycle pour deux exemples ("ligne 1" et l'exemple 4). Il trouve la solution optimale connue pour l'exemple "P&U" et "ligne 2". De plus, la solution trouvée pour la "ligne 2", où aucune solution optimale n'est connue, coïncide avec la borne inférieure BI. Ceci implique que cette solution est bien une solution optimale.
Le temps de résolution (CPU) varie de 0.5 à 14 secondes dépendamment de l'instance considérée. En effet, les temps de résolution les plus longs sont ceux de l'exemple de "P&U" et de l'"exemple 4". Mais, ces deux exemples sont particulièrement limités en raison des fenêtres de temps très étroites surtout au niveau de la cuve 5 et de la cuve 8, respectivement, qui sont les ressources

critiques. Par conséquent, l'espace de recherche des solutions faisables relatif à ces exemples est réduit et difficile à explorer. En plus, la comparaison des temps de résolution (CPU) avec ceux trouvés dans la littérature permet d'affirmer l'amélioration du temps d'exécution fourni par notre algorithme.

2.5 Conclusion

Dans ce chapitre, nous avons développé une méthode pour la résolution des problèmes de traitement de surface cyclique ou CHSP avec deux robots assurant le déplacement des porteurs entre les différentes cuves. Cette méthode combine deux heuristiques et un programme linéaire mixte. Nous avons généré, par une première heuristique, un ensemble de séquences des mouvements pour un robot, sur la base des travaux de Zhou et Liu (2008). Ensuite, nous avons développé une heuristique qui, partant de chaque séquence générée préalablement, affecte les différents mouvements à l'un des deux robots. Cette heuristique prend en considération, lors de l'affectation, la position de chaque robot sur la ligne et dans le temps. Enfin, nous avons développé un PLM qui permet de vérifier la faisabilité des deux ordonnancements et d'ajuster les valeurs des variables de décision.

Dans une première étape, nous avons testé cet algorithme sur un ensemble de 740 instances. Les résultats trouvés ont montré son efficacité en termes de temps de résolution, ainsi que de déviation du temps de cycle par rapport aux bornes supérieures de chaque instance. Dans une seconde étape, nous avons testé notre algorithme sur cinq exemples pris de la littérature. Les résultats trouvés pour quatre exemples sont identiques ou meilleurs que ceux connus, ce qui prouve à nouveau l'efficacité de cet algorithme.

L'algorithme proposé a été développé avec l'hypothèse que la gamme opéra-

toire des produits à traiter suit l'indexation des cuves. Or cela n'est pas toujours le cas dans tous les ateliers industriels. Pour cela, nous avons étendu notre approche aux cas où la gamme opératoire est différente de l'implantation des cuves. Ce travail fait l'objet du chapitre suivant.

Chapitre 3

Extension du modèle au cas de lignes complexes

Sommaire

3.1 Introduction

Dans le chapitre précédent, nous avons proposé une approche hybride pour la résolution des problèmes de traitement de surface avec deux robots. Cette approche a été définie en tenant compte de plusieurs hypothèses. Parmi ces hypothèses, la gamme opératoire des produits suit l'indexation des cuves. Ainsi, les cuves sont ordonnées d'une manière croissante de la cuve numéro 0 jusqu'à la cuve numéro $n+1$ et chaque produit, lors de son traitement, doit passer de la cuve i vers la cuve $i+1$. Dans ce chapitre nous proposons une généralisation de l'approche présentée dans la section précédente (**section 2.3**) en réalisant une extension portant sur la gamme opératoire des produits lorsqu'elle est différente de l'ordre de l'implantation des cuves dans la ligne. Ainsi, les déplacements en charge ne seront pas toujours d'une cuve vers celle qui lui est adjacente mais tiendront compte de l'ordre des cuves à visiter après chaque traitement c'est-à-dire la gamme opératoire.

Dans ce chapitre, nous allons présenter dans la **section 3.2** le modèle proposé pour la résolution des problèmes **CHSP** avec deux robots et dans le cas où la gamme opératoire des produits est différente de l'implantation des cuves sur la ligne. Dans la **section 3.3**, les résultats obtenus des problèmes industriels trouvés dans la littérature seront présentés et analysés.

3.2 Modèle proposé

3.2.1 Hypothèses et notations

Les mêmes hypothèses que celles décrites dans la **section 2.3.1** sont retenues, exception faite sur les deux points suivants :

- la gamme opératoire est différente de l'ordre d'implantation des cuves. Elle sera définie selon l'exemple à traiter ;
- les cuves peuvent être multi-bacs.

Au cours de ce travail, nous autorisons aux deux robots de dépasser l'extrémité des deux cuves de chargement et déchargement afin de leur donner plus de flexibilité lors de leurs déplacements le long du rail. En effet, si le robot numéro 2 (R_2) doit transporter un porteur d'une cuve i vers la cuve de chargement (cuves de chargement et déchargement associées), alors le robot numéro 1 (R_1) peut s'éloigner suffisamment vers l'extrémité du rail afin de permettre au robot numéro 2 (R_2) d'effectuer son déplacement et de se positionner au dessus de la cuve de chargement. De même, si le robot (R_1) doit transporter un porteur d'une cuve j vers la cuve de traitement située à l'extrémité du rail (cuve de déchargement si les deux cuves de chargement et déchargement sont dissociées ou bien une cuve de traitement dans le cas inverse), le robot (R_2) peut s'éloigner afin de laisser la place au robot (R_1) pour se positionner au dessus de cette cuve.

En tenant un raisonnement semblable à celui mené dans le chapitre précédent, nous proposons un algorithme qui permet la résolution des problèmes du **HSP** Cyclique avec deux robots et avec zone de chevauchement. Il permet la résolution du problème lorsque la gamme opératoire des produits à traiter est différente de la disposition des cuves tout au long de la ligne. L'idée est fondée sur quatre étapes :

- $1^{ère}$ étape : nous conservons le même principe que celui décrit dans la **section 2.3.2.1** pour la génération des ensembles de séquences des mouvements, mais qui tient compte de la gamme opératoire des produits à traiter. Cette méthode sera décrite dans la **section 3.2.2**.
- $2^{ème}$ étape : pour chaque séquence de mouvements, nous proposons une heuristique qui permet d'affecter ces mouvements aux deux robots R_1 et R_2. L'affectation de chaque mouvement tiendra compte de la position de chaque robot sur le rail et l'élimination du risque de collision entre ces robots. Deux sous-séquences seront définies à la fin de l'heuristique, respectivement, pour R_1 et R_2. Cette heuristique sera présentée dans la **section 3.2.3**.

$3^{ème}$ étape : tenant compte des deux sous-séquences de mouvements pour R_1 et R_2, nous proposons un Programme Linéaire Mixte (PLM) afin d'ajuster les valeurs des variables et de vérifier la faisabilité de la solution. Cette faisabilité ne garantit pas la non collision entre les deux robots mais elle garantit le respect des temps de traitement et la disponibilité des robots. Le PLM sera présenté dans la **section 3.2.4**.

$4^{ème}$ étape : pour chaque solution fournie par le PLM , nous proposons une procédure qui permet de tester si une collision existe entre ces deux robots. Si une collision existe, alors cette solution sera rejetée sinon, elle sera acceptée.

Dans ce qui suit, nous posons :

G : vecteur représentant la gamme opératoire des produits à traiter.

$G[i] = j$ signifie que le $i^{ème}$ traitement doit être effectué dans la cuve j, $i = 1, ..., n$, $j = 0, ..., n$.

3.2.2 Génération des mouvements de transport

Nous appliquons la même méthode que celle décrite dans la **section 2.3.2.1** mais en prenant en considération la gamme opératoire du produit à traiter. Ainsi, le calcul des dates de début de transport d'un porteur $m_{G[i]}$ $(i = 1, ..., n)$ sera défini comme suit :

$$m_{G[i]} = m_{G[i-1]} + d_{G[i-1]} + p_{G[i]}, \quad i = 2, ..., n, \quad \text{avec} \quad m_0 = 0 \quad \text{et} \quad G[1] = 0; \tag{3.1}$$

Les différentes valeurs de T_{ij}^k seront calculées par l'équation suivante :

$$T_{ij}^k = (m_{G[j]} - m_{G[i]})/k \quad \text{avec} \quad i = 1, ..., n-1 \quad \text{et} \quad j = i+1, ..., n \quad \text{et} \quad k = 1, ..., n. \tag{3.2}$$

3.2.3 Heuristique d'affectation

Cette heuristique permet un partage dynamique de la ligne en trois zones. Une première zone pour le robot n°1, une deuxième zone pour les deux robots et la troisième zone pour le robot n°2.
Lorsque la gamme ne suit pas l'ordre physique d'implantation des cuves, les déplacements des robots en charge ne s'effectuent plus entre deux cuves adjacentes, mais dépendent de la gamme du produit à traiter. Ainsi, le risque de collision augmente et l'heuristique d'affectation définie dans la **section 2.3.2.2** devient insuffisante pour minimiser le risque de collision et par la suite chercher une bonne affectation des mouvements aux deux robots.

Nous conservons les mêmes notations que celles utilisées dans la **section 2.3.2.2**, auxquelles nous ajoutons :

Pos	le vecteur représentant les numéros des cuves qu'un robot doit atteindre après un déplacement en charge ; $Pos[i] = j$ signifie que le robot doit transporter le porteur de la cuve i vers la cuve j $(i, j = 0, ..., n)$;
d_i	le temps de déplacement du robot en charge entre les cuves i et $Pos[i]$ de la ligne. Ce temps comprend les temps de levée et descente du porteur par le robot et le temps d'égouttage s'il y en a $(i = 0, ..., n)$;
M_i	l'opération d'un déplacement en charge de la cuve i vers la cuve $Pos[i]$ $(i = 0, ..., n)$.

Au cours de l'affectation des mouvements, nous tenons compte de la position des robots après chaque mouvement (en charge ou à vide) ainsi que des dates de fin de ces mouvements. Pour cela, nous calculons ces dates au fur et à mesure qu'un mouvement est affecté.
Dans ce qui suit, nous présenterons, dans un premier temps, les différentes étapes de l'algorithme d'affectation que nous avons proposé, ensuite, nous

décrirons le pseudo code de cet algorithme (Algorithme 7).

- initialisation :
  $ET_1[0] = d_0$
  $ET_2[0] = 0$
- à une étape $curs$ ($curs = 1, ..., n$) de l'algorithme, supposons que $l_1 - 1$ et $l_2 - 1$ mouvements sont déjà affectés, respectivement, à R_1 et R_2 et le mouvement à affecter est $M_{O[curs]}$:

$$EV_r[curs - 1] = ET_r[curs - 1] + C_{Pos[O_r[l_r-1]],O[curs]} \quad r = 1, 2 \tag{3.3}$$

$$ET_r[curs] = EV_r[curs - 1] + d_{O[curs]} \quad r = 1, 2 \tag{3.4}$$

Pour chaque séquence de mouvements O, l'heuristique proposée permet d'affecter ces mouvements aux deux robots présents sur la ligne. Supposant qu'à l'itération $curs$, un robot doit effectuer l'opération $M_{O[curs]}$; et supposant qu'à cet instant, les deux robots R_1 et R_2 se trouvent, respectivement, dans deux positions différentes j et k (R_1 au dessus de la cuve j et R_2 au dessus de la cuve k) et que $j < k$, plusieurs cas se présentent :

Cas n°1 : $O[curs] \geq k$ et $Pos[O[curs]] > k$. Dans ce cas l'opération $M_{O[curs]}$ sera affectée au robot R_2 (voir Figure 3.1) ;

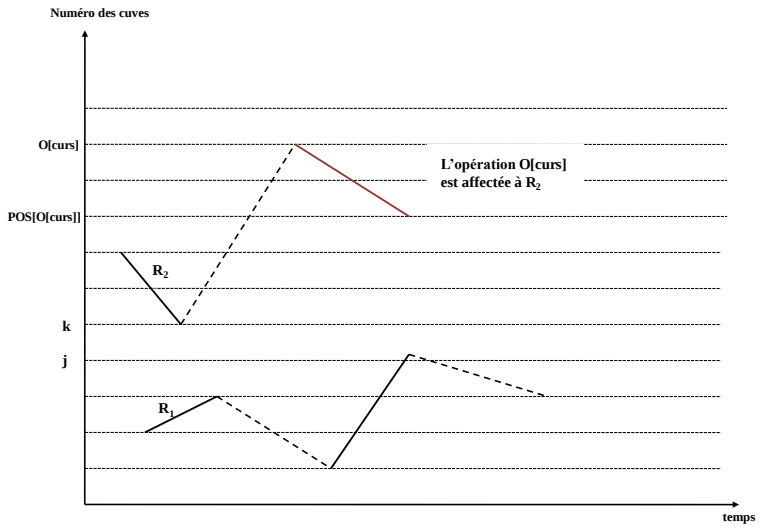

FIGURE 3.1 – Affectation des mouvements aux deux robots (cas n°1)

Cas n°2 : $O[curs] \geq k$ et $Pos[O[curs]] < k$. Dans un premier temps, nous calculons la date à laquelle le robot R_2 dépose un porteur à $Pos[O[curs]]$. Si cette date est supérieure à la date à laquelle le robot R_1 dépose un porteur à la cuve j, alors le mouvement $M_{O[curs]}$ sera affecté à R_2 ; sinon, nous cherchons si une collision peut se produire entre les deux robots. Notre stratégie consiste à vérifier si la date à laquelle R_2 arrive et dépose un porteur à la cuve $Pos[O[curs]]$ est incluse dans un intervalle de temps défini par la date de début et la date de fin de deux mouvements consécutifs (à vide puis en charge) effectués par le robot R_1 en plus de la condition que la cuve $Pos[O[curs]]$ se trouve au dessous d'au moins une des cuves sujet de deux mouvements de R_1. Si cette condition est vérifiée alors une collision est supposée exister, l'opération $M_{O[curs]}$ sera affectée alors à R_1 (Figure 3.2 (a)), dans le cas inverse l'opération $M_{O[i]}$ sera affectée à R_2 (Figure 3.2(b)).

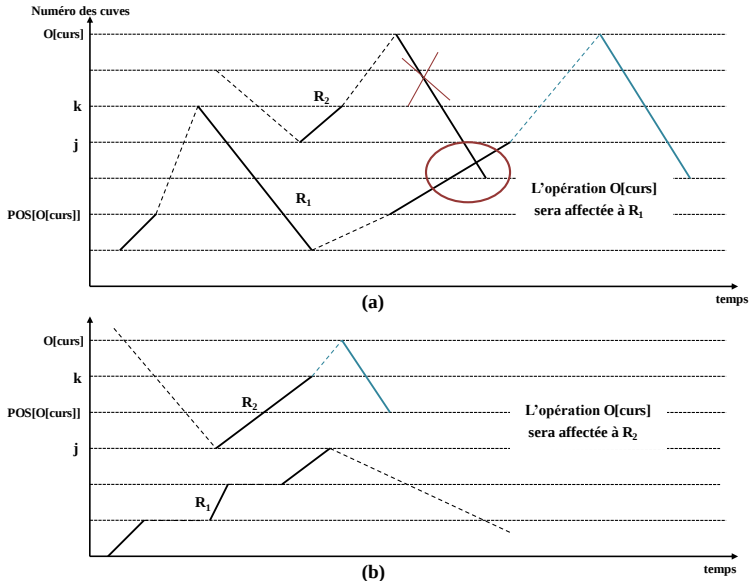

FIGURE 3.2 – Affectation des mouvements aux deux robots (cas n°2)

Ce test est codé par l'Algorithme 3 décrit ci dessous.

Algorithme 3 : Recherche-conflit 1

Input : $curs_1$, $curs_2$.
Output : $indice$.
Initialisation : $indice = 0$.
pour $(cpt \leftarrow (curs_1 - 1)$ à $1)$ **faire**
  si $(Pos[O_1[cpt]] > Pos[O[curs]]$ et $ET_1[cpt] > EV_2[curs - 1] + d_{O[curs]}$ et $EV_1[cpt - 1] < EV_2[curs - 1] + d_{O[curs]})$ **alors**
    $indice = indice + 1$;

Cas n°3 : $O[curs] \leq j$ et $Pos[O[curs]] < j$. Dans ce cas, l'opération $M_{O[curs]}$ sera affectée au robot R_1 comme le montre la Figure 3.3.

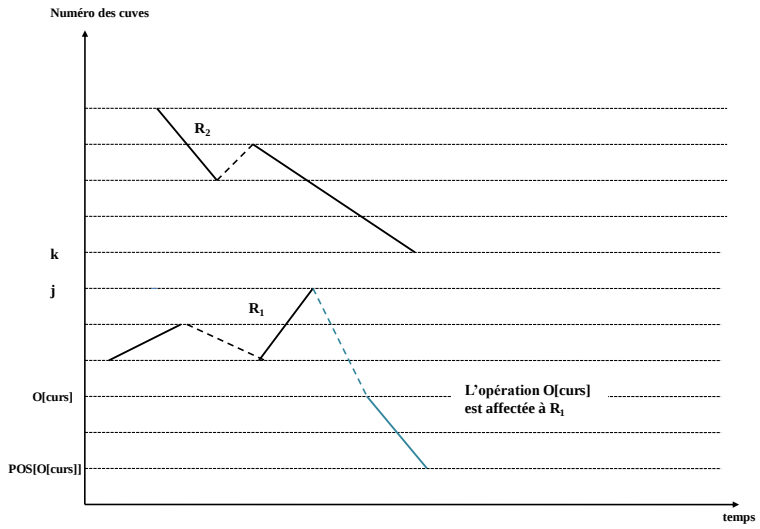

FIGURE 3.3 – Affectation des mouvements aux deux robots (cas n°3)

Cas n°4 : $O[curs] \leq j$ et $Pos[O[curs]] > j$. Alors, nous calculons la date à laquelle le robot R_1 dépose un porteur à $Pos[O[curs]]$. Si cette date est supérieure à la date à laquelle le robot R_2 dépose un porteur à la cuve k alors le mouvement $O[curs]$ sera affecté à R_1. Sinon, nous cherchons si une collision peut se produire entre les deux robots.

Comme dans le deuxième cas, nous cherchons à vérifier si la date à laquelle R_1 arrive et dépose un porteur à la cuve $Pos[O[curs]]$ est incluse dans un intervalle de temps défini par la date de début et la date de fin de deux mouvements consécutifs (à vide puis en charge) effectués par le robot R_2 en plus de la condition que la cuve $Pos[O[curs]]$ se trouve au dessus d'au moins une des cuves sujet de deux mouvements de R_2. Si cette condition est vérifiée alors une collision est supposée exister. L'opération $M_{O[curs]}$ sera affectée à R_2 (Figure 3.4 (b)), dans le cas inverse l'opération $M_{O[i]}$ sera affectée à R_1 (Figure 3.4(a)).

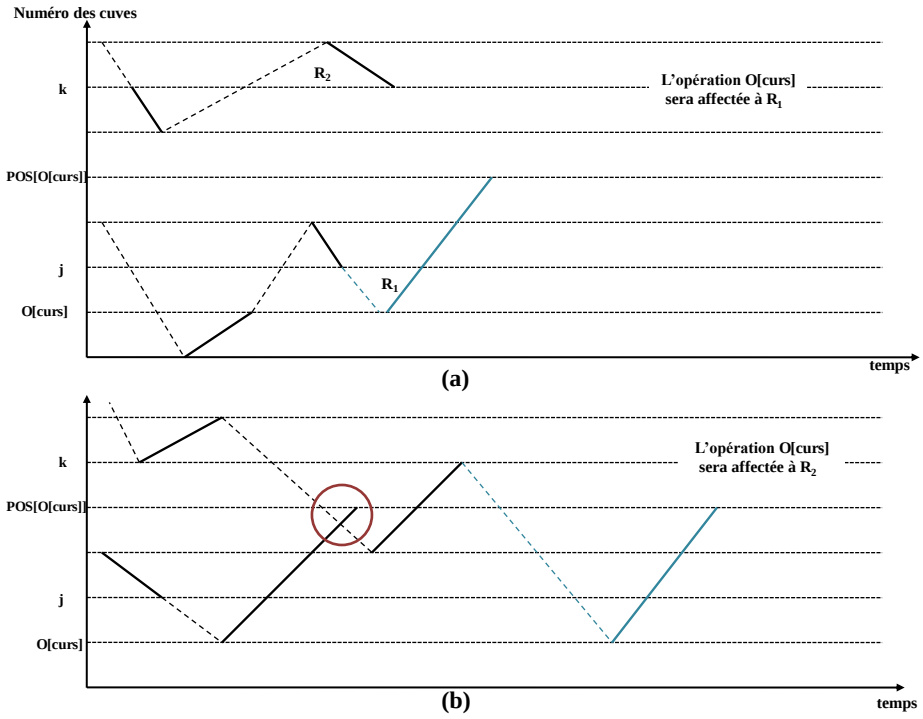

FIGURE 3.4 – Affectation des mouvements aux deux robots (cas n°4)

Ce test est codé comme décrit à l'Algorithme 4 ci dessous.

Algorithme 4 : Recherche-conflit 2

Input : $curs_1$, $curs_2$.
Output : $indice$.
Initialisation : $indice = 0$.
pour $(cpt \leftarrow (curs_2 - 1)$ à $1)$ **faire**
  si $(Pos[O'_2[cpt]] > Pos[O[curs]]$ et $ET_2[cpt] > EV_1[curs - 1] + d_{O[curs]}$ et $EV_2[cpt - 1] < EV_1[curs - 1] + d_{O[curs]})$ **alors**
    $indice = indice + 1$;

Cas n°5 : $j < O[curs] < k$. Dans ce cas, nous identifions le robot qui pourrait arriver vide à la cuve $O[curs]$ avant l'autre. Si R_1 arrive le premier ($EV_1[.] < EV_2[.]$), alors nous testons si la possibilité d'une collision peut se produire entre les deux robots, c'est-à-dire entre R_1 effectuant $M_{O[curs]}$ et un mouvement antérieur (numéro "cpt") effectué par R_2. Autrement dit, nous comparons les positions des cuves $Pos[O[curs]]$, $O_2[cpt]$ et $Pos[O_2[cpt]]$; ainsi, si $Pos[O[curs]]$ est supérieure à l'une des deux autres valeurs alors on suppose qu'une collision risque de se

produire et ainsi le mouvement sera affecté à R_2 (Figure 3.5). Sinon le mouvement sera affecté à R_1.

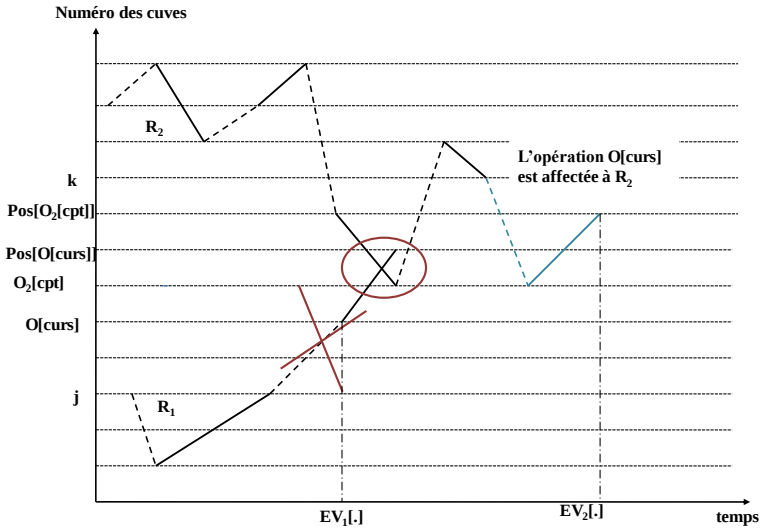

FIGURE 3.5 – Affectation des mouvements aux deux robots (cas n°5)

Le test qui permet de comparer les positions des cuves est codé par l'Algorithme 5 décrit ci dessous.

Algorithme 5 : Recherche-conflit 3

Input : $curs_1$, $curs_2$.
Output : $indice$.
Initialisation : $indice = 0$.
pour ($cpt \leftarrow (curs_2 - 1)$ à 1) **faire**
  si ($O'_2[cpt] < Pos[O[curs]]$ ou $Pos[O'_2[cpt]] < Pos[O[curs]]$) **alors**
    $indice = indice + 1$;

Cas n°6 : $j < O[curs] < k$ et R_2 arrive le premier ($EV_2[.] < EV_1[.]$), alors nous testons si une collision peut se produire entre les deux robots, c'est-à-dire entre R_2 effectuant $M_{O[curs]}$ et un mouvement antérieur (numéro "cpt") effectué par R_1. Autrement dit, nous comparons les positions des cuves $Pos[O[curs]]$, $O_1[cpt]$ et $Pos[O_1[cpt]]$. Si $Pos[O[curs]]$ est inférieure à l'une des deux autres valeurs, alors on suppose qu'une collision risque de se produire et ainsi le mouve-

ment sera affecté à R_1, sinon le mouvement sera affecté à R_2. Le test qui permet de comparer les positions des cuves est codé par l'Algorithme 6 décrit ci dessous.

Algorithme 6 : Recherche-conflit 4

Input : $curs_1$, $curs_2$.
Output : $indice$.
Initialisation : $indice = 0$.
pour ($cpt \leftarrow (curs_2 - 1)$ à 1) **faire**
  si ($O_1[cpt] > Pos[O[curs]]$ ou $Pos[O_1[cpt]] > Pos[O[curs]]$) **alors**
    $indice = indice + 1$;

Algorithme 7 : Algorithme d'Affectation des Mouvements

Input : n, Pos, O : une séquence donnée.
Output : n_1, n_2, O_1,O'_2.
Initialisation : $curs_1 = 1$, $curs_2 = 1$, $O_1[0] = 0$, $O_2[0] = n$, $n_1 = 1$, $n_2 = 0$, $ET_1[0] = d_0$, $ET_2[0] = 0$.
pour ($curs \leftarrow 1$ à n) **faire**
  $EV_1[curs-1] = ET_1[curs-1] + C_{Pos[O_1[curs_1-1]],O[curs]}$;
  $EV_2[curs-1] = ET_2[curs-1] + C_{Pos[O'_2[curs_2-1]],O[curs]}$;
  si ($O[curs] \geq Pos[O'_2[curs_2-1]]$) **alors**
    si ($Pos[O[curs]]) > Pos[O'_2[curs_2-1]]$) **alors**
      /On affecte $M_{O[curs]}$ à R_2/
    sinon
      si ($ET_1[curs-1] < ET_2[curs-1]$) **alors**
        /On affecte $M_{O[curs]}$ à R_2/
      sinon
        $indice = Recherche - conflit \quad 1(curs_1, curs_2)$;
        si ($indice > 0$) **alors**
          /On affecte l'opération $M_{O[curs]}$ à R_1/
        sinon
          /On affecte l'opération $M_{O[curs]}$ à R_2/
  sinon
    si ($O[curs] \leq Pos[O_1[curs_1-1]]$) **alors**
      si ($Pos[O[curs]]) < Pos[O_1[curs_1-1]]$ **alors**
        /On affecte $M_{O[curs]}$ à R_1/
      sinon
        si ($ET_2[curs-1] \leq ET_1[curs-1]$) **alors**
          /On affecte $M_{O[curs]}$ à R_1/
        sinon
          $indice = Recherche - conflit \quad 2(curs_1, curs_2)$;
          si ($indice > 0$) **alors**
            /On affecte l'opération $M_{O[curs]}$ à R_2/
          sinon
            /On affecte l'opération $M_{O[curs]}$ à R_1/
    sinon
      si ($EV_1[curs-1] \leq EV_2[curs-1]$) **alors**
        $indice = Recherche - conflit \quad 3(curs_1, curs_2)$;
        si ($indice > 0$) **alors**
          /On affecte l'opération M_k à R_2/
        sinon
          /On affecte l'opération M_k à R_1/
      sinon
        $indice = Recherche - conflit \quad 4(curs_1, curs_2)$;
        si ($indice > 0$) **alors**
          /On affecte l'opération M_k à R_1/
        sinon
          /On affecte l'opération M_k à R_2/
pour $j \leftarrow 0$ à $n_2 - 1$ **faire**
  $O_2[j] = O'_2[j+1]$

– si un mouvement $O[curs]$ ($curs = 1, ..., n$) est affecté à R_1 (*/on affecte $M_{O[curs]}$ à R_1/*) alors ces instructions sont effectuées :

$$\boxed{\begin{array}{l} O_1[curs_1] = O[curs] \\ curs_1 = curs_1 + 1 \\ n_1 = n_1 + 1 \\ ET_1[curs] = EV_1[curs-1] + d_{O[curs]} \\ ET_2[curs] = ET_2[curs-1] \end{array}}$$

– si un mouvement $O[curs]$ $(curs = 1, ..., n)$ est affecté à R_2 (*/on affecte $M_{O[curs]}$ à R_2/*) alors ces instructions sont effectuées :

$$\boxed{\begin{array}{l} O'_2[curs_2] = O[curs] \\ curs_2 = curs_2 + 1 \\ n_2 = n_2 + 1 \\ ET_2[curs] = EV_2[curs - 1] + d_{O[curs]} \\ ET_1[curs] = ET_1[curs - 1] \end{array}}$$

3.2.4 PLM proposé

Le PLM que nous avons développé dans la **section 2.3.2.3** du chapitre 2 est généralisé dans cette section. En effet, nous avons pris en considération la gamme opératoire du produit à traiter (G) lors du développement des contraintes. Les équations (3.6-3.19) présenteront le PLM modifié, essentiellement au niveau des indices i remplacés par $G[i]$:

$$\text{Minimiser} \quad T \tag{3.5}$$

Sous contraintes :

$$t_0 = 0 \tag{3.6}$$

$$t_{G[i+1]} - \left(t_{G[i]} + d_{G[i]}\right) \leq U_{G[i+1]}, \quad i = 0, ..., n-1 \tag{3.7}$$

$$T + t_{G[i+1]} - \left(t_{G[i]} + d_{G[i]}\right) \leq U_{G[i+1]} + M.\left(1 - s_{G[i+1]}\right), \quad i = 0, ..., n-1 \tag{3.8}$$

$$t_{G[i+1]} - \left(t_{G[i]} + d_{G[i]}\right) \geq L_{G[i+1]} - M.s_{G[i+1]}, \quad i = 0, ..., n-1 \tag{3.9}$$

$$T + t_{G[i+1]} - \left(t_{G[i]} + d_{G[i]}\right) \geq L_{G[i+1]}, \quad i = 0, ..., n-1 \tag{3.10}$$

$$t_{G[i+1]} - \left(t_{G[i]} + d_{G[i]}\right) + \delta - \left(U_{G[i+1]} + \delta\right)\left(1 - s_{G[i+1]}\right) \leq 0, \quad i = 0, ..., n-1 \tag{3.11}$$

$$t_{O_1[i_1+1]} - t_{O_1[i_1]} \geq d_{O_1[i_1]} + c_{Pos[O_1[i_1]],O_1[i_1+1]}, \quad i_1 = 0, ..., n_1 - 2 \tag{3.12}$$

$$t_{O_2[i_2+1]} - t_{O_2[i_2]} \geq d_{O_2[i_2]} + c_{Pos[O_2[i_2]],O_2[i_2+1]}, \quad i_2 = 0, ..., n_2 - 2 \tag{3.13}$$

$$T + t_{O_1[0]} - t_{O_1[n_1-1]} \geq d_{O_1[n_1-1]} + c_{Pos[O_1[n_1-1]],O_1[0]} \tag{3.14}$$

$$T + t_{O_2[0]} - t_{O_2[n_2-1]} \geq d_{O_2[n_2-1]} + c_{Pos[O_2[n_2-1]],O_2[0]} \tag{3.15}$$

$$t_{G[i]} \leq T \tag{3.16}$$

$$t_{G[i]} \geq 0, \quad i = 0, ..., n \tag{3.17}$$

$$s_{G[i]} \in [0, 1], i = 0, ..., n \tag{3.18}$$

$$T \geq 0 \tag{3.19}$$

Avec M un entier positif très grand.

3.2.5 Test de collision

L'algorithme d'affectation présenté précédemment (Algorithme 7) réduit les risques de collision, mais ne les élimine pas. En plus le PLM défini dans la section précédente ne contient pas les contraintes de non collision entre les deux robots. Ainsi, toute solution fournie par le PLM ne garantit pas toujours l'absence de collision entre les deux robots. Pour cela, nous allons définir une procédure qui permet de tester l'existence ou pas de collision entre deux mouvements différents effectués par les deux robots R_1 et R_2.
La procédure définie dans le chapitre précédent (**section 2.3.2.4**) est insuffisante pour effectuer ce test vue que la gamme opératoire est différente de l'implantation physique des cuves et par suite les mouvements en charge varient selon l'ordre des traitements des produits.

Manier (1994) a identifié 50 configurations possibles d'interférence entre deux robots. 8 parmi ces configurations identifiées ne présentent aucun risque d'interférence. Elles ont été déterminées en fonction du type de partage de la ligne en zones d'affectation des robots. Manier (1994) a identifié 4 types de zones d'affectation : zones d'affectation disjointes continues ou discontinues et zones d'affectation non disjointes continues ou discontinues.

Supposons que le robot R_1 doive transporter un porteur de la cuve j_1 vers la cuve j_2 et que le robot R_2 doive transporter un porteur de la cuve k_1 vers la cuve k_2. Lorsque les quatre cuves considérées sont distinctes, Manier (1994) a déterminé 24 configurations possibles d'interférences entre les deux robots. Elle a pris en considération, aussi, le cas où l'une des cuves est commune aux opérations que les deux robots doivent effectuer. Il s'agit, soit d'une cuve pivot ($j_2 = k_1$ ou $k_2 = j_1$), soit d'une cuve multi-fonctions ($j_1 = k_1$ ou $j_2 = k_2$). Dans ce cas, 24 autres configurations possibles sont alors identifiées. Finalement, ces deux mouvements peuvent ne faire intervenir que deux cuves seulement à savoir les cuves pivots. Dans ce cas, seules 2 configurations sont recensées. Manier (1994) a, aussi, défini les différentes contraintes à respecter pour interdire les collisions potentielles pour chaque type d'interférences. Zhou et Liu (2008) ont proposé 20 cas de positions relatives de deux mouvements (c'est-à-dire numéros des cuves composant un mouvement en charge) effectué respectivement par les deux robots. Pour chaque cas, des contraintes qui gèrent la non collision sont proposées en fonction des positions des mouvements précédents les mouvements actuels. Connaissant à priori les deux sous-séquences relatives aux deux robots, les auteurs ont introduit les contraintes de non collision dans le PLM qu'ils ont proposé selon le cas correspondant c'est-à-dire les positions des cuves correspondantes aux deux sous-séquences.

Notre objectif est ici de tester si une collision entre deux robots peut se produire sans prendre en considération l'ensemble des contraintes qui gèrent la non collision entre les deux robots. Pour cela, partant de deux séquences O_1 et O_2 relatives au deux robots R_1 et R_2 nous proposons une méthode qui permet de tester si une collision existe entre deux mouvements effectués par les deux robots. Nous nous sommes inspirées du diagramme de Gantt qui représente graphiquement une solution et qui permet de schématiser l'ensemble des mouvements des robots (en charge ou à vide) tout au long d'un cycle.

En effet, la fonction représentant le déplacement d'un robot en charge peut être croissante ou décroissante selon la position de la cuve à visiter pour le prochain traitement (contrairement à l'hypothèse où la gamme des produits suit l'implantation physique des cuves).

Soit la fonction affine f définie par :

$$f(t) = a * t + b$$

Si le déplacement d'un robot est défini par la fonction f, alors nous aurions :

t les dates au cours desquelles un robot est en déplacement en charge ou à vide ;

$f(t)$ les différentes positions des cuves.

Cette fonction est représentée par une droite ($y = a * t + b$), dont a est le coefficient directeur et b l'ordonnée à l'origine. Soit deux points t_1 et t_2 et leurs images correspondantes $f(t_1)$ et $f(t_2)$ par la fonction f. Les deux valeurs a et b sont alors définies par :

$$a = \frac{f(t_2) - f(t_1)}{t_2 - t_1}$$

$$b = \frac{t_2 * f(t_1) - t_1 * f(t_2)}{t_2 - t_1}$$

Un déplacement du robot en charge entre deux cuves i et $Pos[i]$ est représenté au niveau du diagramme de Gantt par un segment dont l'équation est définie sur l'intervalle $[t_i, t_i + d_i]$ par :

$$y = (\frac{Pos[i] - i}{d_i}) * t + i + (\frac{i - Pos[i]}{d_i} * t_i)$$

Sachant que $f(t_i) = i$ et $f(t_i + d_i) = Pos[i]$ (voir Figure 3.6).

Un déplacement du robot à vide entre deux cuves $Pos[i]$ et j est représenté au niveau du diagramme de Gantt par un segment dont l'équation est définie sur l'intervalle $[t_i + d_i, t_i + d_i + c_{Pos[i],j}]$ par :

$$y' = (\frac{j - Pos[i]}{c_{Pos[i],j}}) * t' + Pos[i] + \frac{(t_i + d_i)(Pos[i] - j)}{c_{Pos[i],j}}$$

Sachant que $f(t_i + d_i) = Pos[i]$ et $f(t_i + d_i + c_{Pos[i],j}) = j$ (voire Figure 3.6).

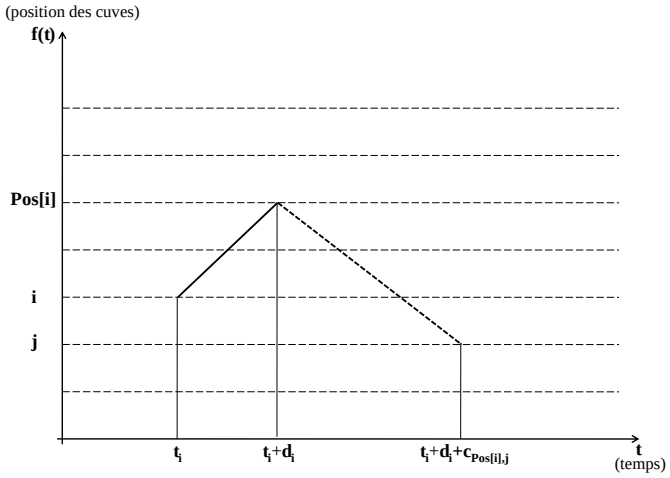

FIGURE 3.6 – Coordonnées des points représentant les déplacements d'un robot

Une collision se produit lorsque les deux robots se croisent soit en effectuant chacun le même mouvement (en charge ou à vide) ou bien en effectuant deux mouvements différents (l'un en charge et l'autre à vide). Donc, graphiquement ceci correspond à l'intersection des deux segments représentant deux mouvements. Trois cas peuvent se présenter :

Cas n°1 : intersection entre deux déplacements en charge M_{i_1} et M_{i_2}. Ces

deux mouvements seront définis par les équations (3.20) et (3.21) ci-après :

$$y_1 = (\frac{Pos[i_1] - i_1}{d_{i_1}}) * t_1 + i_1 + (\frac{i_1 - Pos[i_1]}{d_{i_1}}) * t_{i_1} \tag{3.20}$$

$$y_2 = (\frac{Pos[i_2] - i_2}{d_{i_2}}) * t_2 + i_2 + (\frac{i_2 - Pos[i_2]}{d_{i_2}}) * t_{i_2} \tag{3.21}$$

Le point d'intersection est tels que :

$$\begin{cases} y_1 = y_2 \\ t_1 = t_2 \end{cases}$$

En remplaçant t_2 par t_1 dans l'équation (3.21), ce qui donne :

$$t_1 = \frac{B}{A} \tag{3.22}$$

où :

$$A = \frac{Pos[i_1] - i_1}{d_{i_1}} - \frac{Pos[i_2] - i_2}{d_{i_2}}$$

$$B = i_2 - i_1 + \frac{t_{i_2}(i_2 - Pos[i_2])}{d_{i_2}} - \frac{t_{i_1}(i_1 - Pos[i_1])}{d_{i_1}}$$

Proposition 1 :
Si $t_1 \in [t_{i_1}, t_{i_1} + d_{i_1}] \cap [t_{i_2}, t_{i_2} + d_{i_2}]$ alors les deux robots sont en collision.

La Figure 3.7 permet d'illustrer la proposition 1 en présentant le cas où une collision existe (a) et le cas où une collision n'existe pas (b) entre deux mouvements en charge effectués respectivement par les deux robots R_1 et R_2.

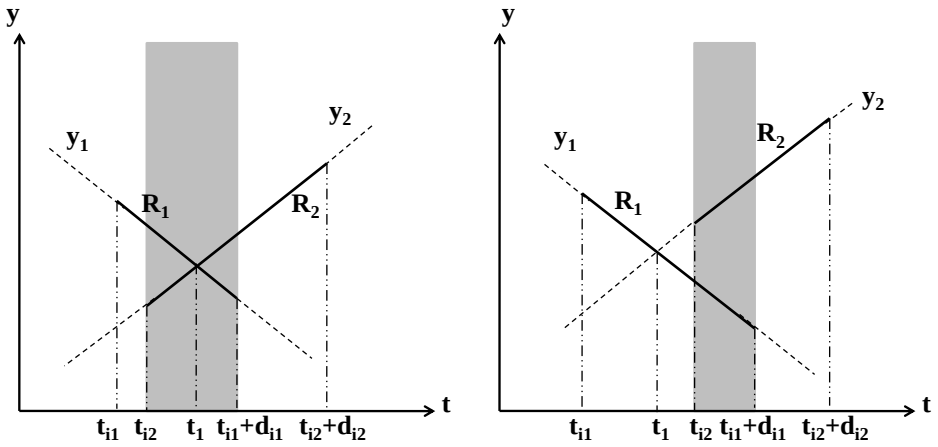

FIGURE 3.7 – illustration de la proposition 1

Cas n°2 : intersection entre deux déplacements à vide (de $Pos[i_1]$ à j_1 et de $Pos[i_2]$ à j_2). Ces deux mouvements seront définis par les équations suivantes :

$$y'_1 = (\frac{j_1 - Pos[i_1]}{c_{Pos[i_1],j_1}}) * t'_1 + Pos[i_1] + \frac{(t_{i_1} + d_{i_1}) * (Pos[i_1] - j_1)}{c_{Pos[i_1],j_1}} \tag{3.23}$$

$$y'_2 = (\frac{j_2 - Pos[i_2]}{c_{Pos[i_2],j_2}}) * t'_2 + Pos[i_2] + \frac{(t_{i_2} + d_{i_2}) * (Pos[i_2] - j_2)}{c_{Pos[i_2],j_2}} \tag{3.24}$$

Le point d'intersection est tel que :

$$\begin{cases} y'_1 = y'_2 \\ t'_1 = t'_2 \end{cases}$$

Nous remplaçons t'_2 par t'_1 dans l'équation 3.24, ce qui donne :

$$t'_1 = \frac{B'}{A'} \tag{3.25}$$

où :

$$A' = \frac{j_1 - Pos[i_1]}{c_{Pos[i_1],j_1}} - \frac{j_2 - Pos[i_2]}{c_{Pos[i_2],j_2}}$$

$$B' = Pos[i_2] - Pos[i_1] + \frac{(t_{i_2} + di_2)(Pos[i_2] - j_2)}{c_{Pos[i_2],j_2}} - \frac{(t_{i_1} + di_1)(Pos[i_1] - j_1)}{c_{Pos[i_1],j_1}}$$

Proposition 2 :
Si $t'_1 \in [t_{i_1} + d_{i_1}, t_{i_1} + d_{i_1} + c_{Pos[i_1],j_1}] \cap [t_{i_2} + d_{i_2}, t_{i_2} + d_{i_2} + c_{Pos[i_2],j_2}]$ alors les deux robots sont en collision.

Cas n°3 : intersection entre deux déplacements l'un à vide et l'autre en charge. Ces deux mouvements seront définis par les équations suivantes :

$$y''_1 = (\frac{j_1 - Pos[i_1]}{c_{Pos[i_1],j_1}}) * t''_1 + Pos[i_1] + \frac{(ti_1 + d_{i_1}) * (Pos[i_1] - j_1)}{c_{Pos[i_1],j_1}} \tag{3.26}$$

$$y''_2 = (\frac{Pos[i_2] - i_2}{d_{i_2}}) * t_2 + i_2 + (\frac{i_2 - Pos[i_2]}{d_{i_2}}) * t_{i_2} \tag{3.27}$$

Le point d'intersection est tel que :

$$\begin{cases} y''_1 = y''_2 \\ t''_1 = t''_2 \end{cases}$$

Nous remplaçons t''_2 par t''_1 dans l'équation 3.27, on aura :

$$t''_1 = \frac{B''}{A''} \tag{3.28}$$

où :

$$A'' = \frac{j_1 - Pos[i_1]}{c_{Pos[i_1],j_1}} - (\frac{Pos[i_2] - i_2}{d_{i_2}})$$

$$B'' = i_2 - Pos[i_1] + \frac{(i_2 - Pos[i_2]) * t_{i_2}}{d_{i_2}} - \frac{(ti_1 + d_{i_1}) * (Pos[i_1] - j_1)}{c_{Pos[i_1],j_1}}$$

Proposition 3 :
Si $t''_1 \in [t_{i_2}, t_{i_2} + d_{i_2}] \cap [t_{i_1} + d_{i_1}, t_{i_1} + d_{i_1} + c_{Pos[i_1],j_1}]$ alors les deux robots sont en collision.

Basé sur les deux séquences fournies par l'algorithme d'affectation et les dates t_i définies par le PLM, ce test consiste à vérifier si une collision existe entre deux mouvements effectués, respectivement, par les deux robots. Partant d'une séquence de mouvements pour R_1, pour laquelle chaque mouvement $O_1[i_1], \quad (i_1 = 0, ..., n_1 - 1)$ est défini par une date de début et une date de fin, nous cherchons s'il existe un mouvement effectué par R_2 tel que ces deux conditions sont remplies :

- la date de début du mouvement effectué par R_2 est inférieure à la date de fin du mouvement $O_1[i_1]$;
- la date de fin du mouvement effectué par R_2 est supérieure à la date de début du mouvement $O_1[i_1]$.

Si un tel mouvement existe alors nous calculons la valeur de t_1 ou de t'_1, de t''_1 selon la nature des deux mouvements. Une fois l'une des valeurs calculée, nous testons si elle appartient aux intervalles définis dans les propositions. Par suite, nous pouvons vérifier si une collision existe ou non.

3.2.6 Extension au cas de lignes comportant des cuves multi-bacs

Pour augmenter la productivité d'une ligne de traitement de surface, certaines cuves dont le traitement a la plus grande durée sont démultipliées. Ainsi, la capacité d'accueil d'une cuve multi-bacs devient égale au plus au nombre des bacs.

Considérons les mêmes hypothèses que celle des cuves simples, à savoir : chaque porteur ne sera traité que dans un seul bac. Nous ajoutons l'hypothèse émise par Shapiro et Nuttle (1988). Ces auteurs considèrent que les porteurs sont retirés

des bacs d'une cuve selon la loi FIFO (premier entré, premier sorti).

Dans cette situation, la durée de traitement d'un porteur dans une cuve est toujours la même quelque soit le bac où il est placé. Par contre l'occupation des cuves n'est pas la même au début de chaque cycle. En effet, au cours de chaque cycle un porteur entre et un autre sort de chaque cuve. Pour l'affectation des porteurs aux différents bacs de la cuve i, nous avons adopté la stratégie suivante : le bac qui recevra un porteur est le bac vide le plus proche de la cuve $i-1$.

Soit b_i : le nombre des bacs pour la cuve i.

Si tous les bacs de la cuve i sont utilisés, alors il faudra b_i cycles pour qu'un porteur termine son traitement dans la cuve i.

Posons l'hypothèse que : tous les bacs sont utilisés.

La figure 3.8 donne un aperçu sur la manière dont les bacs sont occupés. Ainsi, si le chargement d'une cuve multi-bacs s'effectue avant son déchargement alors ces deux opérations ne s'effectuent pas au niveau du même bac, car elles sont relatives à deux produits différents (cas (a)). Dans ce cas, les deux opérations associées à un même produit, donc s'effectuant dans le même bac, ont lieu dans deux cycles successifs. Par contre, si le chargement d'une cuve multi-bacs s'effectue après son déchargement alors ces deux opérations peuvent s'effectuer au niveau du même bac (cas (b)).

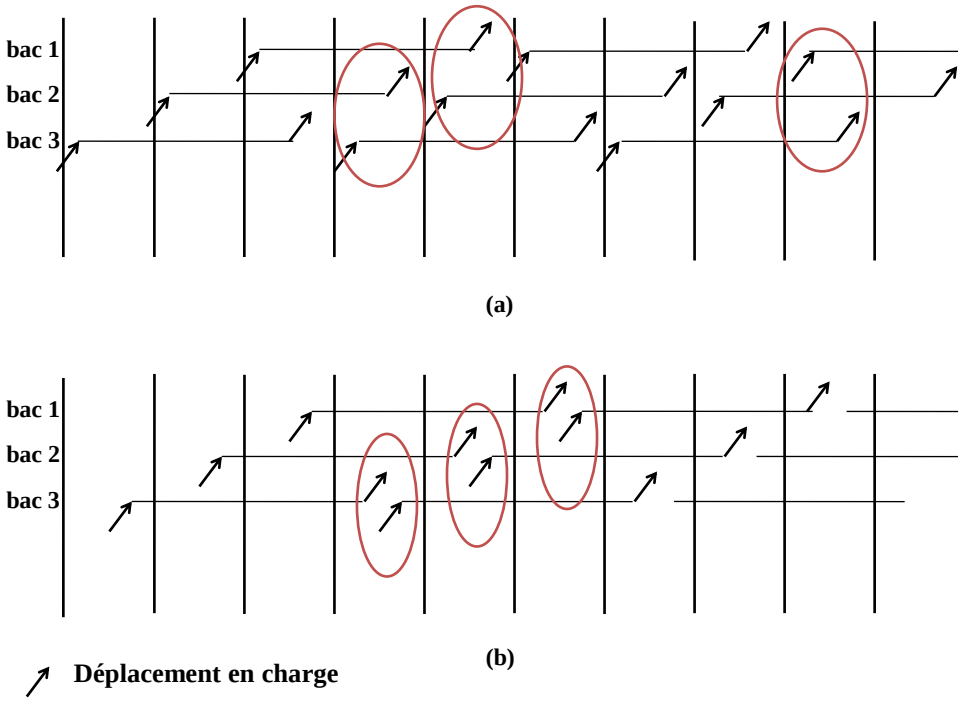

FIGURE 3.8 – Affectation des porteurs aux cuves

Prenons l'exemple de la ligne *Zinc*, dont les temps de traitement minimaux et maximaux ainsi que le nombre de bacs pour la cuve multi-bacs sont présentés dans l'Annexe C.

Si le temps de traitement dans la cuve numéro 15 est égal à la durée de traitement minimale ($L_{15} = 1680$), alors l'intervalle de temps entre l'entrée d'un porteur et la sortie d'un autre dans un cycle sera égal à $(1680 - 3 * T)$ (voir Figure 3.9).

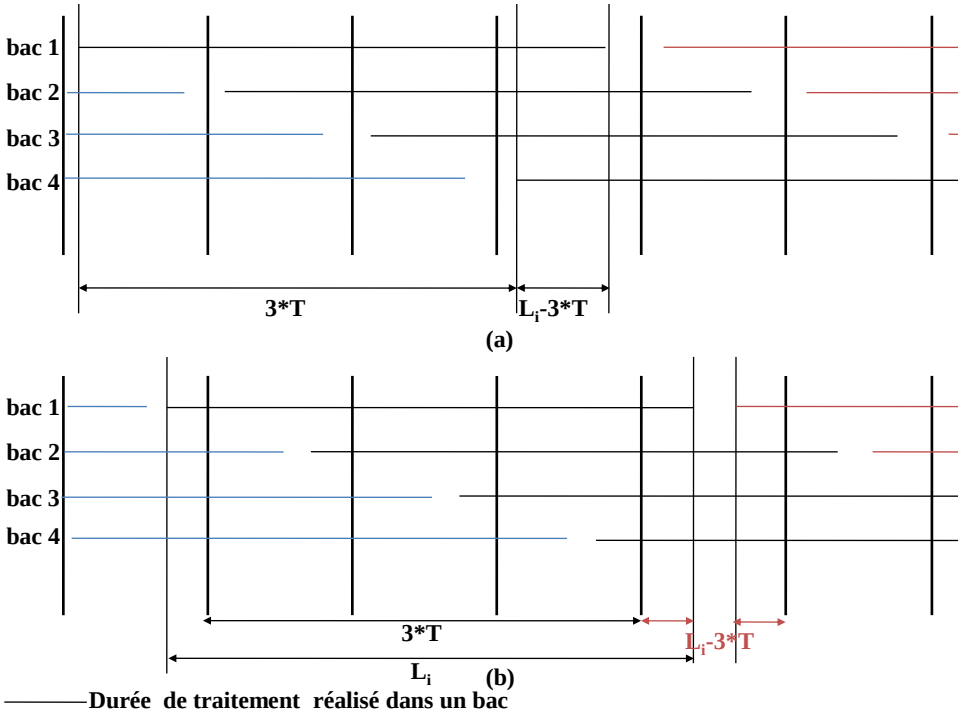

FIGURE 3.9 – Temps de traitement dans la cuve multi-bacs

Soientt :

$L_i^{'}$: la durée de traitement minimale dans la cuve multi-bacs i dans un cycle,

$U_i^{'}$: la durée de traitement maximale dans la cuve multi-bacs i dans un cycle.

Si la durée de traitement dans la cuve multi-bacs est égale à (L_i) ou (U_i), alors les valeurs $L_i^{'}$ et $U_i^{'}$ seront définies par :

$$L_i^{'} = L_i - (b_i - 1) * T \tag{3.29}$$

$$U_i^{'} = U_i - (b_i - 1) * T \tag{3.30}$$

Zhou et Li (2003) ont défini des bornes pour la valeur de T. En effet, si on considère la durée de traitement dans un bac durant un cycle au cours duquel aucun porteur n'y entre et n'y sort, alors cette durée sera égale à T.

En se basant sur les travaux de Manier (1994), qui a démontré que pour une

durée opératoire p_i effectivement réalisée durant le processus de traitement dans la cuve i, la relation suivante est définie :

$$(b_i - 1)T \leq p_i \leq b_i T$$

Ce qui donne séparément :

$$p_i \leq b_i T$$

ET

$$(b_i - 1)T \leq p_i$$

Ces deux relations peuvent s'écrire comme suit sachant que $b_i \neq 0$:

$$T \geq p_i / b_i$$

ET

$$p_i / (b_i - 1) \geq T$$

Ce qui donne :

$$p_i / b_i \leq T \leq p_i / (b_i - 1)$$

– si la durée de traitement p_i est égale à L_i, alors la valeur de T sera bornée par :

$$L_i / b_i \leq T \leq L_i / b_i - 1 \tag{3.31}$$

– si la durée de traitement p_i est égale à U_i, alors la valeur de T sera bornée par (voir Figure 3.10) :

$$U_i / b_i \leq T \leq U_i / b_i - 1 \tag{3.32}$$

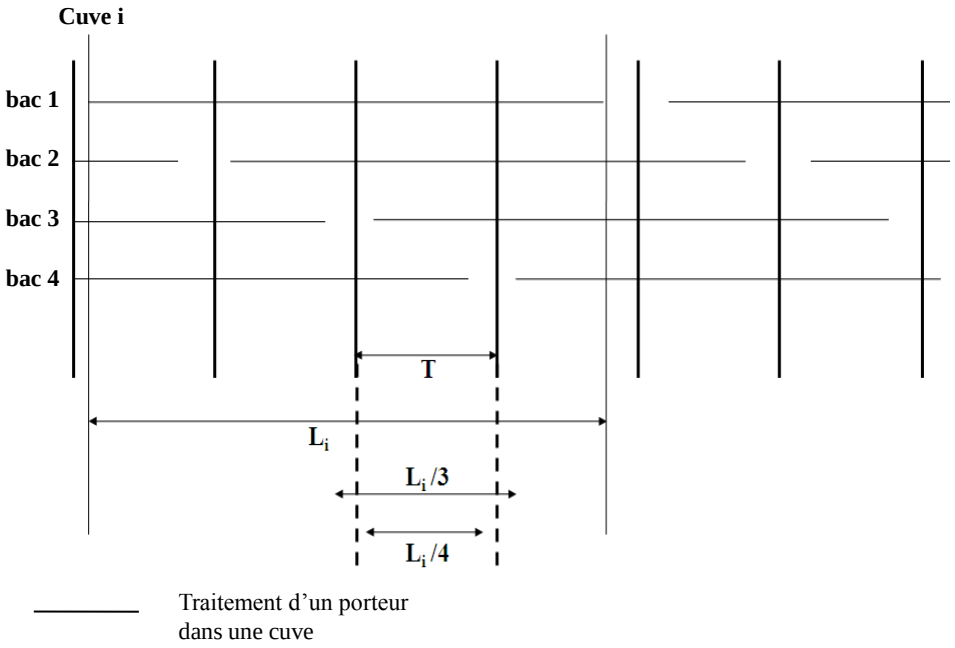

FIGURE 3.10 – Nouvelles bornes pour T

La Figure 3.10 schématise les nouvelles bornes pour T. Nous avons modifié, ensuite, les valeurs des bornes de traitement des cuves multi-bacs. Pour l'équation (3.29), nous avons remplacé la valeur de T par l'expression $L_i/b_i - 1$. Ainsi :

$$L_i^{'} = L_i - (b_i - 1) * (L_i/b_i - 1) = 0 \tag{3.33}$$

Pour l'équation (3.30), nous avons remplacé la valeur de T par l'expression U_i/b_i. Nous avons obtenu :

$$U_i^{'} = U_i - (b_i - 1) * (U_i/b_i) = U_i/b_i \tag{3.34}$$

Les deux valeurs définies par les deux équations (3.33) et (3.34) sont utilisées comme les temps de traitement minimaux et maximaux pour les cuves multi-bacs, pour la génération des différentes séquences de mouvements des robots comme montré à la **section 3.2.2**.

Au niveau du programme linéaire défini dans la **section 3.2.4**, sa modification tient essentiellement des contraintes qui garantissent le respect des bornes inférieures et supérieures des temps de traitement dans les cuves multi-bacs. Ainsi, partant des deux séquences de mouvements relatives aux robots 1 et 2, l'ordre d'exécution de ces mouvements sera connu à priori. De ce fait, soient les deux mouvements i et $Pos[i]$ tel que $Pos[i]$ est la cuve multi-bacs.

- si le mouvement i (M_i) est effectué avant le mouvement $Pos[i]$ $(M_{Pos[i]})$, comme illustré dans la Figure 3.9(a). Alors on définit les contraintes suivantes :

$$(b_i - 1)T + t_{Pos[i]} - t_i \geq L_{Pos[i]} + d_i \tag{3.35}$$

$$(b_i - 1)T + t_{Pos[i]} - t_i \leq U_{Pos[i]} + d_i \tag{3.36}$$

- si le mouvement i (M_i) est effectué après le mouvement $Pos[i]$ $(M_{Pos[i]})$ comme illustré dans la Figure 3.9(b), alors on définit les contraintes suivante :

$$b_i T + t_{Pos[i]} - t_i \geq L_{Pos[i]} + d_i \tag{3.37}$$

$$b_i T + t_{Pos[i]} - t_i \leq U_{Pos[i]} + d_i \tag{3.38}$$

Ainsi, pour les cuves multi-bacs les contraintes (3.7)-(3.11) seront remplacées par les contraintes (3.35)-(3.38).

Dans la suite, nous allons valider nos extensions sur trois problèmes industriels de lignes de traitement de surface contenant des cuves multi-bacs.

3.3 Résultats expérimentaux

Notre objectif était de pouvoir résoudre les problèmes **HSP** Cycliques avec deux robots et avec zone de chevauchement pour tout type de produits. En effet, les produits sont caractérisés par une gamme opératoire qui ne suit pas l'implan-

tation des cuves. Pour cela, nous avons proposé un algorithme qui permet de définir deux séquences de mouvements pour chaque robot.
Afin de valider cet algorithme, nous l'avons testé sur trois problèmes industriels qui ont été décrits par Manier (1994) et proposés dans la thèse de Shapiro (1985) et qui portent sur des lignes unidirectionnelles comportant des cuves mono-fonction. La particularité de ces exemples est qu'ils contiennent des cuves multi-bacs. Dans la suite, nous allons définir ces trois exemples industriels pris en considération. Ceux ci sont la ligne *Copper* (**section 3.3.1**), la ligne *Black Oxide* (**section 3.3.2**) et la ligne *Zinc* (**section 3.3.3**).

3.3.1 Ligne *Copper*

La Figure 3.11 schématise la ligne, ainsi que la gamme de traitement. Les fenêtres des temps de traitement du problème ainsi que les temps de déplacement du robot en charge et la matrice des temps de déplacement à vide sont présentés dans l'Annexe C.
Les postes de chargement et déchargement dans la ligne *Copper* sont associés.

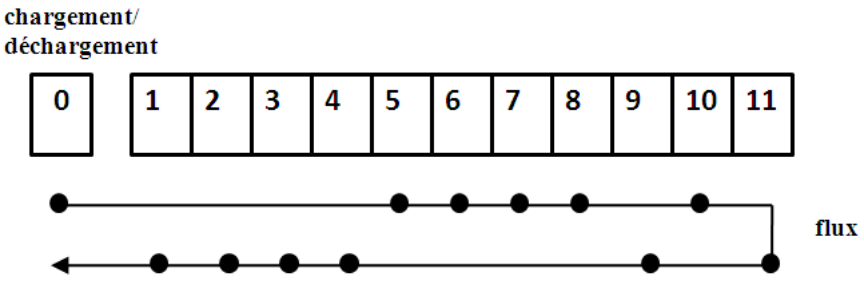

FIGURE 3.11 – Configuration physique et gamme opératoire de la ligne *Copper*

Nous avons résolu le problème de la ligne *Copper* pour deux robots. La solution obtenue est comparée à la solution fournie par Manier (1994).
Cette solution a été trouvée après 9 itérations. Le temps d'exécution est égal à 7 secondes. La durée du cycle obtenue est $T = 307,9$. Les différentes valeurs de

t_i seront présentées dans le Tableau 3.1 ainsi que la Figure 3.12. La description en détail de la solution trouvée est définie dans l'Annexe C.

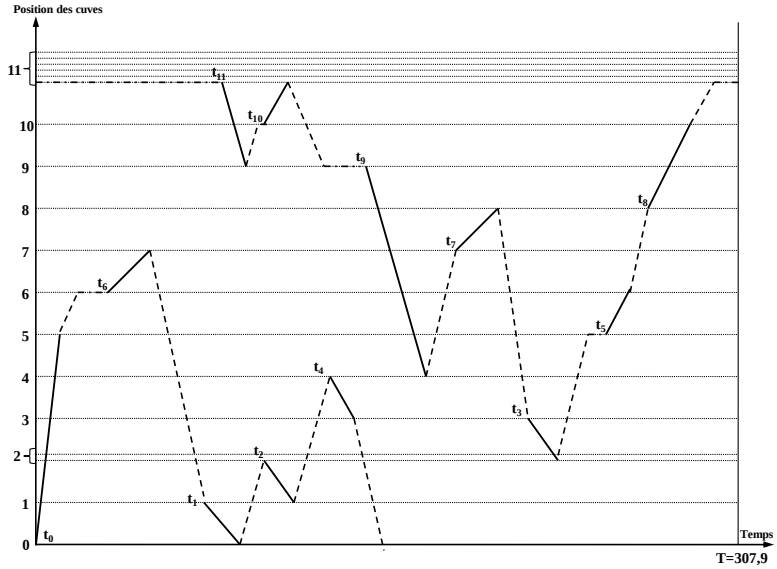

FIGURE 3.12 – Diagramme de Gantt pour l'exemple de la ligne *Copper*

Les diagrammes de Gantt présentés dans ce chapitre ne sont pas à l'échelle mai une présentation approximative, et le détail sur les différentes dates de retrait et de dépôt est défini dans l'Annexe C. Nous n'avons pas aussi représenté les bacs dans les diagrammes de Gantt, car nous avons supposé que la durée de déplacement entre ces bacs est égale à 0.

TABLE 3.1 – Résultat du problème *Copper*

Cuves (i)	0	1	2	3	4	5	6	7	8	9	10	11
t_i	0	$77,5$	$100,6$	$210,7$	$122,2$	$256,7$	$51,3$	$188,9$	$274,3$	$164,3$	$105,6$	$80,7$
Robot	R_1	R_1	R_1	R_2	R_1	R_2	R_1	R_2	R_2	R_2	R_2	R_2

Manier (1994) a résolu ce problème pour 1 et 2 robots, les durées T du cycle trouvées pour les deux cas sont identiques, et sont égales à 320. En effet, dans ce problème la ressource critique est la cuve numéro 11 qui est une cuve multi-bacs dans laquelle le temps de traitement minimal moyen est de 300 secondes (1800/6). Pour cette raison, l'écart entre ce temps et la durée du cycle est très faible. Le nombre de robots n'influe pas d'une manière considérable sur la durée

du cycle.

Malgré tout, nous avons pu améliorer la durée du cycle en diminuant la valeur de T de 12,1 secondes (3,8%).

3.3.2 Ligne *Black Oxide*

La Figure 3.13 schématise la ligne, ainsi que la gamme de traitement. Les fenêtres des temps de traitement du problème ainsi que les temps de déplacement du robot en charge et la matrice des temps de déplacement à vide sont présentés dans l'Annexe C.

Les postes de chargement et déchargement dans la ligne *Black Oxide* sont associés.

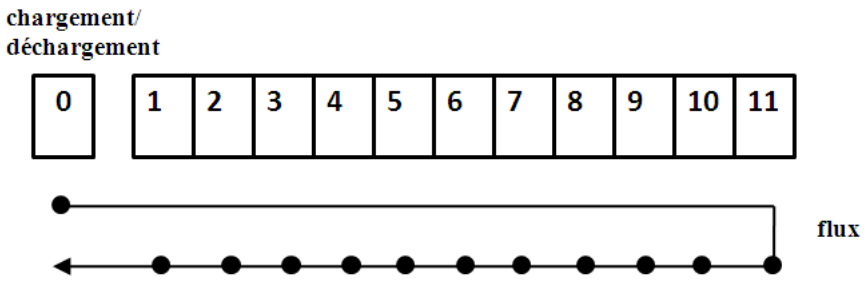

FIGURE 3.13 – Configuration physique et gamme opératoire de la ligne *Black Oxide*

Nous avons résolu le problème de la ligne *Black Oxide* pour deux robots. La durée du cycle obtenue est $T = 210,2$. Les différentes valeurs de t_i sont présentées dans le Tableau 3.2 et le diagramme de Gantt relatif à cette solution est présenté dans la Figure 3.14. La description détaillée de la solution trouvée est définie dans l'Annexe C.

Manier (1994) a résolu le problème *Black Oxide* pour 1 robot. La solution fournie est optimale avec $T = 268,9$. Ainsi, avec deux robots nous avons pu améliorer la valeur de T de 21,83%.

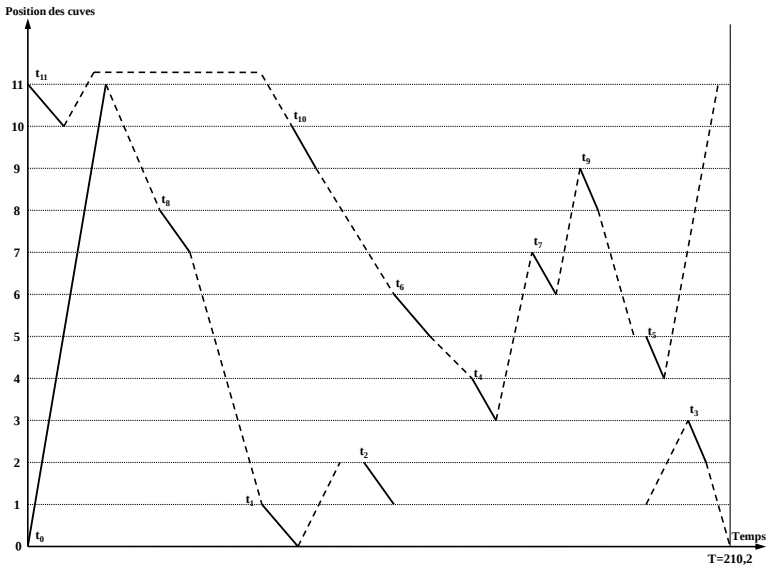

FIGURE 3.14 – Diagramme de Gantt pour l'exemple de la ligne *Black Oxide*

TABLE 3.2 – Résultat du problème *Black Oxide*

Cuves (i)	0	1	2	3	4	5	6	7	8	9	10	11
t_i	0	55, 5	85, 9	193, 4	114, 6	173, 1	94, 5	138, 2	29, 1	154, 3	78, 6	0
Robot	R_1	R_1	R_1	R_1	R_2	R_2	R_2	R_2	R_1	R_2	R_2	R_2

3.3.3 Ligne *Zinc*

La Figure 3.15 schématise la ligne, ainsi que la gamme opératoire des produits. Les fenêtres des temps de traitement du problème ainsi que les temps de déplacement du robot en charge et la matrice des temps de déplacement à vide sont présentés dans l'Annexe C.

Les postes de chargement et déchargement dans la ligne *Zinc* sont associés.

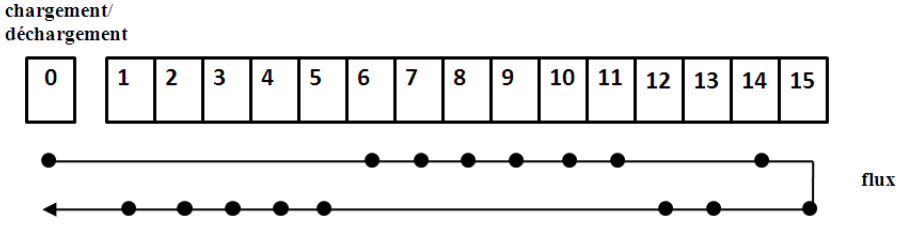

FIGURE 3.15 – Configuration physique et gamme opératoire de la ligne *Zinc*

Nous avons résolu le problème de la ligne *Zinc* pour deux robots. La solution obtenue est comparée à la solution fournie par Manier (1994).
La durée du cycle obtenue est $T = 435,85$. Cette solution est présentée dans le Tableau 3.3 et schématisée dans la Figure 3.16. La solution trouvée est détaillée dans l'annexe C. Cette solution a été trouvée après 3 itérations et le temps d'exécution est égal à 1,75 secondes. Comparée à la solution fournie par Manier (1994) où $T = 435,9$, nous remarquons que notre approche a réussi à améliorer la valeur de T de 0,05 secondes.

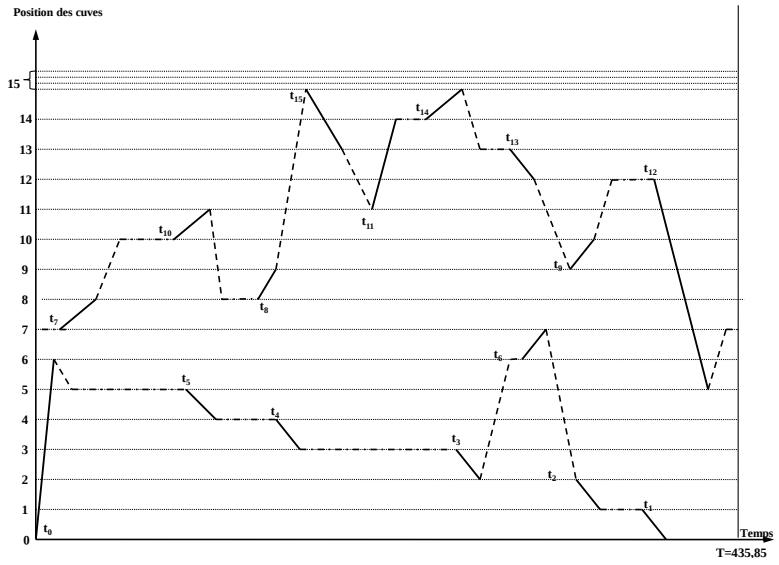

FIGURE 3.16 – Diagramme de Gantt pour l'exemple de la ligne *Zinc*

TABLE 3.3 – Résultat du problème *Zinc*

Cuves	0	1	2	3	4	5	6	7
Temps	0	322, 9	283, 8	223	164, 5	142, 2	256, 9	17, 35
Robot	R_1	R_1	R_1	R_1	R_1	R_1	R_1	R_2
Cuves	8	9	10	11	12	13	14	15
Temps	155, 45	286, 25	122, 25	198, 25	382,55	251,85	222,05	176,75
Robot	R_2	R_2	R_2	R_2	R_2	R_2	R_2	R_2

En conclusion, la méthode proposée a prouvé sa performance pour la résolu-

tion des problèmes industriels réels. En effet, l'algorithme d'affectation proposé permet de trouver une bonne affectation des mouvements aux deux robots, tout en minimisant le risque de collision et parfois tout en éliminant ce risque. En effet, pour les deux exemples ligne *Copper* et ligne *Zinc*, dans les solutions fournies par l'algorithme d'affectation, aucun risque de collision n'a été détecté. En outre, le modèle linéaire proposé permet de calculer les dates de début des opérations de déplacement des porteurs par les robots, tout en garantissant le respect des temps de traitement ainsi que les durées de déplacement des robots.

3.4 Conclusion

Dans ce chapitre, nous avons adapté notre algorithme pour la résolution des problèmes CHSP pour des lignes complexes avec deux robots et dans le cas où la gamme opératoire des produits à traiter est différente de l'implantation des cuves. Dans ce dernier cas, l'affectation des mouvements des robots en charge est plus difficile car ces mouvements ne s'effectuent plus entre cuves adjacentes, mais plutôt entre des cuves quelconques sur la ligne. Ceci restreint par conséquent l'espace du mouvement de l'autre robot.

Cet algorithme a été testés sur trois exemples pris de la littérature. Les résultats fournis sont satisfaisants et permettent de trouver un ordonnancement pour chacun des deux robots, tout en évitant les collisions.

Chapitre 4

Ordonnancement des lignes de traitement de surface dans le cadre d'un mode de fonctionnement mixte

Sommaire

4.1 Introduction

Le cas des productions mono-produit ou mono-gamme que nous avons considérés dans les chapitres précédents, ne constitue qu'une partie des problèmes HSP en général. D'autres problèmes peuvent surgir selon le type de production de l'entreprise, ou bien selon la conception des lignes, etc...

Varnier (1996) a défini trois types de production : les productions en grande série, en moyenne série et en petite série.

- pour les productions en grande série, les lignes sont généralement conçues pour le traitement d'un produit caractérisé par une gamme opératoire. Le calcul d'un ordonnancement cyclique est l'opération la plus importante car elle conditionne la rentabilité de l'entreprise ;
- les productions en moyenne série, où les lots de produits à traiter doivent être suffisamment grands pour permettre à la ligne de fonctionner en mode cyclique. Pour chaque lot, un ordonnancement est déjà calculé, le point critique dans ce mode de fonctionnement est le passage d'un lot à un autre, que l'auteur a appelé **phase transitoire**.

  Le travail de Varnier (1996) a porté sur ces deux types de productions.
- les productions en petite série où les lignes doivent être de grandes tailles (nombre de cuves important) et doit accueillir divers types de produits identifiés par des gammes opératoires. Elles peuvent recevoir des commandes de produits qui doivent être prises en compte lors de l'ordonnancement.

Par analogie aux différents types de HSP que nous avons définis dans la **section 1.3.2**, le premier type est le HSP cyclique, le deuxième type fait partie du HSP prédictif et le troisième type est le HSP dynamique.

Dans ce chapitre, nous allons nous intéresser à ce dernier type de production (production en petite série) ainsi qu'au premier type. Dans certaines lignes, l'entreprise peut traiter simultanément différents lots de produits qui peuvent être de types multi-produits ou mono-produit. Notre objectif est de chercher un

ordonnancement qui permet de passer d'une production multi-produits à une production mono-produit tout en minimisant la durée de traitement de ces deux lots. Connaissant à priori un ordonnancement pour chaque lot, l'objet de ce travail est de traiter simultanément ces deux lots sans pour autant vider la ligne entre leurs traitements et de trouver rapidement l'ordonnancement cyclique déjà connu.

La suite de ce chapitre se décomposera comme suit. Nous définissons, dans la **section 4.2**, le problème traité. Ensuite, nous présentons le problème DHSP dans la **section 4.3**. Dans cette section, nous présentons, dans un premier lieu, un état de l'art sur ce type de problème, ensuite nous proposons un PLM pour la résolution du problème DHSP. Dans la **section 4.4**, un algorithme est proposé pour la résolution du problème dans le cadre d'un fonctionnement mixte (DHSP-CHSP).

4.2 Définition du problème

Le travail que nous proposons consiste à résoudre un HSP, dans le cadre d'un fonctionnement mixte, pour lequel il est nécessaire de traiter dans une même installation des produits différents et des rafales de produits identiques. Dans ces conditions, la solution la plus intéressante pour les industriels est d'être capable d'alterner des modes de production dynamiques et cycliques.

L'objectif de cette partie de thèse est donc de proposer une méthode efficace permettant de résoudre le problème d'ordonnancement associé à la phase transitoire relative à ce type de fonctionnement. La principale difficulté identifiée consiste ici à passer du mode dynamique au mode cyclique, c'est-à-dire à rejoindre un cycle à partir d'une solution courante donnée, en supposant que ce cycle est connu à priori. Nous souhaitons donc passer d'une gestion dynamique de la ligne (dans le cas d'une fabrication de type multi-produits) à un pilotage

automatisé sous forme de séquence répétitive de mouvements des robots (cas statique, avec un ordonnancement correspondant à un cycle). L'objectif est de minimiser la durée du régime transitoire (Figure 4.1)

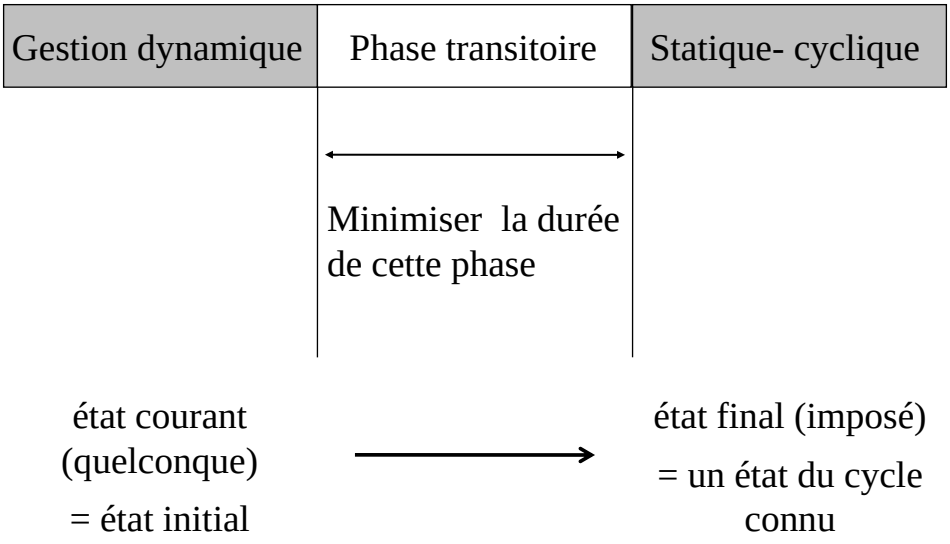

FIGURE 4.1 – Problématique de la phase transitoire

A notre connaissance, aucun résultat académique n'existe pour ce type de HSP. Par contre, Varnier (1996) a étudié le problème de la minimisation de la phase transitoire entre un mode de fonctionnement cyclique pour un premier lot et un mode de fonctionnement cyclique pour le deuxième lot. Il a proposé un modèle qui comporte trois types de contraintes. Le premier type de contraintes porte, d'une part, sur la gamme opératoire des produits qui doit être respectée et, d'autre part, la durée opératoire de chaque produit qui doit être bornée par une valeur minimale et une valeur maximale connues. Le second type de contraintes est lié à la transition qui impose que l'ordonnancement de la transition soit cohérent avec le dernier cycle optimal pour le premier lot et avec le premier cycle optimal pour le deuxième lot. Enfin, le troisième type de contraintes dites

disjonctives sur les ressources de la ligne. Il s'agit de contraintes qui tiennent compte de la disponibilité des cuves ainsi que des robots. Dans le cas multi-robots, il s'agit des contraintes de non collision. Ce modèle a été implémenté à l'aide du langage de PLC (Prolog III). Au début de la résolution, toutes les contraintes déterministes (les deux premiers types de contraintes) sont posées et constituent le système courant. Ensuite, les contraintes disjonctives sont arbitrées par une procédure semblable à une procédure de type "séparation et évaluation".

Pour le HSP en mode de fonctionnement mixte, la question principale qui se pose est : pouvons-nous, dans tous les cas, atteindre un état donné du système en termes d'occupation des cuves, robots et temps opératoires écoulés dans les cuves à partir d'un état courant quelconque ?
Évidemment la réponse est affirmative puisque dans la situation la plus défavorable, il suffit d'attendre que la ligne se vide avant de commencer une production cyclique mono-produit. Mais cela ne garantit pas les performances en termes de minimisation du temps de régime transitoire entre les deux modes de fonctionnement. La réponse à la question peut alors être fournie sous forme de méthode efficace d'ordonnancement des opérations pour cette phase transitoire. Toutefois, il serait intéressant d'étudier également les propriétés de l'espace des solutions (notamment celui des solutions cycliques mono-produit), pour extraire éventuellement des conditions de passage d'un état du système à l'autre (d'une solution à une autre).

4.3 Présentation des problèmes HSP dynamiques

Lorsque le nombre des types de produits à traiter est assez important, à la différence du cas mono-produit, les séquences des mouvements ne sont pas répétitives et le calcul d'un ordonnancement est réalisé chaque fois qu'un porteur entre dans la ligne.

L'objectif primordial pour les problèmes HSP Dynamique est la maximisation de la productivité tout en assurant la qualité des produits.

Généralement, le problème du traitement de surface dynamique procède de la façon suivante : à l'instant t_0 un porteur entre dans la ligne. A cet instant, l'état de la ligne, la localisation ainsi que les temps de traitement écoulés des temps de traitement pour les porteurs précédents, sont déterminés. Ainsi, il faut insérer le nouveau porteur dans la ligne, le plus tôt possible, tout en tenant compte de l'état de la ligne et tout en évitant les deux conflits suivants identifiés par (Lamothe, 1996) :

- conflit de cuves : ce conflit est constaté lorsque un produit termine son traitement dans une cuve quelconque et il est prêt à être transporté à la prochaine cuve correspondante à son prochain traitement. Mais, cette dernière n'est pas vide et contient déjà un autre produit (porteur) qui est en cours de traitement. Et puisque chaque cuve ne peut contenir qu'un seul produit à la fois, cette situation peut créer une infaisabilité au niveau de l'ordonnancement.
- conflit de robot : ce conflit est détecté lorsque un produit est prêt à être transporté par le robot mais ce dernier n'est pas disponible et il est en train de transporter un autre produit. De plus, un autre conflit de robot peut se produire lorsque le robot n'a pas le temps nécessaire pour venir à vide à un produit qui a terminé son traitement à temps.

Ces deux situations de conflits sont illustrées dans la Figure 4.2 ci-après.

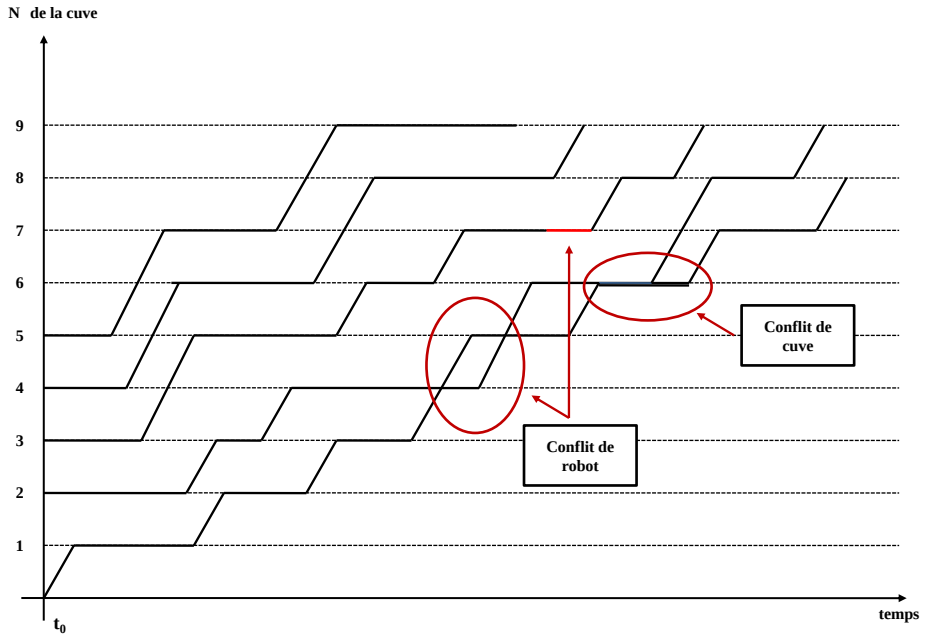

FIGURE 4.2 – Conflit de cuve et conflit de robot (Lamothe, 1996)

Un aperçu sur les différentes méthodes de résolution qui ont été proposées dans la littérature est présenté dans la section suivante.

4.3.1 Etat de l'art sur le Hoist Scheduling Problem Dynamique (DHSP)

L'approche de résolution la plus simple consiste à faire introduire un porteur dans la ligne le plus tôt possible. Si l'évaluation de cette solution montre qu'un conflit se produit, la date d'entrée du porteur est reculée et les dates prévues de transport des porteurs déjà en cours de traitement ne seront pas modifiées. Ce principe de non remise en cause des décisions précédentes est utilisé dans l'heuristique Earliest Starting Time (EST) proposée par Song *et al.* (1993). Aussi, Yin et Yih (1992) ont proposé, dans un premier temps, une heuristique simple qui permet de minimiser la durée d'exécution de toutes les tâches. Cette heuristique omet les tolérances sur les durées de trempe.

La durée de traitement utilisée dans chaque cuve correspond à la borne inférieure des temps de traitement. Après détermination des dates de début des mouvement du transport pour chaque porteur qui entre dans le système, l'existence de conflits est cherchée (conflits de cuves et conflits de robots). Une fois un conflit détecté, la date d'entrée du nouveau porteur est retardée. Ensuite, après chaque décalage, l'algorithme recherche s'il existe de nouveaux conflits. D'autres auteurs ont proposé des heuristiques de résolution pour ce type de problème qui utilisent principalement les marges sur les durées de traitement .

Yin et Yih (1992) ont proposé, dans un second temps, une heuristique qui utilise les tolérances sur les durées de traitement du porteur j lors de la détection d'un conflit de robot. Cette procédure teste si la tolérance au niveau d'une opération k (différence entre borne supérieure et borne inférieure des temps de traitement) est suffisante pour éviter ce conflit, sinon la date d'entrée du porteur j est retardée. Cette heuristique n'utilise les tolérances que sur le nouveau porteur et sur l'opération de traitement où un conflit est détecté. Ainsi, les dates de transport des porteurs déjà existants sur la ligne ne seront pas modifiées.

Yih (1994) a proposé dans une première étape un algorithme appelé algorithme de base et qui consiste à introduire un porteur le plus tôt possible dans la ligne sans que cela ne provoque de conflits. Les temps de trempe utilisés sont les durées de traitement minimales. Ensuite, l'auteur a proposé une procédure itérative qui a pour but de faire entrer dans la ligne un nouveau porteur le plus tôt possible. Cette heuristique est construite sur la base du concept de la tolérance des temps de traitement (la différence entre le maximum et le minimum des temps de traitement). Ce concept fournit de la flexibilité pour l'ordonnancement des mouvements du robot, ce qui améliore la productivité du système.

Lorsqu'un produit j entre dans le système, l'algorithme débute avec une date

d'entrée initiale (ET_j) égale à sa date d'arrivée (AT_j). A chaque itération la date d'entrée est mise à jour jusqu'à l'obtention d'un ordonnancement réalisable. Deux phases sont nécessaires pour la mise à jour des dates d'entrée : une première phase qui traite le conflit des cuves, la deuxième traite le conflit de robot. Après la première phase, la date d'entrée garantit le non conflit de cuves pour l'ordonnancement courant. La deuxième phase détecte tous les conflits des robots éventuels et résout ce problème par, soit une mise à jour de la date d'entrée, soit par, l'utilisation du concept de la tolérance. Si la tolérance permet de résoudre le conflit, le temps de traitement dans la cuve courante sera prolongé et les dates de début de tous les processus associés à cette tâche seront retardées par la même durée. D'autre part, si la tolérance ne permet pas la résolution de ce conflit, la date d'entrée de ce job sera retardée pour empêcher ce conflit. Cette procédure se répète jusqu'à ce qu'aucun changement ne marque cet ordonnancement.

Fargier et Lamothe (2001) ont proposé une formulation générale pour le DHSP sous la forme d'un programme linéaire. Ensuite, ils ont appliqué la logique floue comme approche de résolution pour rechercher un ordonnancement des opérations de trempe au niveau de chaque cuve. Cet ordonnancement est remis en cause chaque fois qu'un nouveau produit entre dans la ligne. Le problème traité suppose que les temps de traitement dans les cuves peuvent être choisis au sein des intervalles flous. Le degré de satisfaction d'un intervalle flou se réfère à une évaluation de la qualité du traitement. Ainsi, le degré de satisfaction (qualité) d'un porteur est évalué comme le plus faible degré de satisfaction de ses opérations de trempe, c'est-à-dire le plus mauvais traitement qu'il a subi. Les auteurs ont proposé une approche d'aide à la décision qui converge rapidement vers des compromis entre les deux critères : la qualité et la productivité, plutôt que d'une agrégation de ces derniers, sous la forme d'un seul critère. Finalement, c'est au décideur de prendre la décision.

Hindi et Fleszar (2004) ont proposé un algorithme qui utilise un modèle non standard de satisfaction de contraintes où ils utilisent des variables d'ordres correspondant aux opérations de déplacement du robot. Le processus de planification est réalisé en affectant des étiquettes aux dates de début de déplacement du produit j de la cuve i ($S^H_{i,j}$). Au cours de chaque étape, un ensemble des opérations du robot est disponible, et une seule de ces opérations est choisie selon le critère de leur plus petite date de début. Une fois l'opération à ordonnancer est sélectionnée, les dates de début et de fin des opérations de transport (robot) et les dates de début et de fin des opérations de trempe (cuves) sont calculées. Notons que, nous désignons par "opération du robot", l'opération de déplacement du porteur ou du produit d'une cuve à une autre. Ainsi, tout au long du processus d'ordonnancement, tous les conflits des robots et des cuves sont évités. Mais, l'infaisabilité de l'ordonnancement peut se produire dans le cas où il y a violation de la borne supérieure des temps de traitement. Formellement, ce cas est identifié lorsque la date de retrait d'un porteur d'une cuve ($E^T_{j,i}$) est supérieure à la date de début de trempe de ce porteur dans cette cuve plus la borne maximale de la durée de traitement ($S^T_{j,i} + L_{j_i}$). Dans ce cas, un retour en arrière (backtracking) est initialisé afin d'éliminer cette violation. L'algorithme prend fin lorsque toutes les opérations sont ordonnancées. Les auteurs ont comparé leur algorithme à celui de Yih (1994) en faisant les expérimentations sur les instances que cette dernière a généré. Cet algorithme a réalisé de meilleures solutions que celles réalisées par l'algorithme de Yih (1994). Cette évaluation a été effectuée en calculant le pourcentage d'amélioration par rapport au makespan trouvé par l'algorithme de base proposé par Yih (1994).

Paul *et al.* (2007) ont proposé une heuristique qu'ils l'ont nommé "Adaptive Time Window" pour l'ordonnancement multi-produits dans les lignes de production. Cette heuristique prend avantage de la flexibilité des temps de

traitement. Les auteurs ont défini pour chaque opération de trempe ($O^T_{i,j}$) deux intervalles de temps. Le premier intervalle est borné par la date de début au plus tôt et la date de début au plus tard de l'opération ($O^T_{i,j}$). Le deuxième intervalle est borné par la date de fin au plus tôt et la date de fin au plus tard de l'opération ($O^T_{i,j}$). Ces intervalles de temps sont actualisés en continu, tout au long de l'heuristique. Cette heuristique débute par l'ordonnancement de la première tâche en définissant les différents intervalles pour les opérations de trempe dans chaque cuve. Ensuite, une fois qu'une nouvelle tâche est introduite dans la ligne, les dates de début des opérations sont définies en tenant compte des dates de début et de fin au plus tôt et les dates de début et de fin au plus tard des opérations déjà planifiées. Une fois que la date de début de l'opération courante est déterminée sans causer aucun conflit, les intervalles de temps relatifs seront initialisés. Si un conflit est déterminé alors un retour en arrière est appliqué. Cette heuristique a été testée sur un problème industriel. Ensuite, des instances ont été générées en se basant sur les données du cas industriel mais en faisant varier les intervalles des temps de traitement.

4.3.2 Modélisation du problème DHSP

Le modèle que nous présentons est défini pour le DHSP mono-robot. Nous définissons dans la suite de nouvelles notations.
Les données du problèmes sont :

n le nombre de cuves de traitement ; la cuve de chargement est assimilée à une cuve fictive appelée cuve 0, et la cuve de déchargement à la cuve $n+1$, fictive aussi ;

J le nombre de produits à traiter dans le lot multi-produits ;

$d_{i,j}$ le temps de déplacement du robot en charge du produit j entre les cuves i et $i+1$ de la ligne. Ce temps peut comprendre les temps de levée et descente du porteur par le robot et le temps d'égouttage s'il y en a $(i = 0, ..., n; \quad j = 1, ..., J)$;

$c_{u,v}$ le temps de déplacement du robot à vide entre deux cuves quelconques u et v, $(u, v = 0, ..., n+1)$;

$L_{i,j}$ la durée minimale de trempe du produit j dans la cuve i $(i = 1, ..., n; \quad j = 1, ..., J)$;

$U_{i,j}$ la durée maximale de trempe du produit j dans la cuve i $(i = 1, ..., n; \quad j = 1, ..., J)$;

$O_{i,j}$ désigne l'opération de retrait du produit j de la cuve i $(i = 0, ..., n; \quad j = 1, ..., J)$.

M un entier très grand.

Les variables du problème sont :

C_{max} la durée d'achèvement du traitement du dernier produit ;

$t_{i,j}$ la date de retrait d'un produit j de la cuve i $(i = 0, ..., n; \quad j = 1, ..., J)$;

$y_{ij,kl}$ une variable binaire définie par :

$$y_{ij,kl} = \begin{cases} 1 & \text{si l'opération } O_{i,j} \text{ est effectuée après l'opération } O_{k,l}, \text{ pour} \\ & i, k = 0, ..., n \,;\, j, l = 1, ..., J \\ 0 & \text{sinon} \end{cases}$$

Le programme linéaire pour le cas mono-robot est défini par le programme

(P_1) présenté ci-dessous par les équations 4.1-4.11.

$$(P_1) \qquad \text{Minimiser} \quad C_{max} \tag{4.1}$$

Sous contraintes :

$$t_{0,1} = 0 \tag{4.2}$$

$$t_{i+1,j} - t_{i,j} \leq U_{i+1,j} + d_{i,j}, \quad i = 0, ..., n-1; \quad j = 1, ..., J \tag{4.3}$$

$$t_{i+1,j} - t_{i,j} \geq L_{i+1,j} + d_{i,j}, \quad i = 0, ..., n-1; \quad j = 1, ..., J \tag{4.4}$$

$$t_{i,j} - t_{i-1,l} \leq d_{i-1,l} + y_{ij,il}.M, \quad i = 1, ..., n; \quad j, l = 1, ..., J \tag{4.5}$$

$$t_{i,l} - t_{i-1,j} \leq d_{i-1,j} + (1 - y_{ij,il}).M, \quad i = 1, ..., n; \quad j, l = 1, ..., J \tag{4.6}$$

$$t_{k,l} - t_{i,j} \geq d_{i,j} + c_{i+1j,kl} - y_{ij,kl}.M, \quad i, k = 0, ..., n; \quad j, l = 1, ..., J \tag{4.7}$$

$$t_{i,j} - t_{k,l} \geq d_{k,l} + c_{k+1l,ij} - (1 - y_{ij,kl}).M, \quad i, k = 0, ..., n; \quad j, l = 1, ..., J \tag{4.8}$$

$$C_{max} - t_{n,j} \geq d_{n,j}, \quad j = 1, ..., J \tag{4.9}$$

$$t_{i,j} \geq 0, \quad i = 0, ..., n; \quad j = 1, ..., J \tag{4.10}$$

$$y_{ij,kl} \in \{0, 1\}, i, k = 1, ..., n; \quad j, l = 1, ..., J \quad \text{et} \quad (i, j) \neq (k, l) \tag{4.11}$$

Interprétation des contraintes :

Le critère à optimiser (C_{max}) dans l'équation (4.1) est la durée d'exécution de tous les produits .

La contrainte (4.2) définit la date de début de déplacement du produit 1 de la cuve 0 (le début de l'opération de chargement d'un porteur ou produit).

Les contraintes (4.3) et (4.4) assurent que les durées de traitement de chaque produit j dans chaque cuve i sont bornées par une durée de traitement minimale ($L_{i,j}$) et une durée de traitement maximale ($U_{i,j}$).

Les contraintes (4.5) et (4.6) garantissent qu'une cuve ne peut pas recevoir simultanément plus de produits que de sa capacité. Dans ce programme, la capacité

des cuves est unitaire. Considérons deux opérations $O_{i,j}$ et $O_{i,l}$ utilisant la même cuve i :

- si l'opération $O_{i,j}$ est effectuée avant l'opération $O_{i,l}$ alors $y_{ij,il} = 0$ et la date de retrait $(t_{i,j})$ du produit j de la cuve i doit être inférieure à la date de dépôt $(t_{i-1,l} + d_{i-1,l})$ du produit l dans la cuve i. Ceci est vérifié par ces contraintes qui deviennent :

$$t_{i,j} - t_{i-1,l} \leq d_{i-1,l}$$

$$t_{i,l} - t_{i-1,j} \leq d_{i-1,j} + M$$

- si l'opération $O_{i,j}$ est effectuée après l'opération $O_{i,l}$ alors $y_{ij,il} = 1$ et la date de retrait $(t_{i,l})$ du produit l de la cuve i doit être inférieure à la date de dépôt $(t_{i-1,j} + d_{i-1,j})$ du produit j dans la cuve i. Ceci est vérifié par ces contraintes qui deviennent :

$$t_{i,j} - t_{i-1,l} \leq d_{i-1,l} + M$$

$$t_{i,l} - t_{i-1,j} \leq d_{i-1,j}$$

Les contraintes (4.7) et (4.8) garantissent la disponibilité du robot pour l'exécution des déplacements des produits. Ainsi, le robot doit avoir suffisamment de temps pour se déplacer à vide entre deux opérations de transport.
Les contraintes (4.9) permettent de définir C_{max}. En effet, la durée d'exécution de tous les produits doit être supérieure à la durée d'exécution de chaque produit.

Le programme (P_1) est défini pour les problèmes DHSP mono-robot. Nous ajustons ce programme pour le cas multi-robots et plus précisément pour les problèmes de ligne avec deux robots. Ainsi, les contraintes (4.7) et (4.8) seront remplacées par les contraintes (4.12) et (4.13) qui permettent d'affecter les différents mouvements à l'un des deux robots, d'assurer la disponibilité

de chaque robot pour effectuer deux déplacements et éliminer tout risque de collision entre les deux robots.

Nous ajoutons une autre variable binaire définie par :

$$Z^r_{i,j} = \begin{cases} 1 & \text{si l'opération } O_{i,j} \text{ est effectuée par le robot } r, \text{ pour} \\ & i = 0, ..., n,\ j = 1, ..., J \text{ et } r = 1, 2 \\ 0 & \text{sinon} \end{cases}$$

$$t_{i,j} + d_{i,j} + c_{i+1j,kl} - t_{k,l} \leq M.(2 + y_{ij,kl} - Z^r_{i,j} - \sum_{h=r}^{2} Z^h_{k,l})$$
$$i, k = 0, ..., n; j, l = 1, ..., J, k < i, r = 1, 2 \qquad (4.12)$$

$$t_{k,l} + d_{k,l} + c_{k+1l,ij} - t_{i,j} \leq M.(3 - y_{ij,kl} - Z^r_{i,j} - \sum_{h=r}^{2} Z^h_{k,l})$$
$$i, k = 0, ..., n; j, l = 1, ..., J, k < i, r = 1, 2 \qquad (4.13)$$

Ces contraintes permettent d'affecter les mouvements aux deux robots et de gérer la non collision. Rappelons que R_1 se trouve à gauche de R_2.

1. Supposons que le robot R_1 est affecté à l'opération $O_{i,j}$ et le robot R_2 est affecté à l'opération $O_{k,l}$ sachant que $k < i$. Alors nous aurions $Z^1_{i,j} = 1$ et $Z^2_{k,l} = 1$.
   - si l'opération $O_{i,j}$ est effectuée après l'opération $O_{k,l}$ alors $y_{ij,kl} = 1$ et les contraintes 4.12 et 4.13 deviennent :

     pour $r = 1$, on aura :

$$t_{i,j} + d_{i,j} + c_{i+1j,kl} - t_{k,l} \leq M \qquad (4.14)$$

$$t_{k,l} + d_{k,l} + c_{k+1l,ij} - t_{i,j} \leq 0 \qquad (4.15)$$

pour $r = 2$, on aura :

$$t_{i,j} + d_{i,j} + c_{i+1j,kl} - t_{k,l} \leq 2M \tag{4.16}$$

$$t_{k,l} + d_{k,l} + c_{k+1l,ij} - t_{i,j} \leq M \tag{4.17}$$

L'équation (4.15) garantit que le robot R_1 arrive à la cuve i pour effectuer l'opération $O_{i,j}$ après la date à laquelle le robot R_2 arrive vide à la cuve i une fois l'opération $O_{k,l}$ est effectuée. Cette situation est illustrée par la Figure 4.3. Les autres équations sont satisfaites.

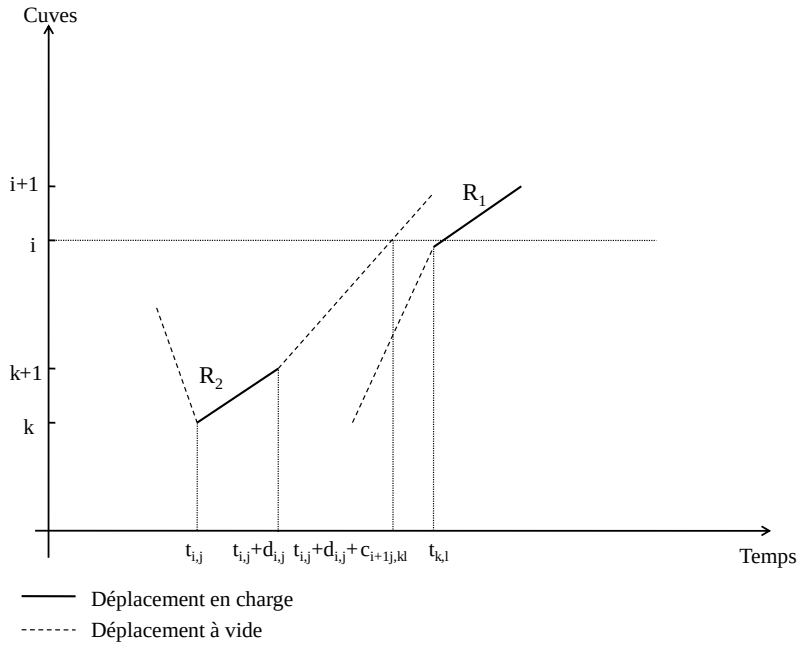

FIGURE 4.3 – Diagramme de Gantt pour les contraintes 4.13

– si l'opération $O_{i,j}$ est effectuée avant l'opération $O_{k,l}$ alors $y_{ij,kl} = 0$ et les contraintes 4.12 et 4.13 deviennent : pour $r = 1$, on aura :

$$t_{i,j} + d_{i,j} + c_{i+1j,kl} - t_{k,l} \leq 0 \tag{4.18}$$

$$t_{k,l} + d_{k,l} + c_{k+1l,ij} - t_{i,j} \leq M \tag{4.19}$$

pour $r = 2$, on aura :

$$t_{i,j} + d_{i,j} + c_{i+1j,kl} - t_{k,l} \leq M \tag{4.20}$$

$$t_{k,l} + d_{k,l} + c_{k+1l,ij} - t_{i,j} \leq 2M \tag{4.21}$$

Dans ce cas, et d'après l'équation (4.18) le robot R_2 doit arriver à la cuve k pour effectuer l'opération $O_{k,l}$ après la date à laquelle le robot R_1 arrive vide à la cuve k après avoir effectué l'opération $O_{i,j}$. Cette situation est illustrée par la Figure 4.4.

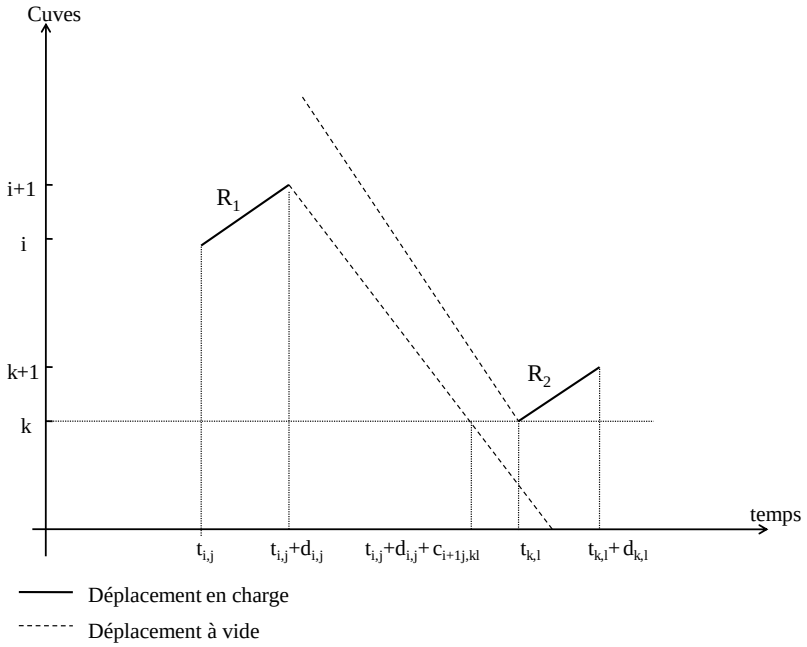

FIGURE 4.4 – Diagramme de Gantt pour les contraintes 4.12

2. Supposons que le robot R_1 (R_2) est affecté aux deux opérations $O_{i,j}$ et $O_{k,l}$ sachant que $k < i$. Alors nous aurions $Z^1_{i,j} = 1$ et $Z^1_{k,l} = 1$.
   – si l'opération $O_{i,j}$ est effectuée après l'opération $O_{k,l}$ alors $y_{ij,kl} = 1$ et les contraintes deviennent :

$$t_{i,j} + d_{i,j} + c_{i+1j,kl} - t_{k,l} \leq M$$

$$t_{k,l} + d_{k,l} + c_{k+1l,ij} - t_{i,j} \leq 0$$

Ces contraintes sont équivalentes aux contraintes (4.7) et (4.8) relatives à la disponibilité du robot dans le problème mono-robot.

– si l'opération $O_{i,j}$ est effectuée avant l'opération $O_{k,l}$ alors $y_{ij,kl} = 0$ et les contraintes deviennent :

$$t_{i,j} + d_{i,j} + c_{i+1j,kl} - t_{k,l} \leq 0$$

$$t_{k,l} + d_{k,l} + c_{k+1l,ij} - t_{i,j} \leq M$$

Chaque opération de déplacement doit être affectée à un seul robot. Ainsi, nous ajoutons les contraintes (4.22) qui assurent que si, par exemple, une opération est affectée au robot 1 alors $Z_{i,j,1} = 1$ et $Z_{i,j,2}$ doit être égale à 0.

$$\sum_{r=1}^{2} z_{i,j,r} = 1 \qquad \text{pour tout} \quad i = 0, ..., n; \quad j = 1, ..., J \tag{4.22}$$

Ainsi, nous définissons le programme (P_2) sous les contraintes (4.2) à (4.6), (4.9) à (4.11), (4.12), (4.13) et (4.22) pour la résolution du DHSP avec deux robots.

4.3.3 Exemple illustratif

Pour illustrer le modèle (P_2) que nous avons élaboré, considérons un exemple relatif à trois produits sur une ligne de cinq cuves. Les durées de déplacement du robot en charge et à vide sont égales à 4 et 2 secondes, respectivement. Les fenêtres du temps des durées de traitement sont représentés par le Tableau 4.1.

TABLE 4.1 – Données pour l'exemple illustratif

Type de produit	cuve i				
	1	2	3	4	5
1	[44,215]	[27,121]	[41,199]	[31,148]	[62,299]
2	[25,119]	[64,318]	[76,311]	[24,111]	[46,196]
3	[30,131]	[73,350]	[75,363]	[31,154]	[41,189]

En appliquant le modèle (P_2) à cet exemple, nous obtenons une solution telle que :

- la durée du makespan est égale à $C_{max} = 384$,
- les dates de début des opérations de transport sont données par le Tableau 4.2 :

TABLE 4.2 – Dates de début des opérations de transport pour l'exemple illustratif

Type de produit	cuve i					
	0	1	2	3	4	5
1	0	48	79	124	159	234
2	45	76	144	224	252	302
3	84	144	221	300	335	380

Cette solution est représentée par le diagramme de Gantt de la Figure 4.5. Cette figure permet de schématiser l'affectation des différentes opérations aux deux robots. Nous remarquons également l'absence de collision entre ces deux robots. En effet, nous remarquons que le robot R_2 est resté immobile (en attente) vide au dessus des cuves 5 et 6 afin d'éviter toutes collisions avec R_1 lorsque ce dernier effectue les déplacement à vide, respectivement, de la cuve 4 à la cuve 1 et de la cuve 5 à la cuve 2.

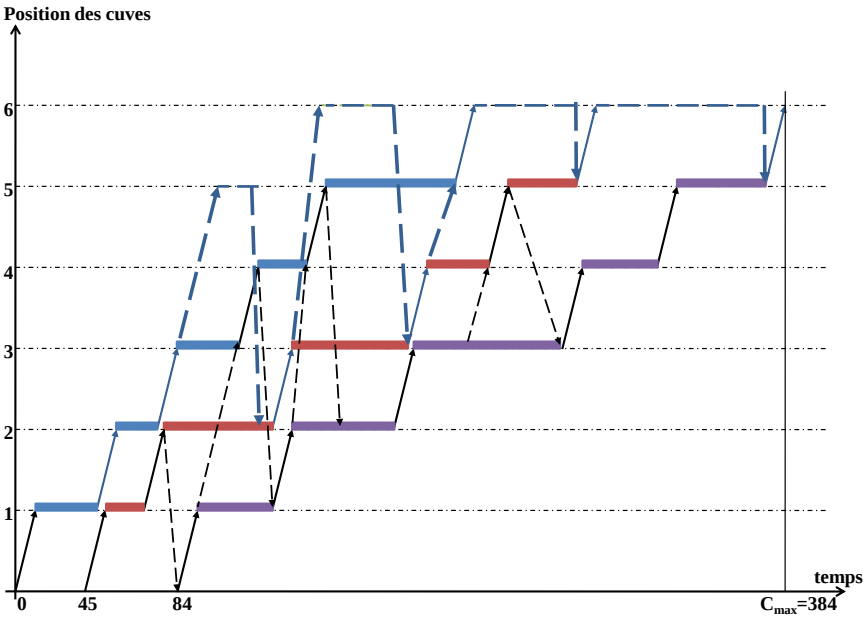

FIGURE 4.5 – Diagramme de Gantt pour la solution de l'exemple illustratif

4.4 Méthode de résolution pour le problème du HSP dans un fonctionnement mixte

Nous considérons le cas où l'entreprise de traitement de surface doit traiter simultanément les deux lots L_1 et L_2. Le lot L_1 est multi-produits, dont chacun des produits se caractérise par des fenêtres de temps différentes et une gamme opératoire qui suit l'ordre d'implantation des cuves, alors que le lot L_2 est mono-produit. Dans cette section, nous allons proposer un algorithme qui permet de déterminer les dates d'entrée au plus tôt ainsi que les différentes dates de déplacement des différents produits appartenant au lot L_2, tout en éliminant les conflits (conflits de cuves et conflits de robots).

4.4.1 Hypothèses et notations

Hypothèses :

H_1 : nous définissons la date de début de la transition comme la date d'entrée du dernier produit du lot L_1.

H_2 : la fin de la transition est la date d'entrée du produit P tel que P est le numéro du produit qui entre immédiatement après la sortie du dernier produit du lot L_1.

Notations :

J_1 le nombre de produits à traiter appartenant au lot L_1 ;

J_2 le nombre de produits à traiter appartenant au lot L_2 ;

$t_{i,j}^l$ la date de déplacement du produit j appartenant au lot l ($l = 1$ pour L_1 et $l = 2$ pour L_2) entre les cuves i et $i+1$ ($i = 0, ..., n; \quad j = 1, ..., J$) ;

S_{dhsp} la solution représentant les dates $t_{i,j}^1$ pour le lot L_1 déterminée par le programme (P_2) ;

S_{chsp} la solution cyclique représentant les dates $t_{i,j}^2$ pour le lot L_2 déterminée par l'algorithme proposé dans le deuxième chapitre ;

Sol_1 la solution du problème qui consiste à la détermination des différentes dates $t_{i,j}^2$ après vérification de la disponibilité des cuves ;

Sol_2 la solution du problème qui consiste à la détermination des différentes dates $t_{i,j}^2$ après vérification de la disponibilité du robot R_1 ;

Sol_f la solution du problème qui consiste à la détermination des différentes dates $t_{i,j}^2$ après vérification de la disponibilité du robot R_2 ;

Lors de la détermination de chaque solution (Sol_1, Sol_2 et Sol_f), il n'y a pas de remise en cause des valeurs de $t_{i,j}^1$.

4.4.2 Algorithme proposé

Le principe général de l'algorithme développé consiste à chercher un ordonnancement au niveau de la phase transitoire, où des produits appartenant aux

deux lots sont en traitement, tout en imposant dès que possible le cycle déjà connu pour le lot L_2. Partant d'une solution optimale pour le lot L_1 déterminée par la résolution du programme (P_2) et d'une solution pour le lot L_2 déterminée par l'algorithme proposé dans le deuxième chapitre (**section** 2.3.2), les différentes étapes de cet algorithme sont :

1. un état de la ligne au début de la phase transitoire est déterminé par la procédure **Etat-ligne** en localisant les différents produits de L_1 qui sont en cours de traitement ;
2. selon l'occupation des cuves par les différents produits de L_1, des intervalles de temps définissant les temps d'occupation des cuves au cours de la phase transitoire sont déterminés par la procédure **Calcul-dates** ;
3. en se basant sur ces intervalles, les dates d'entrée des produits du lot L_2 seront calculées en éliminant tout conflit de cuves (procédure **Test-cuve**) ;
4. ensuite, une procédure **Test-robot1** est appliquée pour tester s'il existe des conflits de robots pour R_1. Dans le cas où un conflit est détecté, les dates d'entrée des produits de L_2 seront recalculées tout en gérant la disponibilité des cuves et tout en imposant comme contrainte que deux produits de L_2 sont toujours séparés en entrée par T secondes ;
5. finalement, une procédure **Test-robot2** est appliquée pour tester la disponibilité du robot R_2 et calculer une solution réalisable.

Notre objectif est de déterminer la date d'entrée au plus tôt du premier produit appartenant au lot L_2 ($t^2_{0,1}$). Une fois cette date définie, les différentes valeurs de $t^2_{i,j}$ seront déterminées en se basant sur la solution trouvée, initialement, pour le lot L_2 en appliquant la relation (4.23) présentée ci-dessous :

$$t^2_{i,j} = t^2_{0,1} + t_i + kT; \qquad i = 0, ..., n, \quad j = 1, ..., J_2, \quad k \in \aleph \tag{4.23}$$

Où t_i est la solution cyclique générée par l'algorithme proposé dans le deuxième chapitre pour le lot L_2.

La solution initiale (Sol_{init}) est définie en introduisant le premier produit de L_2 juste après le retrait du produit J_1 de la première cuve. La date d'introduction du premier produit de L_2 à la ligne est définie par :

$t^2_{0,1} = t_{1,J_1} + d_{1,J_1} + c_{(2,J_1),(0,1)}$

Le fonctionnement général de l'algorithme proposé est présenté dans l'Algorithme 8 ci-après.

Algorithme 8 : Schéma général pour l'algorithme proposé

Entrée : S_{dhsp}, S_{chsp}.
Sortie : S_f : matrice des dates $t_{i,j}$ pour le lot L_2.
$liste$=**Etat-ligne**(S_{dhsp}, $t^2_{0,1}$) ;
$dates$=**Calcul-dates**($liste$) ;
Sol_1=**Test-cuve**($dates$) ;
Sol_2=**Test-robot1**(Sol_1) ;
Sol_f=**Test-robot2**(Sol_2) ;

Dans ce qui suit, nous détaillons les différentes étapes de cet algorithme.

4.4.2.1 La procédure Etat-ligne

La procédure **Etat-ligne** consiste à déterminer la position des produits du lot L_1 en cours de traitement au début de la phase transitoire. Le résultat $liste$ de cette procédure est une structure qui indique le numéro du produit ($liste.job$) et le numéro de la cuve ($liste.cuve$) dans laquelle il est en cours de traitement. Le fonctionnement de cette procédure est présenté dans l'Algorithme 9.

Algorithme 9 : La procédure Etat-ligne

Entrée : S_{dhsp}, $t^2_{0,1}$).
Sortie : $liste$, cpt : la taille de la liste $liste$.
Initialisation : $cpt = 0$.
pour ($i \leftarrow 0$ à n) **faire**
  pour ($j \leftarrow 1$ à J_1) **faire**
    si ($t^1_{i,j} > t^1_{0,J_1}$) **alors**
      $liste.cuve[cpt] = i$;
      $liste.job[cpt] = j$;
      $cpt++$;

4.4.2.2 La procédure Calcul-dates

En se basant sur l'état de la ligne au début de la phase transitoire (*liste*), nous pouvons déduire quels produits vont être traités dans chaque cuve. A l'aide de cette procédure, nous déterminons les temps d'occupation de chaque cuve pendant la phase transitoire. A partir de la liste *liste*, nous calculons pour chaque cuve des intervalles de temps qui définissent leurs temps d'occupation. Ainsi, nous calculons les dates de début de trempe d'un produit du lot L_1 (*dates.debut*) et les dates de fin de trempe (*dates.debut*). Le fonctionnement de cette procédure est présenté dans l'Algorithme 10.

Algorithme 10 : La procédure Calcul-dates

Entrée : *liste*, *cpt*.
Sortie : *dates*.
pour ($i \leftarrow 0$ à $cpt - 1$) **faire**
  $l = 0$ (le nombre des produits qui vont occuper chaque cuve) ;
  pour ($j \leftarrow liste.job[i]$ à J_1) **faire**
    $date.debut[l] = t_{liste.cuve[i]-1,j} + d_{liste.cuve[i]-1,j}$;
    $date.fin[l] = t_{liste.cuve[i],j}$;
    $l++$;

4.4.2.3 La procédure Test-cuve

En partant de la solution initiale, nous testons si la disponibilité des cuves permet de recevoir les produits à traiter selon le calendrier des dates déterminé par la relation (4.23). Pour ceci, nous proposons une procédure **Test-cuve** qui permet d'éliminer tout risque de conflits de cuves. Tout au long de cette procédure, nous testons s'il existe une intersection entre la durée de trempe d'un produit de L_2 et la durée de trempe d'un produit de L_1 au niveau de la même cuve. Si une intersection existe, alors une mise à jour des dates est effectuée en retardant la date $t^2_{0,1}$ de sorte qu'aucun conflit ne sera détecté. Le fonctionnement de cette procédure est présenté dans l'Algorithme 11.

Algorithme 11 : La procédure Test-cuve

Entrée : $dates$, $Sol - init$, $liste$, cpt.
Sortie : Sol_1.
pour ($i \leftarrow 0$ à n) **faire**
  pour ($j \leftarrow 1$ à J_2) **faire**
    pour ($p \leftarrow 0$ à $cpt - 1$) **faire**
      si ($i == liste.cuve[p]$) **alors**
        Chercher si un conflit de cuve existe dans la cuve i
        si (Si un conflit existe) **alors**
          les dates $t^2_{i,j}$ sont retardées

4.4.2.4 Les procédures Test-robot1 et Test-robot2

La solution déterminée par la procédure **Test-cuve** n'élimine pas les conflits de robots c'est-à-dire qu'elle ne vérifie pas la disponibilité des robots à effectuer les mouvements qui leurs sont affectés. Nous avons développé deux procédures qui permettent de vérifier la faisabilité de la solution en vérifiant la disponibilité des robots pour effectuer chaque mouvement : une procédure **Test-robot1** pour vérifier la disponibilité du robot R_1 pour effectuer ses mouvements et une procédure **Test-robot2** pour vérifier la disponibilité du robot R_2. Pour cela, nous avons regroupé dans une liste toutes les dates de début des mouvements de transfert des produits appartenant aux deux lots et effectués par chacun des deux robots. Chaque liste sera triée par ordre croissant, et chaque robot doit avoir suffisamment de temps pour effectuer deux déplacements consécutifs de la liste. Les mouvements relatifs aux transferts des produits de L_1 garantissent la disponibilité des robots puisque la solution du DHSP (lot L_1) prise en considération au début de l'algorithme est réalisable et n'est pas remise en cause ainsi que la solution du CHSP (lot L_2). Il reste donc à vérifier les mouvements des robots lorsque les deux mouvements consécutifs concernent deux produits appartenant à deux lots différents. Si un conflit ou une collision est détecté alors une mise à jour des dates de début de transfert au plus tôt de L_2 est effectuée en retardant ces dates par la durée de conflit. Le fonctionnement de ces deux procédures est présenté dans les Algorithmes 12 et 13.

Algorithme 12 : La procédure Test-robot1

Entrée : Sol_1, $dates1$: liste qui contient les dates de début des déplacements des produit de L_1 et L_2 par R_1.
Soit : $Sol - inter$: une solution après chaque mise à jour des dates $t^2_{i,j}$.
Sortie : Sol_2.
refaire : Chercher un conflit de robot dans la liste $dates1$;
si (un conflit existe) **alors**
  mise à jour des dates $t^2_{i,j}$;
  Calculer $Sol - inter$;
  /*Chercher un conflit de cuves sur la base de $Sol - inter$;
  $Sol_2 =$ **Test-cuve**($dates$, $Sol - inter$, $liste$, cpt) ;
sinon
  goto refaire

Algorithme 13 : La procédure Test-robot2

Entrée : Sol_2, $dates2$: liste qui contient les dates de début des déplacements des produit de L_1 et L_2 par R_2.
Soit : $Sol - inter$: une solution après chaque mise à jour des dates $t^2_{i,j}$.
Sortie : Sol_f.
refaire : Chercher un conflit de robot dans la liste $dates2$;
si (un conflit existe) **alors**
  mise à jour des dates $t^2_{i,j}$;
  Calculer $Sol - inter$;
  /*Chercher un conflit de cuves sur la base de $Sol - inter$;
  $Sol - inter =$ **Test-cuve**($dates$, $Sol - inter$, $liste$, cpt) ;
  /*Chercher un conflit de robot pour R_1 sur la base de $Sol - inter$;
  $Sol_f =$ **Test-robot1**($SolSol - inter$, $dates1$) ;
sinon
  goto refaire

4.5 Résultats expérimentaux

Afin de valider cet algorithme, nous l'avons testé sur des instances proposées par Mateo et Companys (2007). Ces instances sont générées aléatoirement en fonction de la largeur des fenêtres de temps. Elles ont été décrites dans le chapitre 2 (**section 2.4.1**). Nous avons utilisé le langage de programmation C++ pour coder notre algorithme et nous avons effectué les tests sur un processeur Intel(R) Core(TM)2 Duo cadencé à 2,6 Ghz et possédant un 1 Gbyte de RAM.

Nous avons généré 10 instances pour chaque valeur de n ($n = 5$ et $n = 7$) et pour chaque fenêtre de temps W_1, W_2 et W_3. En totalité, 60 instances ont été

générées. Pour chaque instance, le nombre de produits appartenant au lot L_1 est égal à 5 et le nombre de produits appartenant au lot L_2 est égal à 5.

Une solution est définie pour le lot L_1 en résolvant le PLM (P_2) par le solveur CPLEX 10.0. Pour le lot L_2, une solution est trouvée en appliquant l'algorithme proposé dans le deuxième chapitre. Connaissant à priori ces deux solutions, l'algorithme que nous avons proposé permet de déterminer les dates d'entrée, au plus tôt, des produits de L_2 et par la suite, la date de sortie du dernier produit de L_2 (C_{max}). Ces dates sont déterminées tout en conservant la solution déjà définie pour L_2 c'est-à-dire la durée du cycle ainsi que les dates de début de transport des produits de chaque cuve.

Pour chaque instance, nous avons cherché la date de fin de traitement de tous les produits des deux lots. En effet, nous envisageons deux méthodes de comparaison. La solution déterminée par l'algorithme proposé sera comparée à chaque solution fournie par chaque méthode.

- la première méthode consiste à chercher un ordonnancement des produits appartenant aux deux lots en utilisant le PLM (P_2). La solution déterminée par cette méthode sera notée S_1.
- la deuxième méthode consiste à chercher un ordonnancement en traitant chaque lot à part. Autrement, il suffit d'attendre que la ligne se vide du lot L_1 pour commencer la production cyclique du lot L_2. La solution fournie par cette méthode sera notée S_2 et définit une borne supérieure.

Nous avons calculé le pourcentage de déviation par rapport à la solution S_1 ainsi que le pourcentage d'amélioration par rapport à la solution S_2. Ces deux pourcentages sont calculés, respectivement, par l'équation 4.24 et l'équation 4.25. Ces deux équations sont les suivantes :

$$\%dév = \frac{C_{max} - S_1}{S_1} * 100 \tag{4.24}$$

$$\%amél = \frac{S_2 - C_{max}}{S_2} * 100 \tag{4.25}$$

Où C_{max} est la valeur du makespan fournie par l'algorithme proposé.

Les pourcentages de déviation et d'amélioration obtenus pour les 10 instances pour chaque groupe selon la taille de l'instance ($n = 5$, $n = 7$) et les fenêtres de temps (W_1, W_2 et W_3) seront présentés par les Figures (4.6)-(4.9).

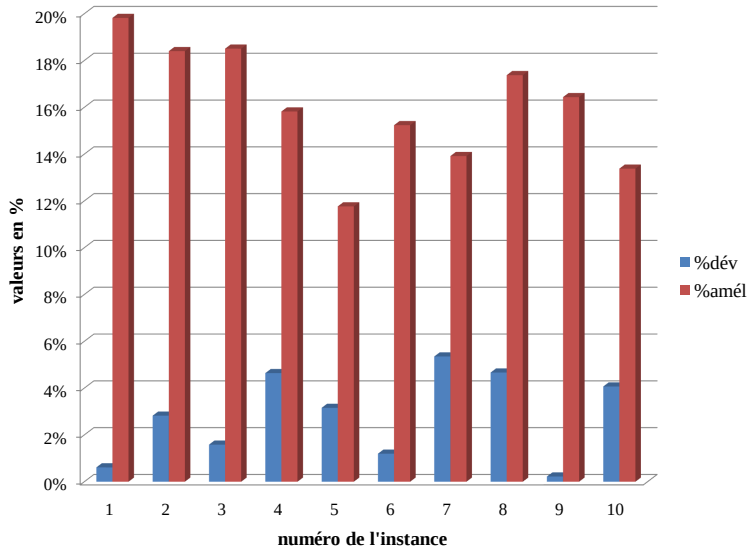

FIGURE 4.6 – Les pourcentages de déviation et d'amélioration des dix instances pour $n = 5$ et pour W_1

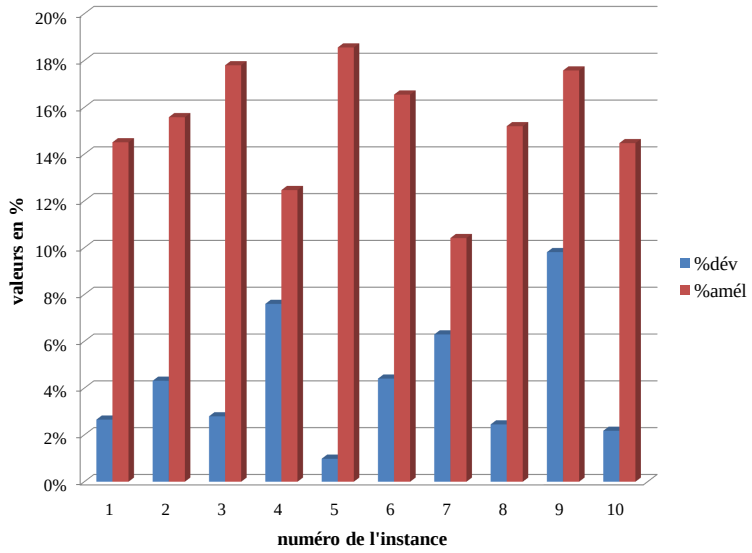

FIGURE 4.7 – Les pourcentages de déviation et d'amélioration des dix instances pour $n = 5$ et pour W_2

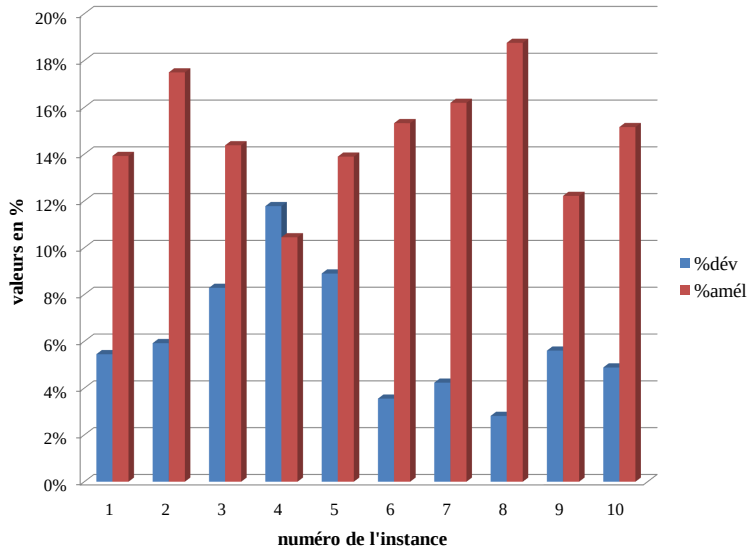

FIGURE 4.8 – Les pourcentages de déviation et d'amélioration des dix instances pour $n = 5$ et pour W_3

Comme le montrent les Figures (4.6), (4.7) et (4.8), les pourcentages de déviation sont nettement inférieurs aux pourcentages d'amélioration, à l'exception de l'instance 4 appartenant au troisième groupe (W_3). Ce dernier cas s'explique

par la présence de conflits qui cause le recul de la date d'entrée des produits de L_2. En outre, plus le pourcentage de déviation est faible plus le pourcentage d'amélioration est élevé. Ceci, s'explique par le fait que plus le C_{max} calculé s'approche de la solution trouvée par (P_2) plus il s'éloigne de la borne supérieure. En plus, aucun des pourcentages de déviation n'est égal à 0 c'est à dire que l'algorithme proposé n'a pas fourni de solutions optimales. Ceci, ne dévalorise pas l'efficacité de l'algorithme que nous avons proposé vu que la solution fournie par le PLM ne garantit pas un ordonnancement périodique pour le lot L_2. Par contre, la solution que nous proposons conserve la solution déjà connue pour L_2 et qui est périodique mais pas forcément optimale.

Pour $n = 7$, la résolution exacte requiert un temps de calcul supérieur à un temps limite que nous avons fixé à 20000 secondes (≈ 5 heures). Pour cette raison, nous n'avons calculé que les pourcentages d'amélioration. Ces pourcentages seront schématisés par la Figure 4.9. Pour le groupe W_1, le plus faible pourcentage est égal à 8,21% par contre le pourcentage le plus élevé est de l'ordre de 25,21%. Même si l'écart entre ces deux valeurs est important, le calcul des pourcentages moyens des 10 instances appartenant aux différents groupes fournit de bon résultat. Ces résultats sont présentés par le Tableau 4.3. Dans ce tableau, nous avons présenté les pourcentages de déviation et d'amélioration moyens ainsi que les temps CPU moyens fournis par l'algorithme 8 (CPU_{moy}) et par la résolution exacte ($CPU_{moy}(P_2)$).

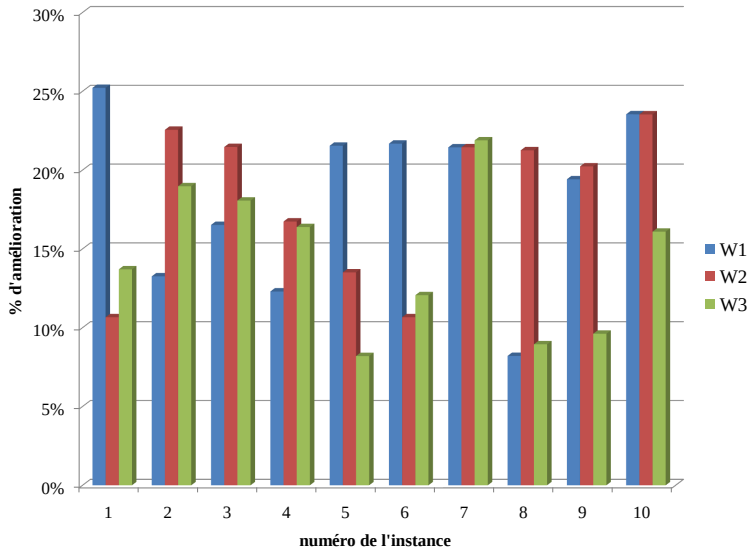

FIGURE 4.9 – Les pourcentages d'amélioration des dix instances pour $n = 7$

TABLE 4.3 – Présentation des résultats pour le premier type d'instances (Mateo et Companys, 2007)

W	$\boldsymbol{n}$	$\%dév(moy)$	$\%amél(moy)$	CPU_{moy}	$CPU_{moy}(P_2)$
W_1	**5**	2,83	16,07	9,282	2540,908
	7	***	18,05	0,264	***
W_2	**5**	6,38	14,33	1,469	8498,751
	7	***	18,20	0,254	***
W_3	**5**	6,14	14,78	7,147	1940,076
	7	***	14,39	0,265	***

D'après le Tableau 4.3, les solutions fournies par notre algorithme sont plus proches des solutions fournies par P_2 pour le groupes d'instances W_1. Les pourcentages de déviation pour W_1 sont plus faibles que ceux des autres groupes. De même pour les pourcentages d'amélioration, ces derniers sont plus élevés pour le groupe W_1. Ceci s'explique par le fait que le nombre de conflits détectés est faible ce qui permet de faire entrer les produits de L_2 le plus tôt possible.

4.6 Conclusion

Dans ce chapitre, nous avons développé une méthode qui permet de déterminer le temps de traitement minimal de deux lots de produits. Ces derniers doivent être traités successivement. Connaissant à priori une solution pour chaque lot, l'objet de l'algorithme proposé est de chercher la date d'introduction au plus tôt du premier produit appartenant au deuxième lot tout en respectant la disponibilité des cuves ainsi que la disponibilité des robots pendant la phase de transition. Cette phase se caractérise par le fait que des produits appartenant aux deux lots sont en cours de traitement.

Nous avons testé, cet algorithme sur un ensemble d'instances. Les résultats trouvés ont montré son efficacité en terme de temps de résolution, ainsi qu'en termes de la déviation des solutions par rapport aux bornes supérieures de chaque instance.

Le but de ce travail était de chercher une solution satisfaisante lorsque l'entreprise doit traiter deux lots de produits successivement. Ainsi, notre intérêt était de proposer une méthode qui permet de trouver les dates d'introduction des produits du deuxième lot dans la ligne sans pour autant attendre qu'elle se vide des produits du premier lot. Dans notre démarche nous n'avons pas remis en cause les ordonnancements initiaux des deux lots. Ceci peut être certes être considéré comme limitatif, mais notre démarche doit plutôt être vue comme une première étape qui a permis de montrer des gains potentiels. Peut être qu'une remise en cause de certains éléments du cycle de L_2 ou certains éléments de L_1 (modifier les dates de transport de certains mouvements) pourrait certainement fournir de plus grands gains.

Conclusion Générale & Perspectives

Nous avons abordé dans cette étude, un ensemble de problématiques, industrielle et scientifique, qui portent sur l'ordonnancement des lignes de traitement de surface. La problématique industrielle, face à une concurrence de plus en plus accentuée entre les pays industrialisés, se rattache à l'augmentation de la productivité et à l'amélioration de la performance de l'entreprise. Le but est de produire avec une bonne qualité, un coût minimum et dans les délais. Ceci revient à respecter un ensemble de contraintes temporelles ainsi que les contraintes de ressources tout au long du procédé de fabrication.

La problématique scientifique s'articule autour du développement de méthodes de résolution qui permettent de trouver un ordonnancement efficace pour les problèmes des ateliers de traitement de surface.

Durant cette thèse, nous avons abordé des problèmes de HSP statiques et plus particulièrement les problèmes de traitement de surface cycliques (CHSP) avec deux robots et avec zone de chevauchement, c'est-à-dire, avec une collision possible entre les deux robots. Nous avons développé une approche de résolution de ce type de problème afin de surmonter certaines insuffisances des méthodes de résolution trouvées dans la littérature. L'approche développée a prouvé sa validité

et sa pertinence à l'aide d'un ensemble de tests effectués sur des instances de la littérature.

Dans un deuxième temps, nous avons étendu notre approche pour tenir compte de gammes opératoires ne suivant pas l'implantation des cuves. Cette méthode a été également testée sur des cas pratiques empruntés de la littérature afin de montrer sa performance.

Sur ces deux approches, notre contribution sur le CHSP multi-robots porte sur 2 aspects :

- amélioration du temps de résolution, avec comme objectif de traiter des lignes complexes de taille important, là où les méthodes exactes sont limitées ;
- proposition de meilleures solutions par rapport à la plupart des autres approches, principalement parce-que l'algorithme élaboré ne travaille pas sur une affectation unique, mais cherche à la fois la meilleure affectation et le meilleur séquencement.

Dans un dernier temps, nous avons abordé le problème dynamique en proposant un programme linéaire mixte qui permet de résoudre le DHSP avec deux robots. Ensuite, nous avons traité un problème de type HSP, dans le cadre d'un fonctionnement mixte, pour lequel il est nécessaire de traiter dans une même installation des produits différents et des rafales de produits identiques. Ainsi, la solution la plus intéressante pour les industriels est d'être capable d'alterner des modes de production non-cycliques et cycliques. Pour cela, nous avons proposé un algorithme qui permet de déterminer la date au plus tôt d'introduction des produits appartenant au deuxième lot tout en évitant tout conflit de cuves et de robots. Cet algorithme a été testé sur des instances et a prouvé son efficacité.

Une des perspectives d'un tel travail est d'étendre nos approches développées à d'autres domaines. En effet, nous ne saurions avoir démontré leur applicabilité de mise en œuvre sur un seul type de problème. Ainsi, des adaptations peuvent être apportées à nos méthodes afin de les appliquer sur d'autres types de pro-

blèmes comme par exemple le problème du voyageur de commerce avec fenêtres de temps. Nous comptons aussi, dans un futur travail développer d'autres méthodes de résolution, qu'elles soient exactes ou approchées pour les problèmes DHSP. Enfin, parmi nos perspectives, nous trouvons naturellement le passage à des lignes comportant plus que deux robots.

Bibliographie

Armstrong, R., S. Gu et L. Lei, 1996. A greedy algorithm to determine the number of transporters in a cyclic electroplating process. *Institute of Industrial Engineers Transactions*, 28(5), pp. 347–355. 29

Armstrong, R. D., L. Lei et S. Gu, 1994. A bounding schedule for deriving the minimal cycle time of a single hoist n-stage with time-window constraints. *European Journal of Operational Research*, 78(1), pp. 130–140. 29, 52

Baptiste, P., C. Bloch et C. Varnier, 2001. *Ordonnancement de la production, ch 9 : Ordonnancement des lignes de traitement de surface.* P. Lopez an F. Roubellat Eds. Hermès. 21

Baptiste, P., B. Legeard, M.-A. Manier et C. Varnier, 1996. Résolution d'un problème d'ordonnancement avec la PLC. *Journal Européen des Systèmes Automatisés*, 30(2-3), pp. 274–284. 29

Bloch, C., M.-A. Manier, P. Baptiste et C. Varnier, 2008. *Production Scheduling, chapter 8 : Hoist scheduling Problem.* Edited by Pierre Lopez and François Roubellat, ISTE - Willey ,London, april 2008, ISBN 9780470611050. 28

Bloch, C., C. Varnier et P. Baptiste, 1999. A taboo search combined with a modified shifting bottleneck heuristic for solving a blocking scheduling problem with bounded processing times. Dans *15 th Conference of the International*

Federation of Operational Research Societies, pages 125–126. Beijing, China. 29

Bracker, W. et W. Chapman, 1985. Optimization of a programmable hoists using linear programming. *Printed Circuit Fabrication*, 8(2), pp. 39–42. 29

Carlier, J., P. Chrétienne, J. Erschler, C. Hanen, P. Lopez, A. Munier, E. Pinson, M. Portmann, C. Prins, C. Proust et P. Villon, 1993. Gotha, (groupe de recherche en ordonnancement théorique et appliqué). les problèmes d'ordonnancement. *RAIRO-Recherche Opérationnelle*, 27(1), pp. 77–150. 6, 9

Caux, C., G. Fleury, M. Gourgand et P. Kellert, 1995. Couplage méthodes d'ordonnancement-simulation pour l'ordonnancement de systèmes industriels de traitement de surface. *RAIRO-Recherche Opérationnelle*, 29(4), pp. 391–413. 22, 29

Caux, C. et H. Pierreval, 1997. Solving a hoist scheduling problem as a sequencing problem. Dans *International Federation of Automatic Control /International Federation for Information Processing Conference on Managment and Control of Production and Logistics*, pages 372–376. Campinas, Brazil. 29

Che, A. et C. Chu, 2004. Single-track multi-hoist scheduling problem : a collision-free resolution based on a branch-and-bound apporach. *International Journal of Production Research*, 42(12), pp. 2435–2456. 49, 60

Che, A. et C. Chu, 2005. A polynomial algorithm for no-wait cyclic hoist scheduling in an extended electroplating line. *Operations Research Letters*, 33(3), pp. 274–284. 29, 39

Che, A. et C. Chu, 2008. Optimal scheduling of material handling devices in a pcb production line : problem formulation and a polynomial algorithm. *Mathematical Problems in Engineering*, 2008(Article ID 364279). 60

Chen, H., C. Chu et J.-M. Proth, 1998. Cyclic scheduling of a hoist with time window constraints. *IEEE Transactions on Robotics and Automation*, 14(1), pp. 144–152. 29, 31, 37, 52

Cheng, C.-C. et S. Smith, 1995. A constraint-posting framework for scheduling under complex constraints. Dans *Symposium on Emerging Technologies and Factory Automation*, Vol. 1, pages 269–280. Paris, France. 29

El-Amraoui, A., M.-A. Manier, A. E. Moudni et M. Benrejeb, 2008. 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*, 2, pp. 612–623. 29

El-Amraoui, A., M.-A. Manier, A. E. Moudni et M. Benrejeb, 2009. Robustness integration in a transport scheduling problem. *International REview of Automatic COntrol (IREACO)*, 2(4), pp. 476–480. 29

El-Amraoui, A., M.-A. Manier, A. E. Moudni et M. Benrejeb, 2012. Resolution of the two-part cyclic hoist scheduling problem with bounded processing times in complex lines configuration. *European Journal of Industrial Engineering (EJIE)*, 6(4), pp. 1–3. 29

Fargier, H. et J. Lamothe, 2001. Handling soft constraints in hoist scheduling problems : the fuzzy approach. *Engineering Applications of Artificial Intelligence*, 14, pp. 387–399. 29, 142

Fleury, G., 1995. Applications de méthodes stochastiques inspirées du recuit simulé à des problèmes d'ordonnancement. *Automatique productique informatique industrielle*, 29(4-5), pp. 445–470. 29

Ge, Y. et Y. Yih, 1995. Crane scheduling with time windows in circuit board production lines. *International Journal of Production Research*, 33(5), pp. 1187–1199. 29

Grötschel, I., M. Jünger et G. Reinelt, 1985. Facets of the linear ordering polytope. *Mathematical Programming*, 33, pp. 43–60. 47

Hanen, C., 1994. Periodic scheduling of several hoists. Dans *Fourth International Workshop on Project Management and Scheduling*, pages 91–101. Leuven, Belgium. 29

Hanen, C. et A. Munier, 1993. Ordonnancement cyclique d'un robot sur une ligne de galvanoplastie : modèles et algorithmes. Dans *LITP/IBP, Pierre and Marie Curie University*, page LITP research report 93.30. Paris. 29

Heinrichs, U. et C. Moll, 1997. *On the scheduling of one-dimensional transport systems*. Report 97-277, Zentrun für Parelleles Rechnen, University of Cologne, Cologne, Germany. 47

Hindi, K. et K. Fleszar, 2004. A constraint propagation heuristic for the single-hoist, multiple-products scheduling problem. *Computers & Industrial Engineering*, 47, pp. 91–101. 29, 143

Hitomi, K., 1996. Manufacturing excellence for 21st century production. *Technovation*, 16(1), pp. 33–41. 5

Jegou, D., D. Kim, P. Baptiste et H. L. Kwang, 2006. A contract net based intelligent agent system for solving the reactive hoist scheduling problems. *Expert Systems with Applications*, 30, pp. 156–167. 29

Kats, V. et E. Levner, 1997. A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling. *Operations Research Letters*, 21(4), pp. 171–179. 29, 64, 65

Lacomme, P., 1998. *Optimisation des systèmes de production : mèthodes stochastiques et approche multi-agents*. Thèse de doctorat, Clermont-Ferrand. 29

Lamothe, J., 1996. *Une approche pour l'ordonnancement dynamique d'un atelier de traitement de surface*. Thèse de doctorat, L'Ecole Nationale Superieure de l'Aeronotique et de l'Espace. vi, 23, 24, 29, 139, 140

Lei, L., 1993. Determining the optimal starting times in a cyclic schedule with a given route. *Computers and Operations Research*, 20(8), pp. 807–815. 29, 31, 37, 38, 40

Lei, L., R. Armstrong et S. Gu, 1993. Minimizing the fleet size with dependent time-window and single -track constraint. *Operations Research Letters*, 14(2), pp. 91–98. 29

Lei, L. et T.-J. Wang, 1989a. On the optimal cyclic schedules of single hoist electroplating processes. *Rutgers University, Reasearch report no.89-0006*. 21, 29, 31, 36

Lei, L. et T.-J. Wang, 1989b. A proof : the cyclic hoist scheduling problem is np-complete. *Rutgers University, Reasearch report no.89-0016*. 2, 22, 28

Lei, L. et T.-J. Wang, 1991. The minimum common-cycle algorithm for cyclic scheduling of two material handling hoists with time window constraints. *Management Science*, 37(12), pp. 1629–1639. 29, 45, 46, 60

Leung, J. et E. Levner, 2006. An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times. *Operations Research Letters*, 34, pp. 465–472. 44

Leung, J. et G. Zhang, 2003. Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence. *IEEE Transactions On Robotics and Automation*, 19(3), pp. 480–484. 29, 47, 59, 60

Leung, J., G. Zhang, X. Yang, R. Mak et K. Lam, 2004. Optimal cyclic multi-hoist scheduling : a mixed integer programming approach. *Operations Research*, 52(6), pp. 965–976. 29, 47, 53, 54, 59, 60, 76, 82, 84, 85, 86, 88

Levner, E., V. Kats et V. Levit, 1997. An improved algorithm for cyclic flowshop scheduling in a robotic cell. *European Journal of Operational Research*, 97(3), pp. 500–508. 29, 31, 39, 48

Lim, J.-M., 1997. A genetic algorithm for a single hoist scheduling in the printed-circuit-board electroplating line. *Computers and industrial engineering*, 33(3-4), pp. 789–792. 29, 31, 39, 40

Liu, J. et Y. Jiang, 2005. An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem. *Operations Research*, 53(2), pp. 313–327. 48, 60

Liu, J., Y. Jiang et Z. Zhou, 2002. Cyclic scheduling of a single hoist in extended electroplating lines : a comprehensive integer programming solution. *IIE Transactions*, 34, pp. 905–914. 29, 31, 33

Lopez, P. et P. Esquirol, 2001. *Ordonnancement de la production, ch 2 : Concepts et méthodes de base en ordonnancement de la production.* P. Lopez an F. Roubellat Eds. Hermès. 8, 10

Manier, M.-A., 1994. *Contribution à l'ordonnancement cyclique du système de manutention d'une ligne de galvanoplastie.* Thèse de doctorat, Université de Besançon. iii, v, 19, 29, 31, 34, 35, 44, 45, 60, 92, 113, 114, 123, 127, 128, 129, 131, 185

Manier, M.-A. et P. Baptiste, 1994. Etat de l'art : ordonnancement de robots de manutention en galvanoplastie. *Automatique Productique Informatique Industrielle (A.P.I.I.-RAIRO)*, 28(1), pp. 7–35. 28

Manier, M.-A. et C. Bloch, 2003. A classification for hoist scheduling problems. *The International Journal of Flexible Manufacturing Systems*, 15, pp. 37–55. 2, 3, 24

Manier, M.-A. et S. Lamrous, 2008. An evolutionary for the design and scheduling

of electroplating facilities. *Journal of Mathematical Modeling and Algorithms*, 7, pp. 197–215. iii, 29, 53, 60, 184, 185

Manier, M.-A., C. Varnier et P. Baptiste, 2000. Constraint-based model for cyclic multi-hoists scheduling problem. *Production Planning and Control*, 11(3), pp. 244–257. 20, 29, 46, 60

Mateo, M. et R. Companys, 2006. Hoist scheduling in a chemical line to produce batches with identical sizes of different products. Dans *Sixième Conférence Francophone de Modélisation et Simulation : Modélisation, Optimisation et Simulation des Système (MOSIM 06)*, pages 677–684. Rabat, Morocoo. 29

Mateo, M. et R. Companys, 2007. *New computational experiences on the hoist scheduling problem for cyclic manufacturing of different products.* Rapport de Recherche Externe, Université Polytechnique de Cataluna, E-08028, Barcelone, Espagne. viii, ix, 29, 82, 83, 84, 85, 86, 88, 89, 159, 164

Mateo, M., R. Companys et J. Bautista, 2000. Bounded cycle time for the cyclic hoist scheduling problem. Dans *First World Conference on Production and Operations Management*, pages 1–10. Seville, Spain. 29

Mintzberg, H., 1982. *The Structuring of Organization : A Synthesis of the Research.* Prentice-Hall, Inc. 5

Ng, W., 1995. Determining the optimal number of duplicated process tanks in a single hoist circuit board production line. *Computers & industrial Engineering*, 28(4), pp. 681–688. 29

Ng, W., 1996. A branch and bound algorithm for hoist scheduling of a circuit board production line. *The International Journal of Flexible Manufacturing Systems*, 8, pp. 45–65. 83

Ng, W. et J. Leung, 1997. Determining the optimal move times for a given cyclic

schedule of material handling hoist. *Computers & Industrial Engineering*, 32(3), pp. 595–606. 29

Paul, H., C. Bierwirth et H. Kopfer, 2007. A heuristic scheduling procedure for multi-item hoist production lines. *International Journal of Production Economics*, 105, pp. 54–69. 29, 143

Pinedo, M., 1995. Scheduling : Theory, algorithms and systems. *Prenctice hall, Englewood Cliffs, NJ*. 26

Riera, D. et N. Yorke-Smith, 2002. An improved hybrid model for the generic hoist scheduling problem. *Annals of Operations Research*, 115, pp. 173–191. 29

Rosse-Bloch, C., 1999. *Contribution à l'ordonnancement dynamique de lignes de traitement de surface*. Thèse de doctorat, Besançon. 29

Shapiro, G., 1985. *Hoist scheduling for a PCB electroplating facility*. Thèse de doctorat, Thesis for the Degree of Master Science (Program in Operations Research), Graduate Faculty of North Carolina State University. 45, 127

Shapiro, G. et H. Nuttle, 1988. Hoist scheduling for a pcb electroplating facility. *IIE Transactions*, 20(2), pp. 157–167. 25, 29, 31, 34, 35, 120

Song, W., R. Storch et Z. Zabinsky, 1995. An example for scheduling a chemical processing tank line. Dans *Symposium on Emerging Technologies and Factory Automation*, Vol. 1, pages 475–482. Paris, France. 29

Song, W., Z. Zabinsky et R. Storch, 1993. An algorithm for scheduling a chemical processing tank line. *Production Planning and Control*, 4(4), pp. 323–332. 29, 31, 32, 140

Spacek, P., M.-A. Manier et A. E. Moudni, 1999. Control of an electroplating line in the max and min algebras. *International Journal of Systems Science*, 30(7), pp. 759–778. 29

Thesen, A. et L. Lei, 1990. An expert scheduling system for material handling hoists. *Journal of Manufacturing Systems*, 9(3), pp. 247–252. 29

Varnier, C., 1996. *Extensions du Hoist Scheduling Problem cyclique. Résolution basée sur un traitement des contraintes disjonctives en programmation en logique avec contraintes*. Thèse de doctorat, Besançon. 22, 29, 135, 137

Varnier, C., A. Bachelu et P. Baptiste, 1997. Resolution of the cyclic multi-hoist scheduling problem. *Information Systems and Operational Research*, 35(4), pp. 309–324. 29

Varnier, C. et N. Jeunehomme, 2000. A cyclic approach for the multi-product hsp. Dans *Workshop on Project Managment and Scheduling*. Osnabrück, Germany. 29

W.Phillips, L. et P. S.Unger, 1976. Mathematical programming solution of a hoist scheduling program. *AIIE Transactions*, 8, pp. 219–225. iii, 13, 29, 31, 33, 40, 92, 183

Yang, G., D. Ju et W. Zheng, 2001. Solving multiple hoist scheduling problems by use simulated annealing. *Journal of Software*, 12(1), pp. 11–17. 29, 52, 60

Yih, Y., 1994. An algorithm for hoist scheduling problem. *International Journal of Production Research*, 32(3), pp. 501–516. 29, 141, 143

Yih, Y., T. Liang et H. Moskowitz, 1993. Robot scheduling in a circuit board production line : a hybrid or/ann approach. *Institute of Industrial Engineers Transactions*, 25(2), pp. 26–33. 29

Yin, C. et Y. Yih, 1992. Crane scheduling in a flexible electroplating line : a tolerance based approach. *Journal of Electronics Manufacturing*, 2, pp. 137–144. 140, 141

Zhou, Z. et L. Li, 2003. Single hoist cyclic scheduling with multiple tanks : a material handling solution. *Computers & Operations Research*, 30, pp. 811–819. 29, 123

Zhou, Z. et L. Li, 2009. A solution for cyclic scheduling of multi-hoists without overlapping. *Annals of Operations Research*, 168, pp. 5–21. iii, 29, 60, 186

Zhou, Z. et J. Liu, 2008. A heuristic algorithm for the two-hoist cyclic scheduling problem with overlapping hoist coverage ranges. *IIE Transactions*, 40, pp. 782–794. iii, vi, viii, 29, 48, 60, 62, 63, 64, 66, 82, 83, 84, 88, 89, 90, 91, 92, 97, 114, 179

Annexe A

Détail de la solution prise des instances de Zhou et Liu (2008)

A.1 Description de la solution pour le robot 1

TABLE A.1 – Description de la solution pour R_1

dates d'exécution		mouvement
debut	fin	
0	13	transfert porteur de 0 à 1
13	17	robot vide de 1 à 2
17	89	pause en 2
89	102	transfert porteur de 2 à 3
102	122	robot vide de 3 à 13
122	135	transfert porteur de 13 à 14
135	143	robot vide de 14 à 10
143	156	transfert porteur de 10 à 11
156	170	robot vide de 11 à 4
170	183	transfert porteur de 4 à 5
183	185	robot vide de 5 à 6
185	217	pause en 6
217	230	transfert porteur de 6 à 7
230	242	robot vide de 7 à 1
242	255	transfert porteur de 1 à 2
255	257	robot vide de 2 à 3
257	275	pause en 3
275	288	transfert porteur de 3 à 4
288	302	robot vide de 4 à 11
302	315	transfert porteur de 11 à 12
315	319	robot vide de 12 à 14
319	332	transfert porteur de 14 à 15
332	352	robot vide de 15 à 5
352	365	transfert porteur de 5 à 6
365	377	robot vide de 6 à 0
377	401	pause en 0

A.2 Description de la solution pour le robot 2

TABLE A.2 – Description de la solution pour R_2

dates d'exécution		mouvement
debut	fin	
45	58	transfert porteur de 8 à 9
58	72	robot vide de 9 à 16
72	85	transfert porteur de 16 à 17
85	158	pause en 17
158	168	robot vide de 17 à 12
168	171	transfert porteur de 12 à 13
171	183	robot vide de 13 à 7
183	229	pause en 7
229	242	transfert porteur de 7 à 8
242	244	robot vide de 8 à 9
244	257	transfert porteur de 9 à 10
257	267	robot vide de 10 à 15
267	280	transfert porteur de 15 à 16
280	288	robot vide de 16 à 20
288	301	transfert porteur de 20 à 21
301	305	robot vide de 21 à 19
305	318	transfert porteur de 19 à 20
318	322	robot vide de 20 à 18
322	335	transfert porteur de 18 à 19
335	357	robot vide de 19 à 8
357	401	pause en 8

Annexe B

Données des exemples pris de la littérature

B.1 Exemple 1 proposé par W.Phillips et S.Unger (1976)

TABLE B.1 – Données de l'exemple de "P&U"

C_{ij}	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	11	14	16	14	19	22	24	26	29	6	8	10	0
1	11	0	2	5	2	8	10	13	15	17	10	3	1	11
2	14	2	0	2	0	5	8	10	13	15	12	6	3	14
3	16	5	2	0	2	3	5	8	10	13	15	8	6	16
4	14	2	0	2	0	5	8	10	13	15	12	6	3	16
5	19	8	5	3	5	0	3	5	7	10	18	11	9	19
6	22	10	8	5	8	3	0	2	5	7	20	14	11	22
7	24	13	10	8	10	5	2	0	2	5	23	16	14	24
8	26	15	13	10	13	7	5	2	0	2	25	19	16	26
9	29	17	15	13	15	10	7	5	2	0	27	21	19	29
10	6	10	12	15	12	18	20	23	25	27	0	7	9	6
11	8	3	6	8	6	11	14	16	19	21	7	0	2	8
12	10	1	3	6	3	9	11	14	16	19	9	2	0	10
13	0	11	14	16	14	19	22	24	26	29	6	8	10	0
L_i	120	150	90	120	90	30	60	60	45	130	120	90	30	
U_i	∞	200	120	180	125	40	120	120	75	∞	∞	120	60	
d_i	31	22	22	22	25	23	22	22	22	47	27	22	30	

B.2 Exemple 2 proposé par Manier et Lamrous (2008)

TABLE B.2 – Données de l'exemple "ligne 1"

C_{ij}	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	13	12	11	10	9	9	8	8	6	6	5	5	0
1	13	0	4	5	6	7	7	8	9	10	10	11	12	13
2	12	4	0	4	5	6	6	7	8	9	9	10	11	12
3	11	5	4	0	4	5	6	6	7	8	9	10	10	11
4	10	6	5	4	0	4	5	6	6	7	8	9	9	10
5	9	7	6	5	4	0	4	5	5	6	7	8	8	9
6	9	7	6	6	5	4	0	4	4	5	6	7	8	9
7	8	8	7	6	6	5	4	0	4	4	5	6	7	8
8	8	9	8	7	6	5	4	4	0	5	5	6	7	8
9	6	10	9	8	7	6	5	4	5	0	4	5	5	6
10	6	10	9	9	8	7	6	5	5	4	0	4	5	6
11	5	11	10	10	9	8	7	6	6	5	4	0	4	5
12	5	12	11	10	9	8	8	7	7	5	5	4	0	5
13	0	13	12	11	10	9	9	8	8	6	6	5	5	0
L_i	180	60	30	120	180	30	30	60	60	300	60	60	120	
U_i	∞	120	90	240	240	90	90	120	120	420	120	120	180	
d_i	23	29	19	24	19	19	24	24	20	24	24	19	30	

B.3 Exemple 3 proposé par Manier et Lamrous (2008)

TABLE B.3 – Données de l'exemple "ligne 2"

C_{ij}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	5	5	6	8	9	10	12	13	15	16	18	19	21	0
1	5	0	4	5	7	8	9	11	12	14	15	16	18	19	5
2	5	4	0	4	6	7	7	10	11	12	14	15	17	18	5
3	6	5	4	0	5	6	6	8	9	11	13	14	15	17	6
4	8	7	6	5	0	4	5	7	8	10	11	12	14	15	8
5	9	8	7	6	4	0	4	6	7	8	10	11	13	14	9
6	10	9	7	6	5	4	0	5	6	7	9	10	12	13	10
7	12	11	10	8	7	6	5	0	4	6	7	8	10	11	12
8	13	12	11	9	8	7	6	4	0	5	6	7	9	10	13
9	15	14	12	11	10	8	7	6	5	0	5	5	7	8	15
10	16	15	14	13	11	10	9	7	6	5	0	4	6	7	16
11	18	16	15	14	12	11	10	8	7	5	4	0	5	6	18
12	19	18	17	15	14	13	12	10	9	7	6	5	0	5	19
13	21	19	18	17	15	14	13	11	10	8	7	6	5	0	21
14	0	5	5	6	8	9	10	12	13	15	16	18	19	21	0
L_i	300	180	60	60	180	60	30	60	60	180	60	60	60	660	240
U_i	∞	300	120	120	240	120	120	120	120	300	180	120	150	720	∞
d_i	15	24	14	25	29	19	20	24	20	25	19	20	25	18	35

B.4 Exemple 4 proposé par Manier (1994)

TABLE B.4 – Données de l'"example 4"

tank		0 (0)	1 (10)	2 (11)	3 (12)	4 (1)	5 (2)	6 (4)	7 (3)	8 (5)	9 (6)	10 (7)	11 (8)	12 (9)	13 (0)
0	(0)	0	6	8	10	11	14	14	16	19	22	24	26	29	0
1	(10)	6	0	7	9	10	12	12	15	18	20	23	25	27	6
2	(11)	8	7	0	2	3	6	6	8	11	14	16	19	21	8
3	(12)	10	9	2	0	1	3	3	6	9	11	14	16	19	10
4	(1)	11	10	3	1	0	2	2	5	8	10	13	15	17	11
5	(2)	14	12	6	3	2	0	0	2	5	8	10	13	15	14
6	(4)	14	12	6	3	2	0	0	2	5	8	10	13	15	14
7	(3)	16	15	8	6	5	2	2	0	3	5	8	10	13	16
8	(5)	19	18	11	9	8	5	5	3	0	3	5	7	10	19
9	(6)	22	20	14	11	10	8	8	5	3	0	2	5	7	22
10	(7)	24	23	16	14	13	10	10	8	5	2	0	2	5	24
11	(8)	26	25	19	16	15	13	13	10	7	5	2	0	2	26
12	(9)	29	27	21	19	17	15	13	15	10	7	5	2	0	29
13	(0)	0	6	8	10	11	14	14	16	19	22	24	26	29	0
L_i		120	120	90	30	150	90	90	120	30	60	60	45	130	120
U_i		∞	∞	120	60	200	120	125	180	40	120	120	75	∞	∞
d_i		31	27	22	30	22	22	22	25	23	22	22	22	47	

B.5 Exemple 5 proposé par Zhou et Li (2009)

TABLE B.5 – Données de l'"example 5"

tank	0	1	2	3	4	5	6	7	8	9	10	11	12
L_i	160	160	150	180	50	20	20	60	20	20	50	180	20
U_i	∞	180	180	200	70	40	40	80	40	50	70	240	40
d_i	14	14	14	14	14	14	14	14	14	14	14	14	14
tank	13	14	15	16	17	18	19	20	21	22	23	24	
L_i	80	30	50	100	70	70	140	50	60	30	70	150	
U_i	120	50	80	130	100	90	180	70	80	50	90	180	
d_i	14	14	14	14	14	14	14	14	14	14	14	14	

Annexe C

Données des lignes et détail de la solution trouvée

C.1 Ligne *Copper*

TABLE C.1 – Données de la ligne *Copper*

C_{ij}	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	0	3,1	4,2	5,6	7,9	8,8	9,9	11,7	10,6	12,9	14	15,2
1	3,1	0	1,1	2,5	4,7	5,7	6,8	8,6	7,5	9,8	10,9	12,1
2	4,2	1,1	0	1,4	3,6	4,6	5,6	7,4	6,4	8,6	9,8	10,9
3	5,6	2,5	1,4	0	2,2	3,2	4,3	6,1	5	7,3	8,4	9,6
4	7,9	4,7	3,6	2,2	0	1	2	3,8	2,8	5,0	6,2	7,3
5	8,8	5,7	4,6	3,2	1	0	1	2,8	1,8	4	5,2	6,3
6	9,9	6,8	5,6	4,3	2	1	0	1,8	0,8	3	4,1	5,3
7	11,7	8,6	8,4	6,1	3,8	2,8	1,8	0	1	1,2	2,3	3,5
8	10,6	7,5	7	5	2,8	1,8	0,8	1	0	2,2	3,4	4,5
9	12,9	9,8	3,9	7,3	5	4	3	1,2	2,2	0	1,1	2,3
10	14	10,9	9,8	8,4	6,2	5,2	4,1	2,3	3,4	1,1	0	1,2
11	15,2	12,1	4,9	9,6	7,3	6,3	5,3	3,5	4,5	2,3	1,2	0
L_i	60	120	45	600	120	240	60	60	60	60	120	1800
U_i		∞	∞	840	300	420	180	120	180	180	300	3000
d_i	16,7	18,9	16,9	18,3	18	16,8	17,6	16,8	19,2	20,8	22,5	23,6
b_i	1	1	1	2	1	1	1	1	1	1	1	6

TABLE C.2 – Description de la solution pour le robot 1

dates d'exécution		**mouvement**
debut	**fin**	
0	16,7	transfert porteur de 0 à 5
16,7	17,7	robot vide de 5 à 6
17,7	51,3	pause en 6
52,3	68,9	transfert porteur de 6 à 7
68,9	77,5	robot vide de 7 à 1
77,5	96,4	transfert porteur de 1 à 0
96,4	100,6	robot vide de 0 à 2
100,6	117,5	transfert porteur de 2 à 1
117,5	122,2	robot vide de 1 à 4
122,2	140,2	transfert porteur de 4 à 3
140,2	145,9	robot vide de 3 à 0
145,9	307,9	pause en 0

TABLE C.3 – Description de la solution pour le robot 2

dates d'exécution		**mouvement**
debut	**fin**	
0	80,7	pause en 11
80,7	104,3	transfert porteur de 11 à 9
104,3	105,4	robot vide de 9 à 10
105,4	105,6	pause en 10
105,6	128,1	transfert porteur de 10 à 11
128,1	130,4	robot vide de 11 à 9
130,4	164,3	pause en 9
164,3	185,1	transfert porteur de 9 à 4
185,1	188,9	robot vide de 4 à 7
188,9	205,7	transfert porteur de 7 à 8
205,7	210,7	robot vide de 8 à 3
210,7	229	transfert porteur de 3 à 2
229	233,6	robot vide de 2 à 5
233,6	256,7	pause en 5
256,7	273,5	transfert porteur de 5 à 6
273,5	274,3	robot vide de 6 à 8
274,3	293,5	transfert porteur de 8 à 10
293,5	294,7	robot vide de 10 à 11
294,7	307,9	pause en 11

C.2 Ligne *Black Oxide*

TABLE C.4 – Données de la ligne *Black Oxide*

C_{ij}	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	0	3	4,1	5,2	6,4	8	9	10	11,1	12,5	13,3	14,3
1	3	0	1,8	2,9	4,1	5,7	6,7	7,7	8,8	10,2	11	12
2	4,1	1,8	0	1,1	2,3	3,9	4,9	5,9	7	8,4	9,2	10,2
3	5,2	2,9	1,1	0	1,2	2,8	3,8	4,8	5,9	7,3	8,1	9,1
4	6,4	4,1	2,3	1,2	0	1,5	2,6	3,6	4,7	6,1	6,9	7,9
5	8	5,7	3,9	2,8	1,5	0	1	2	3,1	4,5	5,3	6,4
6	9	6,7	4,9	3,8	2,6	1	0	1	2,1	3,5	4,3	5,3
7	10	7,7	5,9	4,8	3,6	2	1	0	1,1	2,5	3,3	4,3
8	11,1	8,8	7	5,9	4,7	3,1	2,1	1,1	0	1,4	2,2	3,2
9	12,5	10,2	8,4	7,3	6,1	4,5	3,5	2,5	1,4	0	0,8	1,8
10	13,3	11	9,2	8,1	6,9	5,3	4,3	3,3	2,2	0,8	0	1
11	14,3	12	10,2	9,1	7,9	6,4	5,3	4,3	3,2	1,8	1	0
L_i	120	240	90	60	120	60	120	60	60	60	60	180
U_i		420	120	90	145	90	180	120	90	240	90	240
d_i	25,9	14,6	20,6	12,7	18,8	13,1	18,6	12,6	18,7	13	12,4	18,6
b_i	1	2	1	1	1	1	1	1	1	1	1	1

TABLE C.5 – Description de la solution pour le robot 1

dates d'exécution		**mouvement**
debut	**fin**	
0	25,9	transfert porteur de 0 à 11
25,9	29,1	robot vide de 11 à 8
29,1	47,8	transfert porteur de 8 à 7
47,8	55,5	robot vide de 7 à 1
55,5	70,1	transfert porteur de 1 à 0
70,1	74,2	robot vide de 0 à 2
74,2	85,9	pause en 2
85,9	106,5	transfert porteur de 2 à 1
106,5	190,5	pause en 1
190,5	193,4	robot vide de 1 à 3
193,4	206,1	transfert porteur de 2 à 1
206,1	210,2	robot vide de 2 à 0

TABLE C.6 – Description de la solution pour le robot 2

dates d'exécution		mouvement
debut	fin	
0	18,6	transfert porteur de 11 à 10
18,6	19,6	robot vide de 10 à 11
19,6	19,7	robot vide de 11 à l'extrémité du rail
19,7	77,5	pause
77,5	78,6	robot vide de l'extrémité du rail à 10
78,6	91	transfert porteur de 10 à 9
91	94,5	robot vide de 9 à 6
94,5,4	113,1	transfert porteur de 6 à 5
113,1	114,6	robot vide de 5 à 4
114,6	133,4	transfert porteur de 4 à 3
133,4	138,2	robot vide de 3 à 7
138,2	150,8	transfert porteur de 7 à 6
150,8	154,3	robot vide de 6 à 9
154,3	167,3	transfert porteur de 9 à 8
167,3	172	robot vide de 8 à 5
172	173,1	pause en 5
173,1	186,2	transfert porteur de 5 à 4
186,2	194,1	robot vide de 4 à 11
194,1	210,2	pause en 11

C.3 Ligne *Zinc*

TABLE C.7 – Données de la ligne *Zinc*

C_{ij}	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	1,8	3,7	5,4	8,7	10,3	12,3	14,3	18,2	19,8	21,5	23,3	25,1	26,6	28,4	30,3
1	1,8	0	1,9	3,5	6,9	8,5	10,5	12,5	16,4	18	19,7	21,5	23,3	24,8	26,6	28,5
2	3,7	1,9	0	1,6	5	6,6	8,5	10,6	14,4	16	17,8	19,5	21,4	22,9	24,6	26,5
3	5,4	3,5	1,6	0	3,4	5	6,9	9	12,8	14,4	16,2	17,9	19,8	21,3	23	24,9
4	8,7	6,9	5	3,4	0	1,6	3,6	5,6	9,5	11,1	12,8	14,6	16,4	17,9	23	21,6
5	10,3	8,5	6,6	5	1,6	0	2	4	7,9	9,5	11,2	13	14,8	16,3	18,1	20
6	12,3	10,5	8,5	6,9	3,6	2	0	2,1	5,9	7,5	9,2	11	12,8	14,4	16,1	18
7	14,3	12,5	10,6	9	5,6	4	2,1	0	3,9	5,5	7,2	9	10,8	12,3	14,1	15,9
8	18,2	16,4	14,4	12,8	9,5	7,9	5,9	3,9	0	1,6	3,3	5,1	6,9	8,5	10,2	12,1
9	19,8	18	16	14,4	11,1	9,5	7,5	5,5	1,6	0	1,7	3,5	5,3	6,8	8,6	10,5
10	21,5	19,7	17,8	16,2	12,8	11,2	9,2	7,2	3,3	1,7	0	1,8	3,6	5,1	6,9	8,8
11	23,3	21,5	19,5	17,9	14,6	13	11	9	5,1	3,5	1,8	0	1,8	3,4	5,1	7
12	25,1	23,3	21,4	19,8	16,4	14,8	12,8	10,8	6,9	5,3	3,6	1,8	0	1,5	3,3	5,2
13	26,6	24,8	22,9	21,3	17,9	16,3	14,4	12,3	8,5	6,8	5,1	3,4	1,5	0	1,7	3,6
14	28,4	26,6	24,6	23	19,7	18,1	16,1	14,1	10,2	8,6	6,9	5,1	3,3	1,7	0	1,9
15	30,3	28,5	26,5	24,9	21,6	20	18	15,9	12,1	10,5	8,8	7	5,2	3,6	1,9	0
L_i	60	25	50	30	11,5	180	240	180	120	120	240	30	120	57	6,5	1680
U_i		45	90	60	12,5	300	300	240	300	300	300	60	180	180	7,5	1800
d_i	16,9	14	14,1	10,8	15,6	10,8	16,3	18,1	10,8	13,9	16	17,3	15,5	10,7	18,1	18,1
b_i	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4

TABLE C.8 – Description de la solution pour le robot 1

dates d'exécution		mouvement
debut	fin	
0	16,9	transfert porteur de 0 à 6
16,9	18,9	robot vide de 6 à 5
18,9	142,2	pause en 5
142,2	153	transfert porteur de 5 à 4
153	164,5	pause en 4
164,5	180,1	transfert porteur de 4 à 3
180,1	223	pause en 3
223	233,8	transfert porteur de 3 à 2
233,8	242,3	robot vide de 2 à 6
242,3	256,9	pause en 6
256,9	273,2	transfert porteur de 6 à 7
273,2	283,8	robot vide de 7 à 2
283,8	297,9	transfert porteur de 2 à 1
297,9	322,9	pause en 1
322,9	336,9	transfert porteur de 1 à 0

TABLE C.9 – Description de la solution pour le robot 2

dates d'exécution		**mouvement**
debut	**fin**	
0	17,35	pause en 7
17,35	35,45	transfert porteur de 7 à 8
35,45	38,75	robot vide de 8 à 10
38,75	122,25	pause en 10
122,25	138,25	transfert porteur de 10 à 11
138,25	143,35	robot vide de 11 à 8
143,35	155,45	pause en 8
155,45	166,25	transfert porteur de 8 à 9
166,25	176,75	robot vide de 9 à 15
176,75	194,85	transfert porteur de 15 à 13
194,85	198,25	robot vide de 13 à 11
198,25	215,55	transfert porteur de 11 à 14
215,55	222,05	pause en 14
222,05	240,15	transfert porteur de 14 à 15
240,15	243,75	robot vide de 15 à 13
243,75	251,85	pause en 13
251,85	262,55	transfert porteur de 13 à 12
262,55	267,85	robot vide de 13 à 9
267,85	286,25	pause en 9
286,25	300,15	transfert porteur de 9 à 10
300,15	303,75	robot vide de 10 à 12
303,75	382,55	pause en 12
382,55	398,05	transfert porteur de 12 à 5
398,05	402,05	robot vide de 5 à 7

Résumé :

Dans cette thèse, nous nous intéressons principalement à l'étude du fonctionnement cyclique mono-produit des ateliers de traitement de surface. Notre contribution porte sur le problème d'ordonnancement associé connu dans la littérature sous le nom Cyclic Hoist Scheduling Problem (CHSP). L'objet de cette thèse est de proposer des méthodes efficaces pour la résolution des problèmes de traitement de surface dans le cas où les produits à traiter sont du même type. Nous traitons en particulier le cas où le nombre des robots présents sur la ligne est égal à deux, ce qui augmente le nombre des contraintes du problème, sachant que dans le cas mono robot, ce problème a été prouvé NP-Complet. Pour cela, nous proposons une méthode qui combine deux heuristiques et un programme linéaire mixte. Cette méthode permet notamment d'affecter les mouvements de transport à l'un des deux robots tout en gérant les risques de collision entre eux, lorsque la gamme opératoire des produits à traiter suit l'implantation des cuves. Par la suite, nous proposons une extension du modèle au cas de lignes complexes. Enfin, nous étudions le cas d'un fonctionnement mixte, pour lequel il est nécessaire de traiter dans une même installation des produits différents et des rafales de produits identiques. Dans ces conditions, la solution la plus intéressante pour les industriels est de pouvoir alterner des modes de production dynamiques et cycliques. Pour cela, nous proposons une méthode efficace permettant de résoudre le problème d'ordonnancement associé à la phase transitoire relative à ce type de fonctionnement. Elle consiste en particulier à chercher les dates d'entrée au plus tôt des produits. La principale difficulté identifiée consiste ici à passer du mode dynamique au mode cyclique, c'est-à-dire à rejoindre un cycle à partir d'une solution courante donnée, en supposant que ce cycle est connu à priori. Les méthodes élaborées dans les divers cas traités sont validées par des tests sur des benchmarks de la littérature.

Mots clés : Atelier de traitement de surface, Hoist Scheduling Problem, Ordonnancement cyclique et dynamique, contraintes de transport, Programmation Linéaire Mixte, heuristiques.

Abstract :

In this thesis, our interest is focused on the Cyclic Hoist Scheduling Problem (CHSP) in automated electroplating lines. The aim of this study is to propose an algorithm to solve the two-hoists cyclic scheduling problem. This one consists in finding a repetitive sequence of hoists' moves, while avoiding collision between the hoists which share a common track. The objective is to minimize the period of this repetitive cycle for single part-type production. This problem was proved to be NP-complete for lines with a single hoist. The fact that two hoists are available on the line increases the number of constraints of the problem. Then we propose a solving method combining two heuristics and a Mixed Integer Linear Program. It enables us to solve both assignment and sequencing problems, while considering spatial constraints related to hoist'moves. Subsequently, we propose an extension of the model which is adapted to complex lines. Finally, our interest is focused on solving a HSP for which it is necessary to treat in the same facility a batch of various products and a batch of identical products. Under these conditions, the most interesting solution for manufacturers is to be able to alternate the production of two batches. For this goal, we propose an efficient method to solve the scheduling problem associated. Finally, our proposed methods are validated by experimentations based on benchmarks from the literature.

Keywords : Electroplating facilities, Cyclic and Dynamic Hoist Scheduling Problems, transportation constraints, Mixed Integer Linear Programming, heuristics ;