Ordonnancement des ateliers de traitement de surface pour une production cyclique et mono-produit

Fabien Mangione

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

THESE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : "Génie Industriel"

préparée au laboratoire GILCO (Gestion Industrielle Logistique et COnception)
dans le cadre de **l'Ecole Doctorale "Organisation Industrielle et Systèmes de production"**

présentée et soutenue publiquement

par

Fabien MANGIONE

le 17/07/2003

Ordonnancement des ateliers de traitement de surface pour une production cyclique et mono-produit

Directeur de thèse : Yannick FREIN
Co-encadrant : Bernard PENZ

JURY

M. Michel GOURGAND	, Président
M. Alessandro AGNETIS	, Rapporteur
M. Pierre BAPTISTE	, Rapporteur
M. Stéphane DAUZERE-PERES	, Rapporteur
M. Yannick FREIN	, Directeur de thèse
M. Bernard PENZ	, Co-encadrant
Mme. Nadia BRAUNER-VETTIER	, Examinateur

A mes parents

Remerciements

Je tiens à remercier avant tout Bernard PENZ, professeur à l'INP Grenoble, pour avoir accepter d'encadrer ma thèse et pour m'avoir consacré un grand nombre d'heures de travail lors de réunions très constructives. Il m'a ainsi conforté dans mon désir de poursuivre ma carrière dans l'enseignement et la recherche.

Je tiens à remercier tout particulièrement Nadia BRAUNER, maître de conférence à l'université Pierre Mendes France, pour avoir collaborer à mes travaux ainsi que pour avoir corriger toutes mes erreurs.

Je remercie Yannick FREIN, professeur à l'INP Grenoble, pour avoir été d'accord d'être mon directeur de thèse malgré qu'il avait été décidé qu'on ne travaillerai pas ensemble dés le début. Je le remercie surtout en tant que directeur du laboratoire GILCO pour m'avoir accueilli durant ces trois années.

Mes remerciements vont ensuite à tous les membres du jury :
à Michel GOURGAND, professeur à l'université Blaise Pascal de Clermont-Ferrand, qui m'a fait l'honneur de présider ce jury et qui à travers le groupe Bermudes qu'il dirige m'a permis de rencontrer les membres de la communauté scientifique,
à Alessandro AGNETIS, professeur à l'université de Sienne (Italie), qui a fait l'effort d'être rapporteur bien qu'il ne soit pas francophone,
à Pierre BAPTISTE, professeur à l'université Polytechnique de Montréal (Canada), qui a accepté d'être rapporteur,
à Stephane DAUZERE-PERES, professeur à l'Ecole des Mines de Nantes, pour ses remarques qui ont permis la version définitive de ce document.

Cette thèse ne se serait pas passer dans d'aussi bonnes conditions sans tous les membres du laboratoire GILCO (les enseignants-chercheurs, les doctorants ainsi que les secrétaires et l'ingénieur réseau).

Même si elle n'a pas pris part dans mes travaux de recherche je tiens à remercier Peggy ZWOLINSKI pour tous ses conseils et son aide pour mes enseignements dans le cadre de mon monitorat.

Je remercie mes parents et ma famille pour m'avoir supporté au quotidien du-

rant ces trois années. Enfin je remercie mes amis pour tous les moments passés en dehors du laboratoire et en particulier les deux autres webmasters de Nanarland (www.nanarland.com).

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

La concurrence industrielle nécessite d'utiliser au mieux toutes les ressources potentielles de l'entreprise. Cette optimisation peut se faire d'un point de vue global avec les outils de type *supply chain management*. Mais elle doit aussi s'attacher à traiter les problèmes plus spécifiques à tous les niveaux et notamment au niveau de la production.

Du point de vue des ateliers de production, cette optimisation intervient depuis sa conception jusqu'à son utilisation, voir même jusqu'au recyclage. Afin de rentabiliser les lignes, il faut, pendant la production, mener une gestion efficace qui profite au mieux des capacités offertes.

Cette gestion est souvent liée à l'ordonnancement proposé et ceci quel que soit le type d'atelier (atelier de montage, atelier de fabrication mécanique, atelier d'usinage...). Parmi les ateliers existants, les ateliers de traitement de surface ont certaines particularités qui leur sont propres. Les travaux menés durant ce doctorat traitent de la gestion de ces ateliers.

L'étude présentée fait suite à un travail [36] réalisé au sein du laboratoire GILCO et en collaboration avec une entreprise de la région grenobloise. Cette étude traitait de l'ordonnancement des lignes de galvanoplastie.

La galvanoplastie est un traitement, basé sur l'électrolyse, qui permet d'appliquer un dépôt sur les pièces. Ce procédé est utilisé lorsque les caractéristiques des pièces nécessitent d'être modifiées (conductivité, dureté, anticorrosion ...) ou simplement à titre décoratif (dorure, argenture ...). Ces opérations consistent à immerger, successivement, les pièces dans différents bains. Comme les produits dans les cuves peuvent être nocifs (acides), les entreprises cherchent à automatiser au maximum leurs ateliers. Les transports des pièces sont alors réalisés par l'intermédiaire d'un ou plusieurs robots. Ces ateliers se trouvent généralement en milieu de gamme de fabrication des pièces, entre la phase d'usinage et de montage. Comme les temps d'immersion peuvent être longs suivant les traitements, ce type d'atelier constitue généralement le goulot d'étranglement du procédé de fabrication. Il devient donc nécessaire d'optimiser son fonctionnement.

Les procédés de traitements de surface se différencient des ateliers de montage ou d'usinage car le temps passé par un produit dans une cuve doit se trouver dans un intervalle bien défini. Cette contrainte est appelée "*fenêtre de temps*". La différence entre la durée maximale et la durée minimale est alors appelée marge sur la durée de trempe.

Sur les ateliers de galvanoplastie, comme beaucoup d'autres ateliers faisant appel à des traitements dans des cuves, le robot est souvent la ressource critique. En effet, suivant le nombre de cuves et le nombre de produits sur la ligne, le robot ne se déplace pas suffisamment vite afin de réaliser les déplacements dans les temps. Il est donc impératif de tenir compte des mouvements du robot lors de la mise en place du planning de production.

Cette particularité a été prise en compte dès les années 70 par Phillips et Hunger [58]. Depuis, beaucoup de travaux, utilisant les outils de l'automatique des systèmes à événements discrets, de la recherche opérationnelle et de l'intelligence artificielle, ont été réalisés pour trouver des méthodes de résolutions.

Dans ce mémoire, nous nous limiterons au cas où l'atelier produit un seul type de pièce, sauf lors la section 3.2. Bien que simple dans sa formulation, nous verrons dans la suite que ce problème est très difficile à traiter. La solution du problème est une solution cyclique. Avec ces conditions, l'objectif est de trouver le cycle des mouvements du robot qui maximise la productivité. Les méthodes actuelles ne prennent, généralement, en compte qu'un nombre limité de types de cycles ou nécessitent des temps de calcul beaucoup trop importants. Dans ce mémoire, nous nous attacherons à regarder si les cycles étudiés sont suffisants pour obtenir de bonnes solutions ou si il existe des cycles optimaux plus complexes.

Ces travaux s'attachent donc à trouver les cycles optimaux pour des lignes contenant, dans un premier temps, un nombre restreint de cuves. Puis, dans un second temps, sur un nombre quelconque de cuves, une conjecture sera proposée sur les cycles optimaux. La conjecture sera vérifiée pour un certain nombre de configurations.

Pour présenter ces travaux, le mémoire est composé de six chapitres :

Dans le chapitre 1, nous présenterons brièvement le fonctionnement et le pilotage des lignes de traitement de surface. Nous proposerons une étude de la bibliographie des travaux déjà réalisés sur les problèmes d'ordonnancement des ateliers en ligne et du cas particulier du *hoist scheduling problem*.

Dans le chapitre 2, nous présenterons en détail le problème étudié dans la thèse et poserons les hypothèses que nous utiliserons dans les chapitres suivants. Nous

définirons ensuite les notations et les outils qui seront utilisés par la suite.

Dans le chapitre 3, nous nous attacherons au problème à deux machines. Nous étudierons, dans un premier temps (section 3.1), le cas où un seul type de produits doit être traité et lorsque les temps de trempe sont égaux et compris dans une marge qui peut être nulle ou infinie. Nous élargirons, ensuite, au cas où les durées de trempe sont différentes et les marges quelconques. Dans un second temps (section 3.2), nous regarderons le cas plus général où plusieurs types de produits doivent être traités en même temps. Nous verrons, dans ce chapitre, comment adapter la solution obtenue par l'algorithme de Gilmore et Gomory aux caractéristiques du problème.

Le cas d'une ligne contenant trois cuves et pour une production mono-produit équilibrée sera étudié dans le chapitre 4. Nous montrerons quels sont les cycles optimaux lorsque les marges sur les durées de trempe sont nulles ou infinies. Nous proposerons, ensuite, les cycles optimaux pour des marges quelconques.

Nous examinerons, dans le chapitre 5, le cas d'une ligne à quatre cuves. Nous nous focaliserons sur le cas où toutes les marges sont nulles. Ce problème est alors appelé *flowshop* robotisé sans attente. Nous prouverons que dans le cas d'une ligne équilibrée, il existe des cycles 1-, 2- et 3-périodiques optimaux.

Enfin, le chapitre 6 sera consacré au *flowshop* robotisé sans attente pour m cuves, avec m quelconque. Nous proposerons dans ce chapitre une conjecture sur les cycles optimaux, puis nous prouverons cette conjecture pour un certain nombre de situations.

Une conclusion finale permettra de récapituler tous les résultats obtenus durant le doctorat. Des perspectives de recherche seront aussi proposées sur les problèmes qui restent ouverts.

Chapitre 1

Présentation des lignes de traitement de surface et leurs contraintes pour le pilotage

Dans ce chapitre, nous présenterons, dans un premier temps, les structures physiques d'un cas particulier de ligne de production que sont les lignes de traitement de surface (section 1.1). Nous décrirons l'utilisation de telles lignes ainsi que leurs spécificités. Dans un second temps, nous aborderons les problèmes liés au pilotage de ces lignes en fonction des critères de performance poursuivis (section 1.2). Enfin nous ferons un état de l'art sur le sujet mais aussi sur les problèmes d'ordonnancement proches de celui traité dans ce mémoire (section 1.3).

1.1 Description physique des lignes

1.1.1 Les lignes de traitement de surface simple

Pour des applications mécaniques ou électriques, il est souvent nécessaire de modifier l'état surfacique des pièces utilisées. Pour cela, des opérations de type traitement de surface sont effectuées. Ces opérations consistent à tremper successivement les pièces dans des cuves comportant des produits agissant sur les caractéristiques du matériau. Ces lignes de production se retrouvent notamment dans l'industrie mécanique pour des opérations de trempe par exemple, dans l'industrie électronique pour la fabrication des circuits imprimés ou pour les opérations de dépôt d'argent, de cuivre ou d'autres métaux.

Les traitements de surface se font généralement au milieu de la gamme de fabrication d'un produit, entre l'usinage et le montage. Ces types de ligne de production sont composées de plusieurs cuves contenant un (des) produit(s) qui permettent de réaliser les traitements souhaités : attaque acide, rinçage, dépôt de cuivre, d'argent ou même d'or, bains électrolytiques, etc... La gamme opératoire d'une pièce

est donc constituée d'un enchaînement de traitements à réaliser dans les cuves correspondantes.

Etant donnée la nature de certains bains de traitement un robot est utilisé pour effectuer les déplacements des pièces. Les pièces sont disposées sur des porteurs qui permettent au(x) robot(s) de faire les déplacements de cuve en cuve. Ce robot est généralement installé sur un rail pour permettre les déplacements. Le robot est aussi appelé palan (hoist en anglais) à cause de cette liberté de mouvement dans une seule dimension. On appelle tronçon une partie de l'atelier qui est desservie par tous les robots se déplaçant sur un même rail. La figure 1.1 représente un tronçon d'une ligne de traitement de surface.

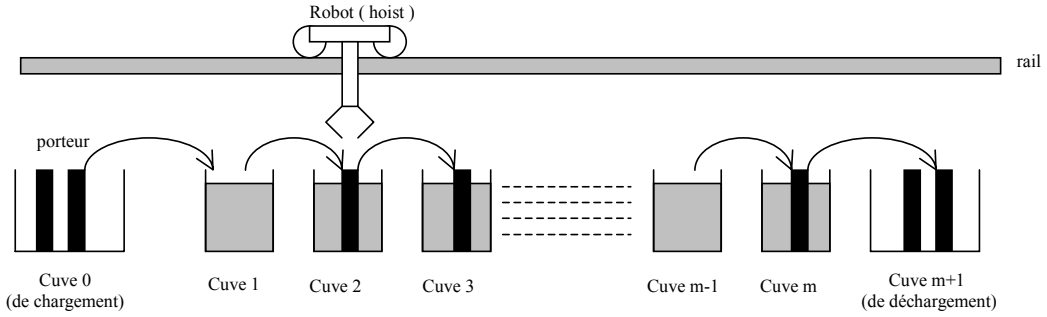

Figure 1.1 – Schéma d'un tronçon

A tout cela il faut ajouter les systèmes assurant les règles de sécurité et d'hygiène tels que les systèmes de ventilation et de lavage de fumées, de traitement des eaux, de récupération, etc...

1.1.1.1 Les cuves

Les cuves (ou bassins) servent à recevoir les produits nécessaires aux traitements souhaités. Elles sont réalisées dans des matériaux neutres par rapport à leur contenant : Polypropylène, fibre de verre, PVC, acier, acier inoxydable...

Les traitements

La principale utilisation de telles lignes est généralement pour des traitements de type galvanoplastie. La galvanoplastie est utilisée pour l'application de dépôts qui peuvent être : protecteurs (chromage, galvanisation), décoratifs (dorure, argenture, nickelage) ou encore destinés à la reproduction de pièces métalliques à partir d'un moule en cire, qu'on rend conducteur au moyen d'un vernis.

Le métal est déposé à partir d'un bain de sels sur la cathode. L'anode est constituée soit du métal à déposer, soit d'un matériau insoluble (graphite, plomb antimoiné, ferrosilicium, etc.). L'électrolyse peut être effectuée sous courant (relativement bas afin d'obtenir un dépôt cohérent) ou par des procédés autocatalytiques (nickel, cuivre). La galvanoplastie trouve des débouchés très importants dans différents domaines : protection anticorrosion, bijouterie (bouchons de parfum), industries électriques (bras de disjoncteurs) et électroniques (antennes de téléphones portables).

Les caractéristiques des produits sont déterminées par des ingénieurs chimistes (dans le cas de bains électrolytiques), par des ingénieurs électriciens (pour des dépôts de matériaux conducteurs ou isolants), ou même par des designers. Ces produits sont vérifiés régulièrement afin de maintenir un niveau et une qualité acceptable pour l'utilisation requise. Chacune des cuves est caractérisée par son utilisation (chargement, déchargement, transfert), par le type de traitement qu'elle contient, par sa capacité et par sa position sur la ligne.

Cuves particulières

– **Cuves de chargement** : Ce sont les cuves d'entrée par où les pièces brutes arrivent des ateliers amonts. Elles ont la particularité d'être généralement considérées de capacité infinie. Elles ne contiennent soit aucun produit, soit des produits neutres permettant aux pièces de ne pas se détériorer. Ce qui fait que l'on peut faire l'analogie entre les cuves de chargement et les stocks d'entrée des lignes de production traditionnelles.

– **Cuves de déchargement** : Ce sont les cuves équivalentes aux cuves de chargement mais en sortie de lignes. Elles sont équivalentes aux stocks de fin de ligne des ateliers plus classiques.

1.1.1.2 Les moyens de transport

Les porteurs

Pour des raisons pratiques (trop grande quantité, difficultés d'accroche. . .), les pièces ne sont pas transportées une par une d'une cuve à l'autre. Toutes les pièces devant subir le même traitement sont installées sur des porteurs. Il existe plusieurs types de porteurs qui dépendent des pièces à transporter :

– les tonneaux permettent de déplacer les pièces de petite taille (antennes de téléphone portable). Les tonneaux sont des cylindres contenant les pièces. Ils sont mis en rotation autour de leur axe afin que, sur chaque pièce, toute la surface soit traitée.

– Les paniers sont des récipients, remplis de pièces à traiter, qui sont déposés dans les cuves. Ils sont utilisés, généralement, pour les pièces de petites tailles.
– les cadres servent pour les pièces de plus grande taille, celles-ci sont alors simplement déposées sur un des crochets du cadre.

Les robots

Les robots (ou palans) ont pour rôle de transporter les porteurs d'une cuve à l'autre. Pour cela, ils se déplacent sur des rails. Chaque robot permet de déplacer les porteurs sur un nombre de cuves défini au moment de la conception de la ligne, ou variable en fonction des productions en cours. Mais ils sont toujours liés à un rail précis et ils ne peuvent pas en changer. Les mouvements d'un robot pour déplacer un porteur se décomposent en quatre mouvements élémentaires.

1. Déplacement à vide, pour venir se positionner au dessus du porteur à déplacer ;
2. Descente et saisie pour récupérer le porteur, puis levée et égouttage ;
3. Déplacement sous charge ;
4. Stabilisation, descente puis libération et remontée à vide.

La figure 1.2 résume ce fonctionnement, les flèches en pointillé représentent les mouvements à vide tandis que les flèches continues représentent les mouvements sous charge (généralement moins rapide que ceux à vide).

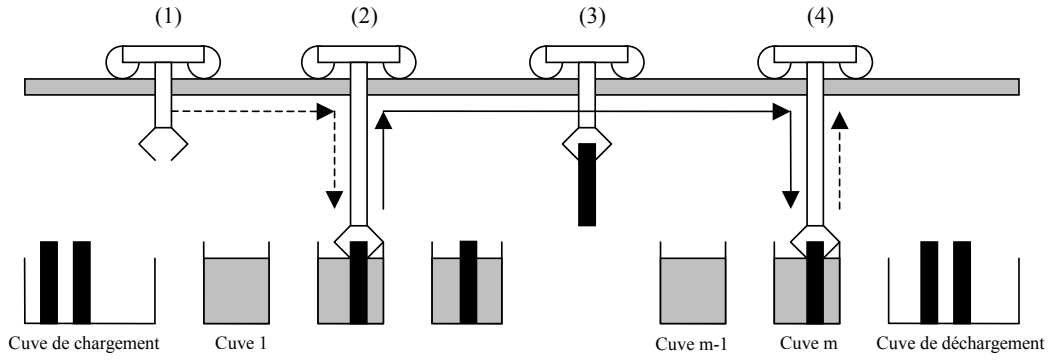

Figure 1.2 – Activités du robot pour le transport d'une pièce

Chaque robot a ses propres caractéristiques mais celles qui servent dans la majeure partie des cas pour le pilotage sont les suivantes : la capacité (nombre de porteurs transportables à chaque voyage), les temps de descente, de montée, d'accélération et de décélération, la vitesse maximale (à vide et sous charge) et les cuves accessibles.

1.1.2 Lignes complexes

Pour des questions de facilité de gestion, de commodité ou de gains potentiels de productivité, il peut être intéressant de ne pas se contenter de lignes contenant simplement des bacs simples avec un unique robot. On utilise alors des cuves de capacité supérieure à un (cuves multi-bacs), ou plusieurs robots, ou plusieurs tronçons en parallèle, etc...

1.1.2.1 Cuves multi-bacs

Si certains traitements, dans une même cuve, sont très longs par rapport aux traitements dans les autres cuves, il est préférable de disposer de cuves multi-bacs. En effet, pour éviter qu'une cuve ne soit goulot d'étranglement, on augmente sa capacité en lui permettant d'accueillir plusieurs porteurs en même temps. La figure 1.3 symbolise une ligne contenant une cuve multi-bac.

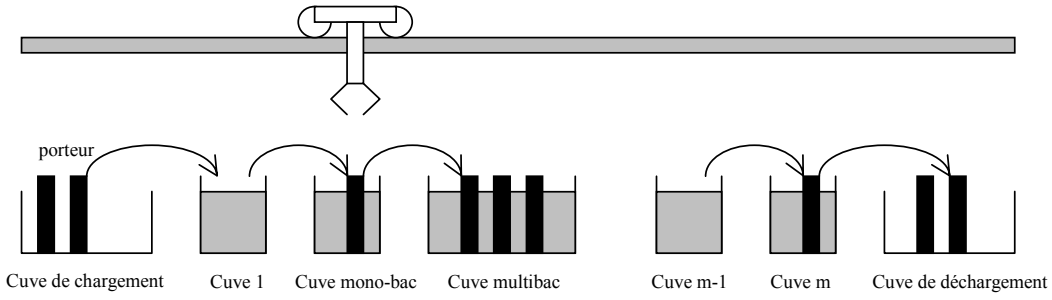

Figure 1.3 – Tronçon avec une cuve multi-bacs

1.1.2.2 Tronçons multi-robots

Afin d'augmenter la productivité, dans le cas où le robot est limitant, il est possible de disposer plusieurs robots sur un même tronçon afin qu'ils se partagent les tâches. On parle alors de tronçons multi-robots. Etant sur un même rail, les robots ne peuvent pas se croiser, il faut donc gérer le risque de collision. Généralement, chaque robot dessert une partie du tronçon et une seule cuve sert de lien entre les parties.

1.1.2.3 Ligne multi-tronçons

Enfin, si le nombre de cuves est trop important, pour une question de place, essentiellement, on ne peut pas disposer les cuves sur un seul tronçon. Plusieurs tronçons sont alors disposés en parallèle, reliés par des cuves de transfert. Ces cuves contiennent majoritairement des produits neutres et sont mobiles sur un rail afin de faire passer les porteurs d'un tronçon à l'autre (figure 1.4).

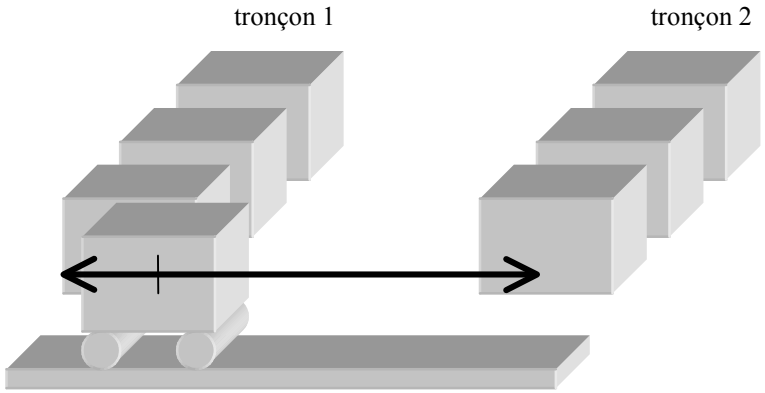

Figure 1.4 – Cuve de transfert

Trois implantations sont habituellement utilisées pour des lignes multi-tronçons (en U, O ou H) comme le montre la figure 1.5. Sur cette figure, les cuves de transfert sont représentées en gris plus foncé et les trajets des porteurs par des flèches en pointillés.

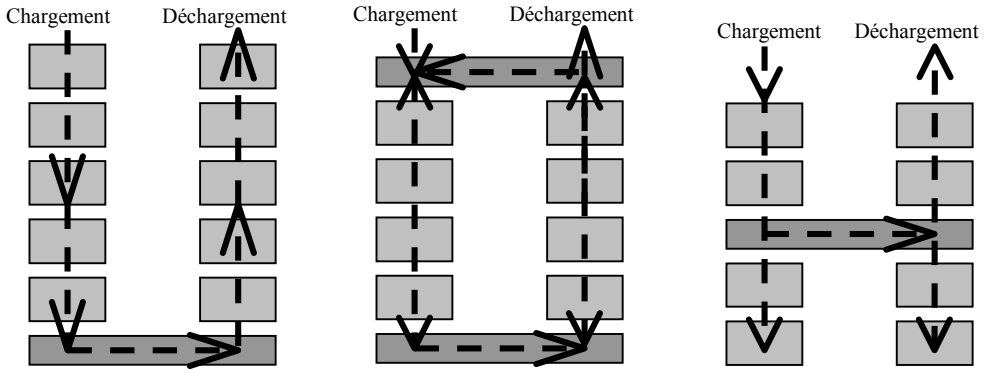

Figure 1.5 – Implantation en U, O ou H

Enfin, pour des lignes produisant un grand nombre de types de pièces différents, toutes les caractéristiques citées sont utilisées. La figure 1.6 représente une ligne réelle avec trois tronçons, trois cuves de transfert (en gris plus foncé) et sept robots. Les cuves accessibles par chacun des robots sont symbolisées sur la figure par les double-flèches. Cette ligne a été étudié dans [36].

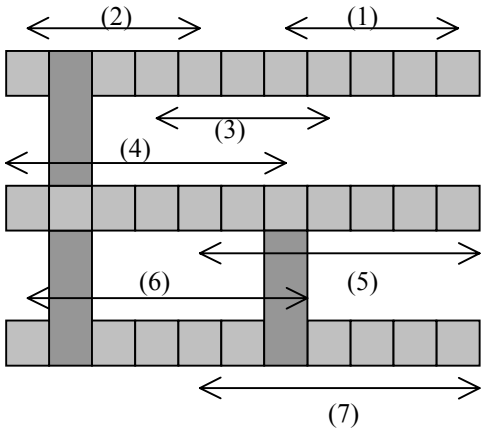

Figure 1.6 – Exemple de ligne industrielle

1.2 Le pilotage des lignes de traitement de surface

Suivant le type de commandes, les méthodes de gestion de la ligne diffèrent. En effet, les types de production vont dépendre du nombre de types de produits à traiter, mais aussi du temps entre la commande et la production.

1.2.1 La production

1.2.1.1 Production statique

Les pièces à produire sont connues à l'avance pour une période donnée. Il faut alors trouver dans quel ordre et à quel moment faire entrer les pièces dans la ligne ainsi que les durées effectives de trempe et les mouvements du robot.

1.2.1.2 Production cyclique

Dans ce type de production, la ligne produit généralement les mêmes types de pièces et dans des quantités qui varient peu sur la période étudiée. Cet type de production est, par exemple, utilisé lorsque l'entreprise décide de produire par campagne. Cette production peut s'appliquer, aussi, quand les gammes de traitement entre les différents types de produits sont les mêmes et que les temps de trempe sont très proches. Il est alors possible de considérer tous les types de produits comme un seul lors de l'ordonnancement.

Le but de l'ordonnancement est de définir quel cycle de production optimise un critère qui peut être la maximisation de la productivité ou la minimisation de l'encours par exemple.

1.2.1.3 Production dynamique

Les pièces à produire ne sont pas connues à l'avance mais arrivent "au fil de l'eau". Il faut alors décider, lorsque les nouvelles pièces arrivent, si l'on doit commencer à traiter la pièce ou si il faut attendre avant de lancer le traitement, tout ceci en fonction de l'état du système et des arrivées de pièces éventuelles.

1.2.2 L'ordonnancement

La gestion des ateliers est un problème récurrent, notamment le fait de savoir qui fait quoi, quand et où. On parle alors de problème d'ordonnancement que l'on peut définir de la manière suivante :

Définition 1 *Ordonnancer : affecter des ressources dans le temps pour achever un ensemble d'activités*

Généralement, les problèmes d'ordonnancement d'ateliers de production consistent à affecter, sur les machines (ressources), un ensemble de tâches (opérations ou travaux), elles-mêmes constituées d'un ensemble d'opérations. Chaque opération doit être réalisée sur une machine de l'atelier de production. Mais, l'ordonnancement doit aussi définir les dates de réalisation de chacune des opérations. Ceci doit être fait en respectant les contraintes liées à l'utilisation des ressources, à la disponibilité des tâches, aux durées d'exécution des opérations... Trouver un bon ordonnancement revient alors à trouver l'ordonnancement qui minimise le critère de performance souhaité (minimisation du coût, de la durée, du retard ou maximisation de la productivité, juste-à-temps...).

Parmi les problèmes d'ordonnancement des lignes de production on différencie trois grands types de problèmes :

- *openshop* : l'ordre des opérations n'est pas fixé à l'avance et doit être déterminé lors de l'ordonnancement,
- *jobshop* : l'ordre des opérations est fixé mais peut varier d'un travail à l'autre,
- *flowshop* : la séquence des opérations est la même pour tous les travaux.

Un *flowshop* est dit de permutation si toutes les machines doivent effectuer les tâches dans le même ordre.

Les contraintes principales de ces problèmes sont les suivantes :

Temps de process : durée pendant laquelle la pièce et la machine sont occupées. Il peut être libre, c'est à dire une fois l'opération terminée la pièce peut attendre sur la machine, ou aller dans un stock intermédiaire. Il peut aussi devoir respecter des contraintes de sans-attente, ce qui signifie que, une fois l'opération réalisée, la pièce doit immédiatement quitter la machine pour que l'opération suivante sur cette pièce ait lieu. L'instant de sortie de la pièce est alors déterminé dès son entrée sur la ligne.

Contraintes de précédence : suivant le type d'opération à réaliser, il peut être permis, par exemple, de commencer une opération avant que l'opération précédente ne soit terminée. Plusieurs types de précédences peuvent être définies :

- début-début : une opération peut débuter seulement lorsque l'opération précédente a commencé depuis un temps fixé.
- début-fin : une opération peut débuter seulement lorsque l'opération précédente est finie depuis un temps fixé.
- fin-fin : une opération peut finir seulement lorsque l'opération précédente est finie depuis un temps fixé.

Temps de préparation : pendant les périodes de préparation, la machine est indisponible, mais l'opération précédente ne doit pas forcément être terminée. Si, par contre, la préparation nécessite la présence des pièces sur la machine, le temps de préparation est intégré dans les temps opératoires.

Parallélisme : lorsque certaines machines peuvent réaliser les mêmes opérations, il faut déterminer, lors de l'ordonnancement, sur quelle machine sera faite chaque opération. Le problème est, alors, appelé problème hybride (*jobshop* hybride, *flow-shop* hybride).

Stocks intermédiaires : suivant la teneur des opérations à effectuer, il est possible de mettre des stocks intermédiaires entre les machines. De plus, pour les cas où ceci est envisageable, il faut aussi tenir compte de la capacité des stocks.

Système de transport : dans certains cas les systèmes de transport (robot, palans, chariot...) doivent être considérés comme des ressources à part entière car s'ils sont en petit nombre ils peuvent limiter les transferts. Il faut donc, dans ces cas, non seulement trouver l'ordre des tâches mais aussi ordonnancer les mouvements du (des) robot(s).

Indisponibilité des ressources : cela permet de prendre en compte les contraintes de type maintenance ou panne, selon que l'indisponibilité est prévue de manière certaine ou modélisée de manière probabiliste.

1.2.3 Spécificités des lignes de traitement de surface

Les contraintes, liées aux procédés utilisés sur les lignes de traitement de surface, nécessitent d'être prises en compte lors de l'élaboration de l'ordonnancement. Le calcul de cet ordonnancement, quand on tient compte des contraintes (fenêtres de temps pour les durées opératoires, disponibilité du robot et des cuves), est alors appelé, dans la littérature, *Hoist Scheduling Problem* [65][66].

1.2.3.1 Fenêtre de temps pour les durées opératoires

Chaque traitement est constitué de plusieurs immersions dans des cuves contenant des produits différents. La teneur de ces produits est donnée par les ingénieurs chimistes. La différence principale entre une ligne de type traitement de surface et une plus "classique" vient du fait que les durées de traitement sont bornées inférieurement et supérieurement. Dans l'exemple où le traitement est une attaque acide, il faut une durée minimale afin que le traitement soit effectué correctement, mais la pièce ne doit pas rester trop longtemps sous peine d'être détériorée et d'entraîner des défauts de qualité. C'est aussi le cas pour des traitements de dépôts d'argent ou d'or où il faut une durée minimale pour obtenir un dépôt suffisant mais aussi une durée maximale afin de ne pas avoir des coûts de matière trop importants. En revanche, pour les lignes de fabrication comme les lignes d'usinage, il y a des temps minimaux qui sont les temps d'opération. Une fois l'usinage terminé, la pièce peut rester dans le mandrin ou être mise en stock afin de libérer la machine.

Sur certains traitements, lors de la conception de la ligne, les ingénieurs chimistes proposent des concentrations possibles dans les cuves. Ceci entraînent une variation des temps de trempe en fonction des concentrations proposées. La personne chargée de l'ordonnancement utilise ensuite ces possibilités pour proposer une concentration ou un taux de produit actif dans les cuves concernées. Cette possibilité permet donc une marge de manœuvre pour le calcul de l'ordonnancement.

Les marges sur les temps d'immersion peuvent varier de 0% pour les dépôts de métaux précieux où les attaques acides jusqu'à des marges très grandes, que l'on pourra assimiler à des marges infinies, pour des opérations de rinçage où la pièce ne subit aucune dégradation lors de durées d'immersion très importantes. Cette contrainte sur les durées d'immersion est généralement appelée contrainte de fenêtre de temps (*time window constraint*).

Une autre contrainte vient s'ajouter à la contrainte de fenêtre de temps, c'est l'impossibilité d'attente entre deux opérations. En effet, contrairement aux productions de type usinage par exemple, dans les lignes de traitement de surface, il n'y a pas de possibilité d'avoir des stocks intermédiaires entre les cuves. De plus, un robot qui vient de récupérer une pièce ne peut pas attendre un événement (libération d'une cuve ou d'un rail) pour finir les déplacements. Donc, à partir du moment ou l'on décide de faire un changement de cuve pour un porteur, on ne peut pas interrompre l'opération (préemption interdite). Pour comprendre cette contrainte, on peut regarder le cas où un porteur sort d'un bain d'acide. Si on le sort du bain et que l'on doit attendre avant de l'immerger dans une cuve de rinçage, l'acide va encore attaquer la pièce et va la dégrader. Cette dégradation entraîne alors le rejet de la pièce en sortie de ligne, pour des raisons de qualité. Cette contrainte est appelée contrainte de sans-attente.

1.2.3.2 Disponibilité du robot

Nous avons vu que les porteurs ont une durée d'immersion bornée dans chacune des cuves, il faut donc qu'à la fin de chaque traitement un robot soit disponible afin de sortir le porteur de sa cuve pour éviter les détériorations ou les coûts supplémentaires éventuels.

Ce problème de disponibilité du robot ainsi que celui de la disponibilité des cuves nécessitent de gérer convenablement les déplacements du robot. Le HSP (*Hoist Scheduling Problem*) consiste à trouver l'ordre optimal d'entrée des porteurs sur la ligne, mais aussi les durées de trempe effectives pour chacune des opérations et les déplacements des robots avec leurs dates d'exécution.

1.2.3.3 Disponibilité des cuves

Comme il a été dit précédemment, les cuves peuvent recevoir un (pour des cuves mono-bac), ou plusieurs porteurs (pour des cuves multi-bacs). Il faut donc vérifier avant chaque déplacement de porteur que l'on va pouvoir déposer celui-ci dans la cuve. Il faut donc que la cuve soit déjà vide pour des cuves mono-bac ou qu'il reste un emplacement libre dans les cuves multi-bacs.

1.2.4 Représentation des résultats de l'ordonnancement

Dans la suite du document, il sera nécessaire de représenter les résultats de l'ordonnancement obtenu. Nous utiliserons deux types d'outils. L'une utilisant les diagrammes de Gantt et une autre plus spécifique à notre problème appelé représentation pyramidale.

1.2.4.1 Diagramme de Gantt

Nous proposons ici, une manière de construire un diagramme de Gantt pour représenter à la fois l'exécution des tâches et les activités du robot. On considère le robot et les cuves comme des ressources équivalentes. L'utilisation du robot est ainsi intégrée aux opérations correspondantes sur les cuves :

- l'opération de transfert de la cuve 0 (de chargement) est intégrée à l'opération dans la cuve 1,
- l'opération de transfert de la cuve i à $i+1$ est insérée dans l'opération de la cuve $i+1$,
- l'opération de transfert jusqu'à la cuve de déchargement est insérée dans la dernière opération.

Un exemple est donné sur la figure 1.7. Les opérations du robot sont représentées en gris foncé et les opérations sur les cuves en clair. Dans notre exemple, la gamme de fabrication de chaque pièce consiste à passer, successivement, dans toutes les cuves et dans l'ordre de disposition des cuves sur la ligne. Les temps de trempe peuvent être différents en fonction des produits à traiter.

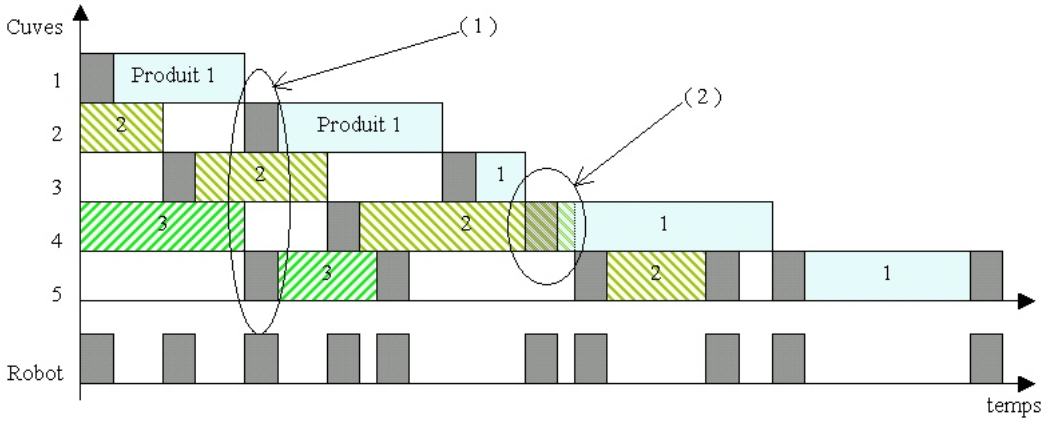

Figure 1.7 – Diagramme de Gantt

Cette méthode permet aussi de visualiser les différents conflits possibles, que ce soit de robot (1) ou de cuve (2). En effet, dans le cas (1) le robot doit, dans le même temps, déplacer le produit 1 de la cuve 1 à la cuve 2 et le produit 3 de la cuve 4 à la cuve 5, ce qui est impossible dans le cas d'une ligne mono-robot. Le conflit (2) signifie que le produit 1 est déplacé de la cuve 3 à la cuve 4, alors que le produit 2 y est encore, ce qui, dans le cas d'une cuve mono-bac, n'est pas permis.

1.2.4.2 Représentation pyramidale

Suivant le type d'ordonnancement, il peut être intéressant de ne prendre en compte que les mouvements du robot. Dans ce cas, on utilise généralement un chronogramme où l'axe des ordonnées est la position du robot et l'axe des abscisses le temps. Les mouvements du robot sous charge sont en trait continu, tandis que ceux à vide sont en trait pointillé. La figure 1.8 donne un exemple de chronogramme.

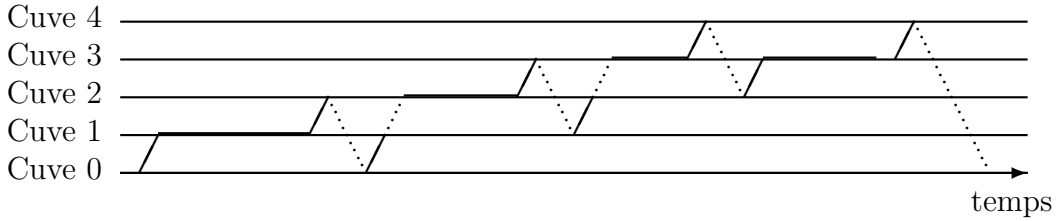

Figure 1.8 – Exemple de représentation sous forme de chronogramme

Si on se focalise essentiellement sur les mouvements du robot et qu'on ne représente pas les temps d'attente, on obtient un autre type de représentation que nous nommerons représentation pyramidale. La figure 1.9 donne un exemple de représentation pyramidale.

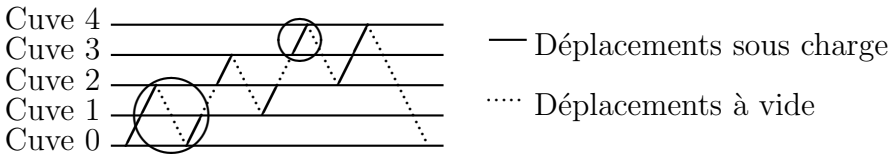

Figure 1.9 – Représentation d'une séquence de mouvements du robot

La figure 1.9 représente l'exemple suivant : soit une ligne à trois cuves, on considère qu'à l'état initial la ligne est vide. Le robot récupère un porteur dans la cuve de chargement et le transporte dans la cuve 1. Ensuite il attend que le produit soit entièrement traité avant de le déplacer dans la cuve 2. Ces deux activités sont représentées par un trait fort qui va de la cuve 0 à 1 puis de la cuve 1 à 2. Comme on ne s'occupe que des mouvements du robot, on ne représente pas les temps d'attente du robot au-dessus des cuves. L'état du système est donc, à ce moment là, le suivant : une pièce dans la cuve 2, aucune dans les cuves 1 et 3 et le robot au-dessus de la cuve 2.

Le robot retourne alors au-dessus de la cuve 0 afin de faire rentrer un nouveau porteur (porteur 2). Le déplacement à vide de la cuve 2 à la cuve 0 est alors représenté en pointillé (partie entourée). Le robot fait donc rentrer le porteur 2 dans la ligne puis retourne récupérer le porteur 1 dans la cuve 2 pour le transporter dans la cuve 3. Enfin il transporte le porteur 2 de la cuve 1 à 2, puis il sort le porteur 1 de la ligne (partie entourée) avant de transporter le porteur 2 de la cuve 2 à 3 et une fois le

traitement achevé, il vide la ligne en déplaçant le porteur 2 de la cuve 3 à la cuve de déchargement. Le système est alors à nouveau dans l'état initial.

Cette représentation permet, entre autre, de facilement voir les temps de trempe minimaux dans chaque cuve, en fonction des mouvements du robot. Sur la figure 1.10, la double flèche représente le temps de trempe minimal nécessaire à la réalisation du cycle pour la cuve 3, c'est à dire le temps que met le robot à faire tous les mouvements entre le moment où il dépose le porteur dans la cuve 3 et le moment où il le retire.

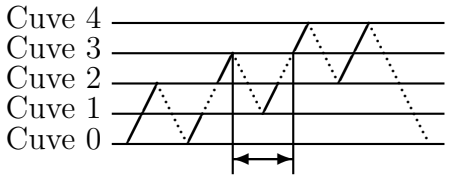

Figure 1.10 – Visualisation des temps de trempe minimaux

1.3 État de l'art

Les problèmes d'ordonnancement de la production couvrent un éventail de recherches important. Beaucoup d'ouvrages sont consacrés à ce domaine [6, 59, 50, 61].

1.3.1 Etat de l'art pour les problèmes de *flowshop* robotisé

Le *Hoist Scheduling Problem* se rapproche très fortement des problèmes d'ordonnancement des ateliers en lignes. Le problème du *flowshop* robotisé peut en effet être vu comme un cas particulier du *HSP* où toutes les fenêtres de temps seraient infinies. A l'inverse, le problème du *flowshop* robotisé sans attente peut être vu comme un *HSP* où les fenêtres de temps seraient nulles.

Le *flowshop* classique (minimisation de la dernière date de sortie), à deux machines sans moyen de transport, a été résolu de manière polynômial, dès les années 50 par Johnson [35]. Pour des cas plus contraints différentes heuristiques ont été proposées, comme par exemple avec dates d'arrivée [60, 38]. Pour un nombre de machines supérieur ou égal à trois Garey *et al.* [25] ont montré que le problème est NP-complet.

Dans la suite, nous traiterons simplement des cas du *flowshop* robotisé avec ou sans attente.

1.3.1.1 *Flowshop* robotisé avec marges infinies

Dans cette section, nous ne détaillerons pas toutes les facettes de ce problème. Pour plus de détails, Crama *et al.*[19] ont réalisé un état de l'art complet sur le *flowshop* robotisé (modèles, complexité et liens avec les autres problèmes d'ordonnancement). Hall *et al.* ont montré que le problème général est NP-difficile pour trois machines (cuves) et différents types de pièces [31]. Crama et van de Klundert [21] ont montré que le problème cyclique mono-produit est NP-complet au sens fort.

Dans le cas d'une production mono-produit cyclique, Sethi *et al.* [64] ont proposé un algorithme polynômial, pour le problème à deux machines, qui donne le 1-cycle ou cycle 1-périodique optimal. On définit par k-cycle un cycle durant lequel k porteurs rentrent sur la ligne et k porteurs en sortent, une définition plus détaillée sera donnée chapitre 2. Ils ont montré que pour trois machines le problème peut être résolu par énumération. De ces travaux, ils ont proposé une conjecture disant que les cycles optimaux sont des 1-cycles. Crama et Van de Klundert [20] ont ensuite généralisé à m machines en proposant un algorithme fortement polynomial en m, dans le cas où les temps de transport sont additifs. Cette conjecture a été prouvée, pour deux et trois machines, par Crama et van de Klundert [22] puis par Brauner et Finke [8]. Si les temps de transport sont euclidiens, Brauner *et al.* [11] ont montré que le problème était fortement NP-complet.

Enfin, Levner et Kats [47] ont, aussi, proposé un algorithme polynômial ($O(m^3)$) pour trouver le 1-cycle qui maximise le taux de sortie, dans le cas du *flowshop* robotisé. Kats et Levner [40] ont aussi proposé, pour le cas multi-robot, un algorithme polynômial ($O(m^3 \log m)$) qui donne le 1-cycle qui maximise le taux de sortie. Ils ont aussi proposé un algorithme fortement polynômial dans le cas du *flowshop* réentrant [39].

1.3.1.2 *Flowshop* robotisé sans attente

Le problème du *flowshop* sans attente est un problème qui est apparu dans les années 70. Reddi et Ramamoorthy [62] ont appliqué l'algorithme polynômial de Gilmore et Gomory [27], pour le problème à deux machines sans robot. Cette application permet de trouver la solution qui minimise la dernière date de sortie. Cet algorithme permet aussi, dans le cas cyclique multi-produit à deux machines, de trouver le cycle (séquence d'entrée des produits) qui minimise le temps de cycle. Sriskandarajah et Wagneur [68] ont proposé une heuristique utilisant cet algorithme afin de définir la séquence d'entrée optimale des produits mais aussi de déterminer les tailles des lots pour une production par batch.

D'un point de vue complexité, pour le *flowshop* sans attente, Röck [63] a étudié le problème qui consiste à trouver le cycle qui minimise la date de fin du dernier produit (C_{max}). Sur une ligne à trois cuves ou plus, sans moyen de transport et pour

une production multi-produit, il a montré que le problème est *NP*-complet. Hall et Sriskandarajah [32] ont proposé un état de l'art sur les problèmes d'ordonnancement de machines avec des contraintes de sans-attente et d'indisponibilité.

Dans le cas d'une production cyclique, Song *et al.* [67] ont montré que l'optimalité des 1-cycles proposée dans le cas de marge infinie n'est pas vérifiée pour le cas sans-attente. Agnetis [1] a proposé une conjecture sur le degré des cycles optimaux pour le problème du *flowshop* robotisé cyclique sans attente. Cette conjecture suggère que, pour une ligne à m machines, les cycles optimaux sont des k-cycles avec $k \leq m - 1$. Nous reviendrons en détail sur cette conjecture plus tard dans le mémoire (section 2.4). Kamoun *et al.* [37] ont proposé un algorithme permettant d'obtenir le cycle optimal pour une ligne à deux machines.

Hanen et Munier [33] ont proposé un algorithme polynomial permettant de trouver le 1-cycle qui minimise le temps de cycle. Enfin Che *et al.* [14] ont prouvé que trouver le 2-cycle qui minimise le temps de cycle peut se résoudre par un algorithme polynômial.

1.3.2 *Hoist Scheduling Problem*

Manier et Baptiste [56] puis Baptiste *et al.* [4] ont réalisé un état de l'art sur le HSP qui décrit les différentes méthodes de résolution avec leurs avantages et inconvénients. Si la séquence des pièces est déjà donnée, Lei [44] propose une méthode pour trouver les dates d'entrée sur la ligne.

Pour trouver les séquences, le problème peut être modélisé de différentes manières permettant une résolution exacte ou approchée :
– par programmation linéaire [67],
– par programmation logique sous contraintes (PLC) [5].

1.3.2.1 Problème cyclique

Le problème du HSP est un problème très complexe à résoudre. Il a été montré que trouver l'ordonnancement cyclique optimal d'une ligne comprenant un seul robot, pour une production mono-produit, fait partie de la classe des problèmes NP-complets, pour des temps de transport quelconques [45] et pour des temps de transport additifs et symétriques [10].

Il existe, pour résoudre ce problème statique, quatre familles de méthodes que l'on va décrire ci-dessous :

– par programmation linéaire [67],
– par programmation logique sous contraintes (PLC) [5],

– par des méthodes de type Branch & Bound [15, 16, 17, 46],
– par des algorithmes génétiques [48].

Programmation linéaire

Dans le cas d'une production mono-produit, Song *et al* [67] modélisent le problème par un programme linéaire mixte. Comme le modèle obtenu nécessite un nombre trop important de variables et de contraintes, la formulation sert alors à développer une heuristique (*EST : earliest start time*) pour générer une solution approchée au problème. Cette heuristique consiste à faire rentrer les produits au plus tôt, tout en évitant les conflits, jusqu'à l'apparition d'un cycle.

Programmation logique sous contraintes

La méthode par programmation logique sous contraintes (PLC), proposées par Baptiste *et al.* [5], consiste à définir, dans un premier temps, toutes les contraintes liées à la chaîne de production. Puis, dans un second temps, les contraintes sont exprimées mathématiquement en fonction des données et des inconnues. La PLC permet alors de résoudre le problème par une méthode de retour en arrière sur une arborescence.

La stratégie pour trouver la solution optimale consiste à maximiser la durée du cycle (T) puis de lancer le calcul d'une solution. Ensuite la valeur obtenue de T sert de borne maximale pour un nouveau calcul et ainsi de suite jusqu'à que ce que l'on n'ait plus de solution. Cette méthode permet non seulement de retrouver le 1-cycle optimal du benchmark proposé par Phillips et Unger [58] mais aussi de trouver trois solutions équivalentes avec des temps de calcul faibles.

Branch and Bound

Chen *et al.*[15, 16] ont proposé un algorithme de type *Branch and Bound* qui permet d'obtenir le 1-cycle optimal pour le problème cyclique mono-produit. La borne utilisée dans l'algorithme est, dans ce cas, calculée grâce à un programme linéaire.

Dans le cas cyclique multi-produit, Varnier et Jeunhomme [72] ont développé une méthode de type branch and bound qui parcours un arbre où toutes les combinaisons sont proposées.

Algorithme génétique

Les algorithmes génétiques ont été introduits par Holland [34], puis élargis à un grand nombre de problèmes [28]. L'utilisation de ces algorithmes, proposée par Lim [48], permet de trouver un cycle proche de l'optimum à partir d'un ensemble de solutions

existantes (population initiale). Pour chaque solution on crée une force (*fitness*) qui reflète sa qualité, un sous-ensemble de solutions est sélectionné (en fonction de leur force) pour se reproduire. La descendance est obtenue avec des opérateurs (*LOX line order crossover, mutation*), certains parents sont éliminés (en fonction de la fonction à optimiser) pour être remplacés par les enfants. On recommence ceci jusqu'à la fin de l'itération ou jusqu'à ce qu'aucun enfant ne soit meilleur que les parents.

1.3.2.2 Problème dynamique

Lorsque la demande est incertaine, il est possible de modéliser les événements aléatoires par des lois stochastiques. Fleury *et al.* [23, 24] se sont intéressés au *HSP* stochastique (SHPS) en proposant une méta-heuristique pour trouver une solution avec comme critère la minimisation du *makespan*. Cette heuristique a été mise en place lorsque les conséquences des événements aléatoires sont faibles.

En ce qui concerne le problème dynamique, différents modes de pilotage ont été proposés pour le résoudre :

– périodique et prédictif ;
– réactif sans prévision ;
– réactif avec prévision.

Pilotage périodique

Le pilotage périodique est utilisé pour des travaux avec un grand horizon de temps (grande série, ou production par campagne). Il consiste à déterminer le cycle optimum pour une suite de traitements. Ce cycle est obtenu en minimisant le rapport (durée du cycle / nombre de porteurs introduits dans le cycle) avec les outils de type "programmation par contraintes". L'inconvénient d'un tel mode de pilotage vient des difficultés d'évolution lors d'un changement de gammes ou de pièces. Les méthodes utilisées dans ces cas sont, soit de vider l'atelier avant de lancer la nouvelle gamme, soit d'insérer le nouveau cycle en trois temps [70] :

– dégradation de l'ancien cycle ;
– insertion du nouveau cycle dégradé dans l'ancien dégradé ;
– dès que l'ancien cycle est terminé revenir au nouveau cycle optimal.

Dans le cas où il faut insérer une nouvelle opération dans un cycle sans en changer la séquence, une technique a été développée en utilisant la théorie des graphes [3].

Pilotage réactif sans prévision

Le pilotage réactif sans prévision est le suivant : après chaque opération de transport on recalcule les nouveaux mouvements du robot. Les temps de calcul devant être faibles pour ne pas retarder le robot, on emploie des heuristiques pour trouver les nouveaux mouvements. Le choix des heuristiques à utiliser est donné par un système expert basé sur la connaissance d'un expert humain. C'est à dire que, suivant la situation de l'atelier, on choisit l'heuristique qui donne la meilleure solution, le choix de cette heuristique est déterminé au préalable par simulation [73]. L'inconvénient d'une telle méthode vient surtout du fait qu'il est très dur de respecter les bornes. Chauvet *et al.* [13] ont proposé un algorithme dynamique, qui garantit la date de sortie minimale pour chaque nouveau porteur. Dans ce même article, les auteurs ont montré comment généraliser cet algorithme au cas multi-robot.

Pilotage réactif avec prévision

Le pilotage réactif avec prévision calcule, à chaque arrivée de porteur, un nouvel ordonnancement en tenant compte de ce qui a été déjà fait et ce qui doit être fait (position des porteurs sur la ligne, gamme des porteurs...). Ce pilotage a pour avantage de respecter les bornes mais aussi d'être peu sensible aux perturbations. L'inconvénient, par contre, est qu'il ne permet pas une optimisation sur le long terme. Pour calculer le nouvel ordonnancement quatre heuristiques sont possibles :

- **l'heuristique simple** [75] : elle regarde si il est possible de faire rentrer un nouveau porteur sur la ligne sans risque de conflits (de cuves ou de robot). Dans le cas d'une réponse positive, elle lance le nouveau porteur, sinon elle décale son entrée sur la ligne. Cette solution a pour particularité de ne pas tenir compte des tolérances sur les durées de séjour dans les cuves. Elle est efficace dans les problèmes où la variance des durées de trempe est faible, le robot rapide et le rapport (moyenne des durées de trempe / vitesse du robot) fort.

- **l'heuristique avec conservation de l'ordonnancement** [75] : cette heuristique consiste à regarder si la tolérance sur le nouveau porteur au niveau du conflit ne permet pas de le supprimer ou de réduire le temps d'attente en entrée. Le fait de jouer uniquement sur les tolérances du nouveau porteur limite l'utilisation de cette méthode aux cas où le robot est rapide et la tolérance sur le porteur en entrée est large.

- **l'heuristique avec conservation de la séquence** [74] : on utilise cette fois les tolérances des deux porteurs en conflit pour voir si, en modifiant les durées de trempe, on peut régler ces conflits. Par contre, cette méthode ne tient pas compte du fait qu'un décalage d'un porteur sur la ligne puisse entraîner un nouveau conflit. Cette heuristique est efficace quand le rapport (moyenne des durées de trempe / vitesse du robot) est intermédiaire, c'est à dire lorsque l'on ne peut pas négliger les temps de transport par rapport aux temps de trempe et inversement.

– **l'heuristique sans conservation** [26] : c'est une méthode de type "*procédure par séparation et évaluation*" où toutes les tolérances sont utilisées pour essayer de résoudre le problème. L'heuristique consiste à s'arrêter dès l'obtention d'une solution réalisable.

Procédure par séparation et évaluation

Une autre méthode pour régler les conflits (ou problème local) a été développée par Lamothe [42]. Elle consiste à modéliser le problème par contraintes puis le résoudre de manière arborescente avec une Procédure par Séparation et Evaluation Progressive (*PSEP*) en profondeur. Pour limiter les temps de calcul, l'auteur met en place des informations d'inconsistance (*Nogood*) et de mémorisation des raisonnements déjà réalisés (*dynamic backtracking*) ce qui permet de ne pas tout recalculer à chaque problème local.

Algorithme évolutionniste

Un algorithme évolutionniste a été mis au point pour résoudre le cas du problème dynamique [7]. Cet algorithme s'appuie sur la méthode génétique décrite précédemment. Il consiste à calculer, à chaque arrivée de nouveaux porteurs, un nouvel ordonnancement en tenant compte des travaux déjà effectués (seuil S1) et des temps de calcul (seuil S2). Pour vérifier la validité des solutions obtenues, un calcul de plus long chemin dans un graphe est utilisé. Une fois une solution trouvée, elle sert d'ordonnancement partiel pour l'itération suivante. Et ainsi de suite jusqu'à l'ordonnancement du dernier travail.

1.3.2.3 Généralisation

Les problèmes soulevés précédemment répondent au cas d'un atelier mono-robot, mono-tronçon et avec des cuves mono-bac. Il faut alors regarder comment appliquer ces méthodes pour des ateliers comportant plusieurs robots, des cuves multi-bac...

Dans le cas multi-robots, Armstrong *et al.* [2] ont proposé un travail sur le calcul du nombre de transporteurs. Une solution est de donner des règles de priorité entre les robots, il en existe plusieurs types :

– **Private turf** : chaque robot se voit attribuer une zone de l'atelier. Dans le cas de deux robots [46], on partage l'atelier en deux : les cuves 0 à V pour le robot 1 et les cuves V à m pour le robot 2. On calcule séparément les cycles optimaux puis on compare leur durées. Si elles sont différentes on fait varier V et on recommence jusqu'à avoir des résultats cohérents.

Varnier *et al.* [71] puis Manier *et al.* [57] ont proposé une méthode par branch and bound pour résoudre le problème avec un nombre quelconque de robots.

– **Overlapping turf** : les bornes des différentes zones sont révisées fréquemment.

– **Nearest hoist first (NHF)** : le robot le plus proche est prioritaire.

– **Modified NHF** : le robot le plus à gauche est affecté.

Il suffit alors d'utiliser un algorithme qui permet de déterminer le robot à affecter puis d'utiliser une des méthodes précédentes pour trouver l'ordonnancement.

Pour les problèmes où certaines cuves sont particulières, notamment dans le cas de cuves multi-bac un algorithme basé sur la PLC permet de trouver le cycle optimal [51]. Liu *et al.* [49] ont proposé une méthode par programmation linéaire mixte pour résoudre le cas multi-bac et lorsque la gamme doit passer plusieurs fois par une même cuve.

Quand la production est dynamique, Lamothe et Delmas [43] proposent une méthode consistant à distinguer des horizons de temps particuliers pour l'ordonnancement des cuves multi-bac.

D'autres types de travaux sont effectués sur le HSP comme les problèmes d'implantation de ligne [30]. La création de modèle objet [29], ou graphiques type réseaux de Petri p-temporels [41]. Ces recherches s'attachent à regarder la commande du système d'un point de vue robustesse [18]. Enfin, les problèmes environnementaux commencent à être pris en compte lors de l'ordonnancement des ateliers [69]. Le problème consiste à partir des solutions de l'ordonnancement réalisables, de déterminer laquelle est la mieux adaptée aux contraintes environnementales qui ne peuvent être prises en compte lors du calcul de l'ordonnancement.

Chapitre 2

Présentation du problème et notations

Dans la suite de ce mémoire, nous nous intéressons à un cas particulier du *HSP*. Nous faisons l'hypothèse d'une production par campagne, c'est à dire que la ligne doit traiter un grand nombre de produits identiques pendant une période suffisamment longue. Dans la section 3.2, nous traiterons un problème plus large, puisqu'il s'agit du problème multi-produit. Trouver les mouvements du robot qui optimisent la productivité revient alors à trouver une solution cyclique qui sera répétée. Nous considérerons que la campagne est suffisamment longue pour négliger les périodes transitoires (mise en route de la ligne, passage d'un cycle à l'autre...).

Nous considérons également que la gamme du produit consiste à traiter les produits successivement dans toutes les cuves (1,2,3...m) avant de sortir le produit par la cuve de déchargement ($m+1$). Ces conditions correspondent donc au problème du *flowshop*.

Lors de l'étude de lignes de traitement de surface industrielles, nous nous sommes aperçu que les traitements ont souvent des marges très faibles (attaque acide, dorure...), ou très importantes (rinçage, dégraissage...). Ce qui nous conduit à faire l'hypothèse que les marges sont nulles ou infinies. Cette hypothèse semble réaliste pour aborder plus simplement des cas réels. De plus, pour des contraintes de qualité, il est souvent préférable que les durées de trempe effectives dans une cuve soient égales pour tous les produits.

Dans ce chapitre nous précisons les notations utilisées tout au long du mémoire. Nous définirons, par la suite, les notions de cycle de production et leurs propriétés (section 2.2), de graphes d'état et de *line-graph* (section 2.3). Enfin, nous définirons les objectifs des travaux qui seront détaillés dans les chapitres suivants (section 2.4). Pour l'étude du cas où plusieurs types de pièces sont produits, nous introduirons d'autres notations dans le chapitre suivant (section 3.2).

2.1 Notations

L'une des contraintes fortes du *HSP* est l'existence des fenêtres de temps. Chaque produit doit donc rester un temps p_i dans la cuve T_i, ce temps de process p_i se trouvant dans un intervalle défini. Le temps minimum de traitement d'un produit (borne inférieure de la fenêtre de temps) dans la cuve T_i est noté l_i (pour *lower bound*) alors que le temps de trempe maximum (borne supérieure de la fenêtre de temps) est noté u_i(pour *upper bound*). La durée de trempe effective p_i doit donc se trouver dans l'intervalle $[l_i; u_i]$ et elle constitue une variable de décision pour le problème.

Excepté pour la section 3.2, nous considérerons que les temps de transport pour aller d'une cuve T_i à une cuve voisine et de δ unités de temps. Dans cette étude, nous négligerons les temps de chargement et de déchargement. Donc le temps pour passer d'une cuve à une autre est le même que le robot transporte un porteur ou se déplace à vide. Ces temps de transport seront supposés additifs, c'est à dire qu'il faudra $|i - j|\delta$ unités de temps au robot pour passer de la cuve T_i à la cuve T_j.

Récapitulatif des données du problème :

m	nombre de cuves de traitement
T_i	cuve i (les cuves étant numérotées de 1 à m)
T_0	cuve de chargement
T_{m+1}	cuve de déchargement
l_i	durée minimale de traitement dans la cuve i
u_i	durée maximale de traitement dans la cuve i
δ	durée du déplacement du robot entre deux cuves consécutives

2.2 Activités et cycles de production

Les mouvements du robot sont décrits en terme d'activités. L'activité A_i ($i = 0, 1 \ldots m$) qui sera souvent notée i pour des raisons de simplification, est représentée figure 2.1. L'activité A_i est constituée des mouvements suivants :

– le robot récupère un porteur dans la cuve T_i ;

– le robot se déplace de T_i jusqu'à T_{i+1} ;

– le robot dépose le porteur dans T_{i+1}.

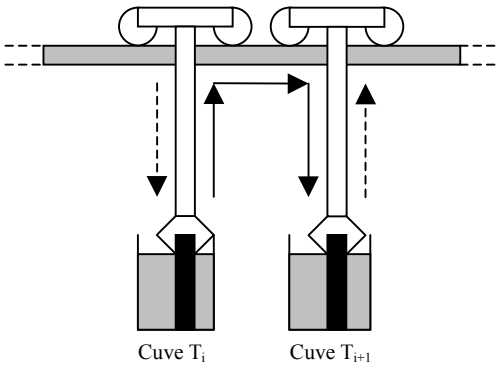

Figure 2.1 – Ensemble des mouvements constituant l'activité A_i

Une activité représente donc une séquence de mouvements du robot chargé. Mais, si l'on définit une séquence d'activité, par exemple la séquence d'activités $(1, 3, 2)$, on obtient aussi les mouvements à vide du robot. Pour l'exemple $1, 3, 2$, cela veut dire que :

- le robot déplace un porteur de la cuve T_1 à la cuve T_2,
- le robot se déplace à vide de la cuve T_2 à la cuve T_3,
- le robot déplace un porteur de la cuve T_3 à la cuve T_4,
- le robot se déplace à vide de la cuve T_4 à la cuve T_2,
- le robot déplace le porteur de la cuve T_2 à la cuve T_3.

Dans notre cas d'étude, certaines séquences peuvent ne pas être réalisables. En effet, entre deux activités A_α consécutives, il doit y avoir exactement une occurrence de l'activité $A_{\alpha-1}$ (présence d'un porteur dans la cuve T_α). Il doit aussi y avoir exactement une occurrence de l'activité $A_{\alpha+1}$ (libération de la cuve $T_{\alpha+1}$).

Enfin certaines séquences sont impossibles du fait des contraintes sur le temps de process. Si l'on se place dans le cas où tous les temps de transports sont fixes et toutes les fenêtres nulles, alors, entre l'activité $A_{\alpha-1}$ et l'activité A_α suivante il ne peut pas y avoir la séquence A_β,$A_{\beta+1}$ (figure 2.2).

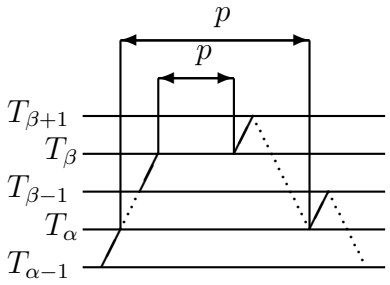

Figure 2.2 – Impossibilité de la séquence $\alpha - 1, \beta, \beta + 1, \alpha$

Dans la suite de ce document, nous considérerons les mouvements cycliques du robot. Nous définissons alors les k-cycles de la manière suivante :

Définition 2 *Un k-cycle (ou cycle k-périodique) C_k est une séquence d'activités dont chaque activité est réalisée exactement k fois et, entre deux occurrences consécutives (au sens cyclique) de l'activité A_i, il y a exactement une occurrence de A_{i-1} et exactement une occurrence de A_{i+1} pour (i=0,1 ... m-1).*

De cette définition nous pouvons déduire que, lors de l'exécution d'un k-cycle, il y a exactement k porteurs qui entrent sur la ligne (k activités 0) et k porteurs qui en sortent (k activités m).

Définition 3 *k est appelé degré du cycle C_k.*

La figure 2.3 représente sous forme pyramidale, le 2-cycle (02132031) pour une ligne à trois machines.

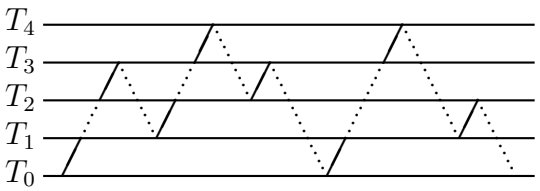

Figure 2.3 – Exemple de 2-cycle : 02132031

La longueur du cycle est noté $T(C_k)$, c'est-à-dire le temps total d'exécution du cycle C_k. Le temps de cycle (relatif) est défini comme la longueur du cycle divisée par le

degré du cycle. Dans l'hypothèse où les temps d'attente ne se répètent pas à l'infinie, le temps de cycle est alors la durée moyenne à l'infinie.

Pour calculer la longueur du cycle, il faut ajouter tous les temps de transport aux temps d'attente du robot. En effet, le robot peut attendre au-dessus d'une cuve que le traitement soit fini.

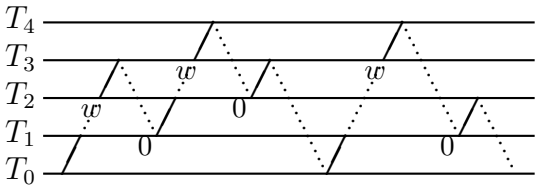

Figure 2.4 – Cycle (02132031) avec les temps d'attente

Sur l'exemple (figure 2.4), la longueur du cycle est donc de $24\delta + 3w$. Soit w le temps d'attente qui peut valoir $\max(0, l - 4\delta)$ dans le cas de fenêtres de temps infinies ou $l - 4\delta$ pour des fenêtres nulles. Ce qui donne un temps de cycle de $12\delta + 3w/2$ pour des fenêtres infinies et $3l/2 + 6\delta$ pour des fenêtres nulles.

Le problème d'ordonnancement, dans le cas cyclique, consiste à trouver le cycle qui minimise son temps de cycle. Ceci nous amène à définir la relation de dominance entre deux ensembles de cycles comme suit :

Définition 4 *Soient S et S' deux ensembles de cycles. S est dit dominant par rapport à S', si pour toute instance, la propriété suivante est vérifiée : pour tout k'-cycle $C_{k'}$ de S', il existe un k-cycle C_k dans S qui vérifie $\frac{T(C_k)}{k} \leq \frac{T(C_{k'})}{k'}$.*

2.3 Graphe d'état et line-graph

Pour une ligne à m cuves, l'état du système peut être représenté par un vecteur de dimension m où la $i^{\text{ème}}$ composante vaut 0 si la cuve est vide et 1 s'il y a un porteur qui est en traitement. Le graphe d'état de la ligne, G_m, est défini par ses nœuds qui sont les états du système et les arcs qui sont les activités du robot qui permettent les passages d'un état à un autre. Si on considère une ligne à trois cuves dans l'état où un porteur est traité dans la cuve T_1 et un porteur est traité dans la cuve T_3, l'état du système peut donc être modélisé par le vecteur (1,0,1). De cet état le robot peut :

– soit déplacer le porteur qui est dans la cuve T_1 jusque dans la cuve T_2 en effectuant l'activité A_1 pour arriver à l'état (0,1,1) ;

– soit déplacer le porteur qui est dans la cuve T_3 jusque dans la cuve de déchargement en effectuant l'activité A_3 pour arriver à l'état (1,0,0).

Nous pouvons donc en déduire que, dans le graphe d'état, il y a un arc qui part du sommet (1,0,1) vers le sommet (0,1,1) et un arc qui va vers le sommet (1,0,0) (figure 2.5).

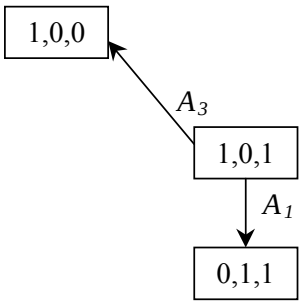

Figure 2.5 – Construction du graphe d'état pour une ligne comportant trois cuves

Si nous faisons la même chose pour tous les états possibles du système, le graphe d'état complet représenté figure 2.6 est obtenu.

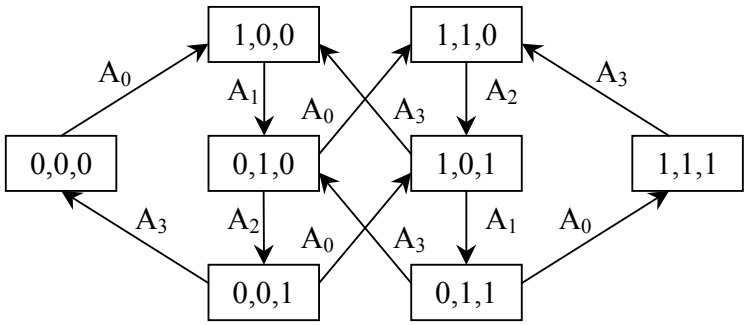

Figure 2.6 – Graphe d'état pour une ligne comportant trois cuves (G_3)

A partir du graphe d'état nous pouvons déduire le *line-graph* associé LG_m de G_m qui est construit de la manière suivante :

– les nœuds de LG_m sont les arcs de G_m ;
– (a, a') est un arc LG_m si et seulement si il existe un nœud v dans G_m dont l'extrémité finale est a et l'extrémité initiale est a'.

Dans le cas d'une ligne à trois cuves, si l'on part du sommet (0,0,0), on peut effectuer uniquement l'activité A_0, puis l'activité A_1 et ensuite soit l'activité A_0 de nouveau,

soit l'activité A_2. Ceci se traduira dans le *line-graph* par l'ensemble des sommets et des arcs représentés figure 2.7.

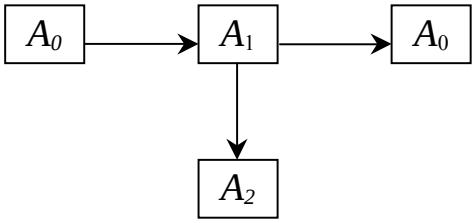

Figure 2.7 – Construction du line-graph LG_3

La figure 2.8 représente le *line-graph* LG_3 complet du graphe d'état G_3.

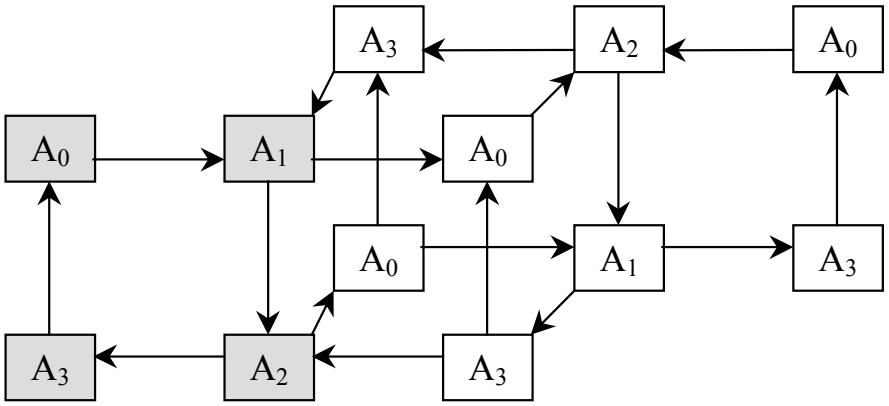

Figure 2.8 – Line-Graph LG_3 du graphe d'état pour un tronçon de trois cuves

L'intérêt du *line-graph* vient du fait que, à tout cycle de production, correspond un circuit unique dans le graphe, et que ce graphe tient compte uniquement du robot et non plus de l'état de la ligne. En effet, si l'on prend par exemple le 2-cycle 01021323, le circuit correspondant est représenté en gras sur la figure 2.9.

De plus l'inverse est aussi vrai, tout circuit dans le graphe correspond à un cycle sur la ligne de production ce qui n'est pas vrai avec les graphes d'état. Donc si l'on doit chercher les cycles réalisables il suffit de regarder les circuits dans le *line-graph*.

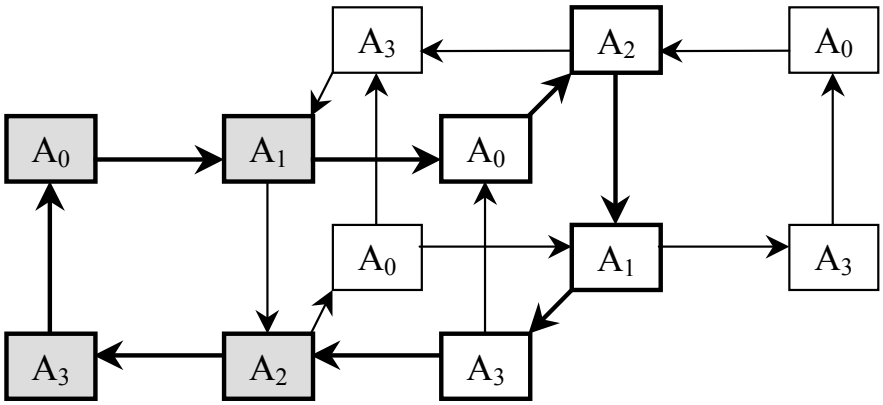

Figure 2.9 – Circuit correspondant au cycle 01021323 dans le line-graph LG_3

A partir de là, Brauner et Finke [9] ont prouvé la propriété suivante qui sera utilisée dans un certain nombre de démonstrations.

Nous notons E_m, l'ensemble des sommets qui compose le cycle $(0, 1, 2 \dots m)$ (sommets gris sur la figure 2.8).

Propriété 1 *Pour toute instance I, soit ℓ la plus petite valeur telle qu'il existe un ℓ-cycle C_ℓ optimal. Alors C_ℓ passe au maximum une fois par chaque sommet de E_m.*

Cette propriété est liée au fait que, lors de l'activité réalisée sur l'un des nœuds de E_m, un seul porteur est sur la ligne. Par conséquent, le cycle est une concaténation de cycles de degré inférieur, et la longueur du cycle complet est la somme des longueurs des sous-cycles. On dit alors que les temps de cycles sont additifs.

Récapitulatif des notations :

A_i ou i	activité du robot qui permet de déplacer une pièce de la cuve T_i à la cuve T_{i+1}
C_k	cycle de degré k
$T(C_k)$	Temps de cycle du k-cycle C_k
$\frac{T(C_k)}{k}$	Temps de cycle relatif du k-cycle C_k
G_m	Graphe d'état pour une ligne à m cuves
LG_m	*Line-graph* associé au graphe d'état G_m
E_m	Ensemble des activités où le robot transporte l'unique porteur présent sur la ligne

Enfin la notation des cycles peut être très longue pour les cycles comportant beaucoup d'activités. Pour simplifier les notations nous utiliserons la notation $\prod$ comme la concaténation d'activités ou de séquences d'activités. Par exemple, pour une ligne à quatre cuves, le cycle constitué des activités (4,3,2,1,0) sera noté $\prod_{i=0}^{4}(4-i)$.

Cette notation permet de décrire, de manière condensée, des séquences d'activités. Prenons l'exemple où la ligne est vide et que l'on veuille faire rentrer quatre produits consécutivement. La figure 2.10 représente cette suite d'activités. Si nous notons toutes les activités la notation est alors (0102103210). Si nous utilisons la notation $\prod$ cela donne la formulation suivante : $\prod_{i=1}^{4}\left(\prod_{j=1}^{i}(i-j)\right)$.

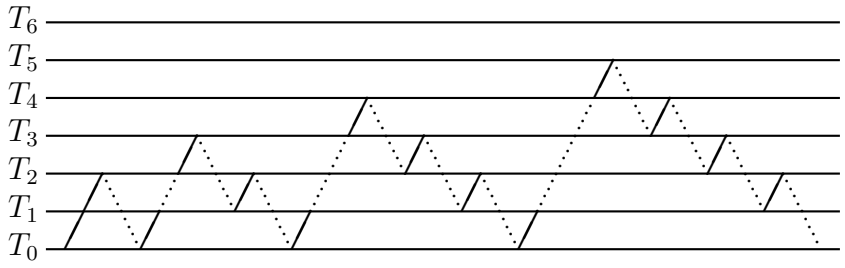

Figure 2.10 – Séquence $\prod_{i=1}^{4}\left(0\prod_{j=0}^{i-1}(i-j)\right)$=(01021032104321)

2.4 Objectif des travaux

Sethi *et al.* [64] ont développé un algorithme polynomial permettant de trouver le meilleur 1-cycle pour le *flowshop* robotisé (marges infinies) à deux et trois machines. Ils ont posé une conjecture encore ouverte dans laquelle les cycles optimaux sont des 1-cycles. Un algorithme en $O(m^3)$ a été proposé par Crama et Van de Klundert [20]. Cet algorithme permet de trouver le 1-cycle, dont le temps de cycle est minimal, pour le problème du *flowshop* robotisé (marges infinies) à m machines.

Par la suite, il a été montré que cette conjecture ne pouvait pas s'appliquer pour le cas sans attente. Agnetis [1] a donc proposé la conjecture définissant les cycles optimaux parmi les k-cycles ($k \leq m-1$). Dans sa publication, il a prouvé cette conjecture pour $m \leq 3$.

Dans ce mémoire nous proposons la conjecture suivante, qui généralise au cas où les fenêtres de temps sont nulles ou infinies.

Conjecture 1 *Les 1-, 2 ... k-cycles $k \leq m-1$ sont dominants pour le problème du* HSP*, dans le cadre d'une production mono-produit, avec des bornes supérieures infinies ou nulles.*

Cette conjecture signifie que pour trouver les cycles optimaux, dans une ligne à m cuves, il suffit de regarder parmi les 1-,2-,... $m-1$-cycles.

Pour $m = 2$, nous montrerons, dans le chapitre 3, que cette conjecture est vérifiée pour le cas où les fenêtres de temps sont quelconques. Pour cela, nous déterminerons

les cycles optimaux en fonction des instances. Nous montrerons ensuite que cette conjecture ne peux pas s'appliquer au cas multi-produit

Dans le chapitre 4, pour $m = 3$, nous prouverons cette conjecture pour le cas où les fenêtres de temps sont quelconques, mais lorsque les temps de trempe minimaux sont égaux. Pour cela, nous déterminerons les cycles optimaux en fonction des instances.

Ensuite, dans le chapitre 5, le cas équilibré sans attente sur toutes les cuves pour $m = 4$ sera étudié, c'est à dire lorsque toutes les fenêtres de temps sont nulles et les temps de trempe égaux. Dans ce cas, nous montrerons qu'il existe des 1-, 2- et 3-cycles optimaux.

Enfin, dans le chapitre 6, une conjecture sur les cycles optimaux, pour le cas sans attente équilibré et m quelconque, sera proposée. Cette conjecture donnera les cycles optimaux en fonctions des instances du problème (nombre de cuves, temps de process et temps de transport). Nous prouverons cette conjecture pour certaines configurations, et donnerons les cas pour lesquels la conjecture reste ouverte.

Chapitre 3

HSP pour une ligne à deux cuves

Dans ce chapitre, nous étudierons deux exemples de production cyclique.

Dans un premier temps, lorsque la production est mono-produit (section 3.1), c'est à dire lorsque la ligne produit un seul type de pièce, nous verrons quels sont les cycles optimaux en fonction des données du problème. Nous étudierons le cas où les marges sont nulles ou infinies, puis le cas où les marges sont quelconques.

Dans un deuxième temps, pour une production toujours cyclique, mais cette fois, quand la ligne doit produire différents types de pièces (section 3.2), nous verrons comment utiliser les résultats donnés par l'algorithme de Gilmore et Gomory, pour le cas sans robot, afin de les adapter au HSP.

3.1 Production mono-produit

Dans cette section, nous commençons par étudier le cas où le temps de trempe minimal dans la cuve T_1 (resp T_2) est l_1 (resp l_2) et les fenêtres de temps sont nulles ou infinies. Nous regarderons à partir du *line-graph*, quels sont les cycles réalisables. Une fois ces cycles déterminés, nous identifierons, en fonctions des valeurs de l_1, de l_2 et des fenêtres de temps, les cycles dominants.

La figure 3.1 représente le *line-graph* LG_2 pour une ligne à deux cuves. Nous pouvons remarquer qu'il n'y a qu'un seul nœud A_1 qui, qui plus est, fait partie de l'ensemble E_2. Donc tous les k-cycles doivent passer par ce sommet k fois. Ainsi, la propriété 1, présentée dans le chapitre précédent, implique que seuls les 1-cycles peuvent être optimaux. Ceci prouve la conjecture 1, présentée dans le chapitre 2, pour le cas d'une ligne à deux cuves.

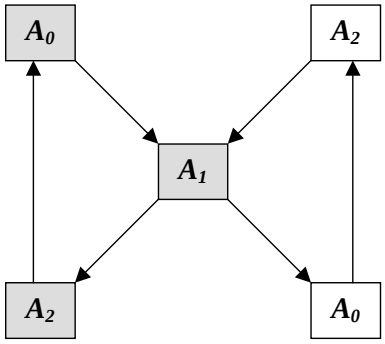

Figure 3.1 – Line-graph LG_2

Le *line-graph* LG_2 indique qu'il y a deux 1-cycles réalisables, $(0, 1, 2)$ et $(0, 2, 1)$.

Durant l'exécution du cycle $(0, 1, 2)$ le robot effectue les opérations suivantes (la valeur entre crochets indique la durée de chaque opération) :

– transport et dépôt d'un nouveau porteur de la cuve T_0 dans la cuve T_1 $[\delta]$;
– attente jusqu'à la fin du traitement au dessus de T_1 $[l_1]$;
– transport et dépôt du porteur dans la cuve T_2 $[\delta]$;
– attente jusqu'à la fin du traitement au dessus de T_2 $[l_2]$;
– transport et déchargement du porteur dans la cuve T_3 $[\delta]$;
– retour de la cuve T_3 vers la cuve T_0 $[3\delta]$.

Ce cycle, appelé cycle identité (Id=$\prod_{i=0}^{m} i$) pour m machines, est toujours réalisable quels que soient les temps de trempe et les marges. Il a un temps de cycle de l_1+l_2+6δ (somme des temps d'opération du robot indiqués ci-dessus entre crochets).

Considérons maintenant le cycle $(0, 2, 1)$. Ce cycle est réalisable si le robot peut laisser un porteur plus de 4δ unités de temps dans chaque cuve, c'est à dire que les durées de trempe ont une marge infinie ou alors que les durées minimales sont supérieures ou égales à 4δ. En effet, entre le moment où un porteur est déposé dans T_1 et le moment où le robot va revenir le chercher, celui-ci doit

– se déplacer de T_1 vers T_2 $[\delta]$;
– exécuter l'activité A_2 $[\delta]$;
– revenir en T_1 $[2\delta]$.

Le même raisonnement est applicable pour le traitement dans la cuve T_2.

Le temps de cycle de $(0, 2, 1)$ est donc de $(\max(4\delta, l_1, l_2) + 4\delta)$ unités de temps. Ce cycle domine donc le cycle $(0, 1, 2)$ dès qu'il est réalisable et que $l_1 + l_2 \geq 2\delta$.

Le tableau 3.1 donne le cycle optimal quelle que soit l'instance. Chaque ligne représente une configuration de l'atelier en fonction des marges sur les traitements. Par exemple, la configuration $(0, \infty)$ signifie que le temps de trempe est fixe sur T_1 ($l_1 = u_1$) et le temps de trempe l_2 dans la cuve T_2 est dans l'intervalle $[l_2, +\infty[$.

Tableau 3.1 – Cycles optimaux pour une ligne de deux cuves

<table>
<tr><td rowspan="3"></td><td rowspan="3">$l_1 + l_2 < 2\delta$</td><td colspan="4">$l_1 + l_2 \geq 2\delta$</td></tr>
<tr><td colspan="2">$l_1 < 4\delta$</td><td colspan="2">$l_1 \geq 4\delta$</td></tr>
<tr><td>$l_2 < 4\delta$</td><td>$l_2 \geq 4\delta$</td><td>$l_2 < 4\delta$</td><td>$l_2 \geq 4\delta$</td></tr>
<tr><td>$0, 0$</td><td colspan="4" rowspan="4">$0, 1, 2$</td><td rowspan="3">$0, 2, 1$</td></tr>
<tr><td>$0, \infty$</td></tr>
<tr><td>$\infty, 0$</td><td>$0, 1, 2$</td></tr>
<tr><td>∞, ∞</td></tr>
</table>

Cette méthode peut facilement être étendue au *HSP* cyclique classique où les fenêtres de temps sont quelconques. Le temps de cycle de (0,2,1) est $\max(4\delta, l_1, l_2) + 4\delta$ donc il est supérieur à 8δ. Comme le temps de cycle de (0,1,2) est de $l_1 + l_2 + 6\delta$, si $l_1 + l_2 < 2\delta$ le cycle Id est dominant quelles que soient les bornes u_1 et u_2. De plus, le cycle (0,2,1) est réalisable uniquement si les bornes u_1 et u_2 sont supérieures à 4δ. Nous pouvons donc en déduire que le cycle (0,2,1) est optimal pour $l_1 + l_2 \geq 2\delta$ et u_1 et u_2 supérieures à 4δ.

Le tableau 3.2 montre les cycles optimaux en fonction des instances du problème.

Tableau 3.2 – Cycles optimaux pour le HSP dans une ligne de deux cuves

<table>
<tr><td rowspan="3"></td><td rowspan="3">$l_1 + l_2 < 2\delta$</td><td colspan="4">$l_1 + l_2 \geq 2\delta$</td></tr>
<tr><td colspan="2">$l_1 < 4\delta$</td><td colspan="2">$l_1 \geq 4\delta$</td></tr>
<tr><td>$l_2 < 4\delta$</td><td>$l_2 \geq 4\delta$</td><td>$l_2 < 4\delta$</td><td>$l_2 \geq 4\delta$</td></tr>
<tr><td>$u_1 < 4\delta$</td><td colspan="3" rowspan="3">0, 1, 2</td><td colspan="2">impossible</td></tr>
<tr><td>$u_2 < 4\delta$</td><td>impossible</td><td>0, 1, 2</td></tr>
<tr><td>$u_1 \geq 4\delta$, $u_2 \geq 4\delta$</td><td colspan="3">0, 2, 1</td></tr>
</table>

3.2 Production multi-produit

Nous allons maintenant étudier le cas d'une production multi-produit, pour une ligne à deux cuves. La ligne doit donc produire n types de produits dans des quantités égales. Le but de l'ordonnancement est alors de trouver, d'une part le séquencement des produits, c'est à dire l'ordre de passage des produits dans la ligne. D'autre part, il faut aussi trouver les mouvements du robot qui permettent de réaliser ce séquencement en tenant compte des contraintes. Nous chercherons la solution cyclique optimale qui, durant un cycle, produit exactement une pièce de chaque type. En effet, nous ne chercherons pas une solution qui serait, par exemple, de produire cinq pièces de type 1 puis 8 pièces de type 2 et ainsi de suite.

3.2.1 Objectif

Dans cette section, nous allons utiliser les travaux réalisés sur le problème du *flow-shop* sans attente à deux machines. Dans le cas d'une ligne à deux machines, Gilmore et Gomory [27] ont proposé un algorithme polynomial qui donne la solution optimale pour le problème $F2|no-wait|C_{max}$.

Pour simplifier, l'algorithme fonctionne de la manière suivante :
Il désigne, comme successeur, de la tâche qui a la plus petite durée dans la cuve T_2, la tâche qui a la plus petite durée dans la cuve T_1 et ainsi de suite. Si l'ordonnancement n'est pas réalisable, il inverse les tâches j et $j+1$ dont l'échange fait perdre le moins.

La solution obtenue minimise donc la date de sortie du dernier produit (C_{max}). Si on applique cet algorithme au cas cyclique on obtient la solution qui maximise le taux de sortie ce qui est équivalent à la solution qui minimise la longueur du cycle.

Notre étude consiste à regarder, pour n produits différents, la solution cyclique obtenue avec l'algorithme de Gilmore et Gomory quand les temps de process utilisés sont les temps minimaux pour tous les produits. Soit C_{opt} le cycle obtenu, la figure 3.2 représente un exemple de solution optimale pour $n = 5$ pour des temps de process donnés dans le tableau suivant.

Produit	l_1	l_2
1	7	7
2	5	7
3	2	2
4	3	4
5	6	4

Pour le *HSP*, il faut tenir compte des mouvements du robot pour le transfert de la cuve T_1 à T_2, pour le chargement (cuve T_0 à T_1) et pour le déchargement (cuve T_2 à T_3) de la ligne. Dans cette partie nous généraliserons le problème précédent

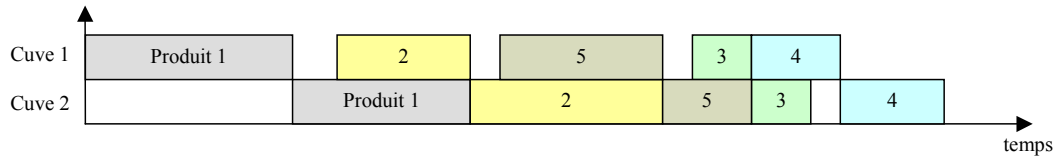

Figure 3.2 – Solution optimale sans le transport sur un exemple

en utilisant des durées de déplacements quelconques. Nous noterons $\delta_{i,j}$ (*resp* $D_{i,j}$) la durée de déplacement sous charge (*resp* à vide) pour passer de la cuve T_i à la cuve T_j. Nous noterons, enfin, respectivement $l_{i,k}$, $u_{i,k}$ et $p_{i,k}$ les durées minimales, maximales et effectives du produit k dans la cuve T_i.

Dans cette section, nous proposons un algorithme basé sur l'algorithme de Gilmore et Gomory pour trouver une solution au problème. L'optimalité de la solution ne sera pas prouvée bien que, expérimentalement, nous n'ayons pas trouvé de contre exemple.

3.2.2 Prise en compte des temps de transfert

Le transfert de la cuve T_1 à la cuve T_2 bloque les deux cuves à la fois car la cuve T_1 contient le porteur et la cuve T_2 doit être vide pour accueillir ce dernier.

Nous allons montrer maintenant que cette méthode donne encore le temps de cycle optimal. Quoi que l'on fasse, les temps de transferts d'une cuve à l'autre sont fixes et indépendants du porteur. Soit $T(C)$ la durée d'un cycle C sans transfert, $T(C_{trans})$ la durée de ce même cycle avec les transferts et $\delta_{1,2}$ le temps de transfert : $T(C_{trans}) = T(C) + n\delta_{1,2}$ or comme $\delta_{1,2}$ est une constante alors $T(C_{opt,trans}) = T(C_{opt}) + n\delta_{1,2}$. Comme $T(C_{opt})$ est donné par l'algorithme de Gilmore et Gomory, on peut trouver le temps de cycle optimal en considérant les temps de transfert d'une cuve à l'autre. La figure 3.3 illustre ces propos avec l'exemple présenté figure 3.2.

Donc, une solution optimale sans temps de transfert reste optimale avec temps de transfert de la cuve T_1 à la cuve T_2.

Dans la suite de notre étude, nous ne mentionnerons plus ces transferts que l'on peut ajouter lorsque l'ordonnancement optimal est trouvé.

3.2.3 Prise en compte des temps de déchargement

Dans notre cas, lors de l'utilisation de l'algorithme de Gilmore et Gomory, nous incluons les temps de chargement dans le temps d'opération dans la cuve T_1 et le temps de déchargement dans le temps d'opération dans la cuve T_2. En effet, quand

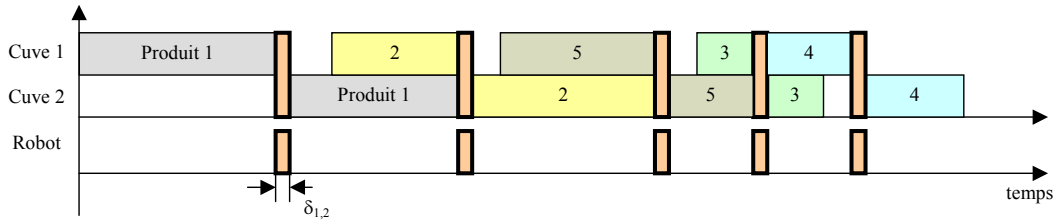

Figure 3.3 – Solution optimale avec les transferts

le robot va faire rentrer un produit sur la ligne la cuve T_1 doit être déjà libre, on peut donc considérer que l'opération sur la cuve T_1 rend indisponible la cuve non seulement pendant le temps de l'opération mais aussi pendant les temps de chargement.

Nous incluons aussi le temps de déplacement à vide pour aller de la cuve T_2 à la cuve T_0 ($D_{2,0}$) dans le temps d'opération dans la cuve 1, ainsi que le temps pour aller de la cuve T_3 à la cuve T_1 dans le temps d'opération dans la cuve T_2.

La figure 3.4 représente les mouvements que doit effectuer le robot entre le moment où il a déposé le produit j dans la cuve T_1 et le moment où il va le retirer.

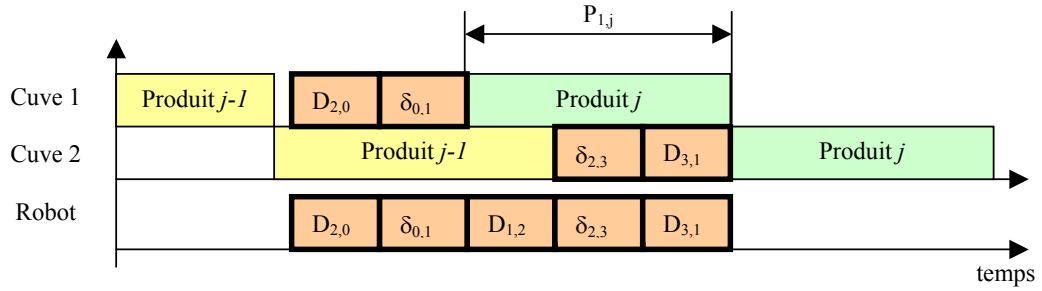

Figure 3.4 – Problèmes engendrés par le déchargement

Entre ces deux moments, le robot doit donc aller au-dessus de la cuve T_2 ($D_{1,2}$ unité de temps), sortir le produit $j-1$ ($\delta_{2,3}$) et retourner au dessus de la cuve T_1 ($D_{3,1}$). Donc si le temps de trempe dans la cuve T_1 est inférieur à $D_{1,2} + \delta_{2,3} + D_{3,1}$ alors l'opération du produit j dans la cuve T_1 ne pourra pas se faire en parallèle avec l'opération du produit $j-1$ dans la cuve T_2.

Nous pouvons remarquer que cette condition n'est pas dépendante de l'ordonnancement, mais simplement des durées de trempe qui sont des données du problème. Cette contrainte existera donc quel que soit l'ordonnancement.

Regardons maintenant comment nous pouvons utiliser les marges disponibles sur les temps de trempes.

Cas 1.

$u_{1,j} < D_{1,2} + \delta_{2,3} + D_{3,1}$ et $l_{2,j} \geq D_{2,0} + \delta_{0,1} + D_{1,2}$

Dans ce cas, même en augmentant la durée de trempe au maximum, on ne pourra pas traiter les deux produits en parallèle. Il faudra donc finir le traitement du produit $j-1$ avant de faire rentrer le produit j (figure 3.5).

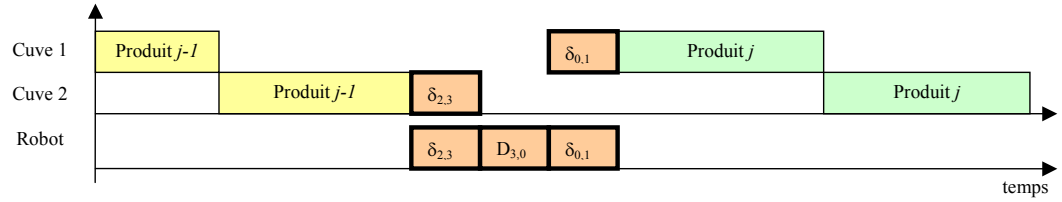

Figure 3.5 – Décalage de l'opération j

Ceci entraîne donc une perte de $D_{3,0} - D_{3,1} + \delta_{0,1} + l_{1,j}$ sur le temps de cycle. Or, cette perte ne dépend pas de l'ordonnancement. Donc, quel que soit le cycle C, le temps de cycle avec les temps de chargement ($T(C_{charg})$) est :
$T(C_{charg}) = T(C) + D_{3,0} - D_{3,1} + \delta_{0,1} + l_{1,j}$

Comme l'algorithme de Gilmore et Gomory donne le cycle optimal sans les temps de chargement, il donne aussi le cycle optimal avec le temps de chargement quand $u_{1,j} < D_{1,2} + \delta_{2,3} + D_{3,1}$.

Cas 2.

$u_{1,j} \geq D_{1,2} + \delta_{2,3} + D_{3,1}$ et $l_{2,j} \geq D_{2,0} + \delta_{0,1} + D_{1,2}$

Dans cette configuration, comme le montre la figure 3.6, non seulement il est possible de traiter les opérations des produits $j-1$ et j en parallèle mais en plus cela n'entraîne pas de perte de temps dans le cycle.

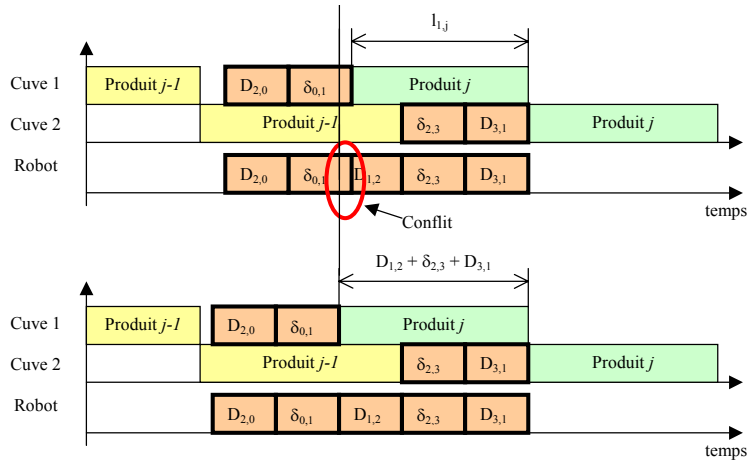

Figure 3.6 – Augmentation du temps de trempe dans la cuve 1

3.2.4 Prise en compte des temps de chargement

Nous pouvons tenir le même raisonnement pour tenir compte des temps de chargement. Comme le montre la figure 3.7, entre le moment ou le robot doit déposer un porteur dans la cuve T_2 et le moment où il le retire, il se passe au minimum $D_{2,0} + \delta_{0,1} + D_{1,2}$.

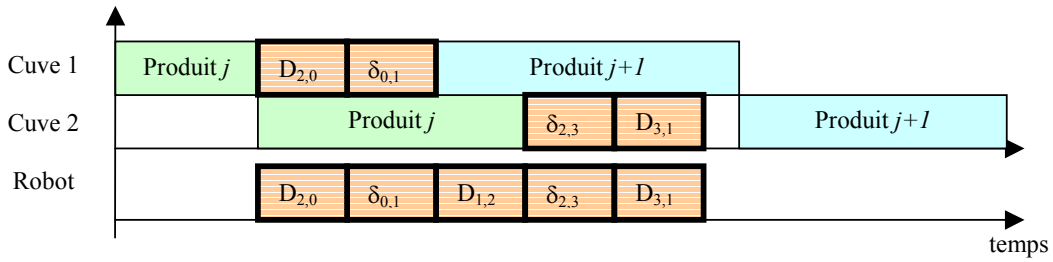

Figure 3.7 – Problèmes engendrés par le chargement

Cas 1.

$u_{2,j} < D_{2,0} + \delta_{0,1} + D_{1,2}$

Dans ce cas même en augmentant la durée de trempe au maximum on ne pourra pas traiter les deux produits en parallèle. Il faudra donc finir le traitement du produit j avant de faire rentrer le produit $j + 1$ (figure 3.8).
Ceci entraîne donc une perte de $l_{2,j} + \delta_{2,3} + D_{3,0} - D_{3,1}$ sur le temps de cycle.

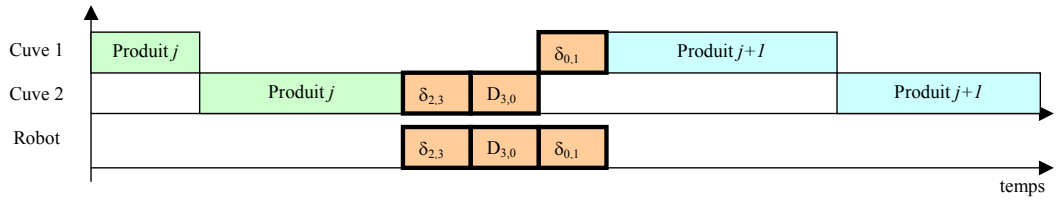

Figure 3.8 – Décalage de l'opération j

Cas 2.

$u_{2,j} \geq D_{2,0} + \delta_{0,1} + D_{1,2}$

Dans cette configuration, comme le montre la figure 3.9, non seulement il est possible de traiter les opérations des produits j et $j+1$ en parallèle et cela n'entraîne pas de perte de temps dans le cycle.

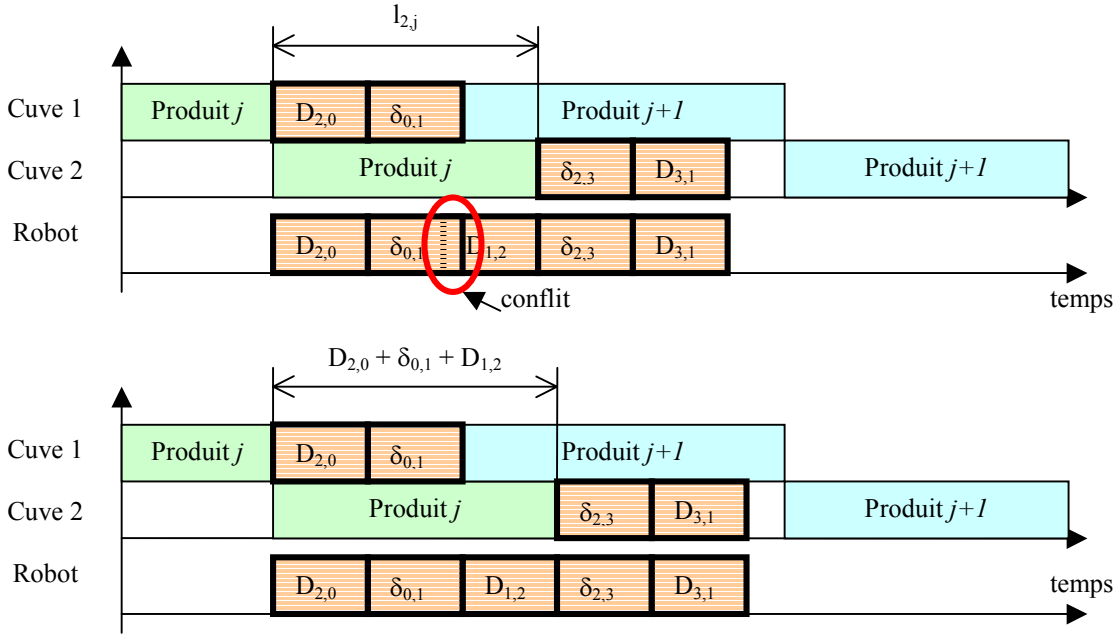

Figure 3.9 – Augmentation du temps de trempe dans la cuve 2

3.2.5 Opérations de courte durée en vis-à-vis

Nous avons vu la perte engendrée sur un temps de cycle lorsque nous ne pouvons pas jouer suffisamment sur la marge. Mais nous avons regardé les cas où seule une opération est trop petite. Or l'algorithme met de manière générale comme successeur d'une opération j celle dont la durée opératoire dans la cuve T_1 est la plus proche de $p_{2,j}$. Ceci entraîne que les pièces ayant les plus petites durées opératoires dans la cuve T_2 sont suivies de celles qui ont les plus petites durées opératoires dans la cuve T_1.

Nous allons maintenant regarder la perte de temps entraînée lorsqu'un porteur dont $l_{2,j} < D_{2,0} + \delta_{0,1} + D_{1,2}$ précède un porteur dont $l_{1,j+1} < D_{1,2} + \delta_{2,3} + D_{3,1}$. Cette situation est représentée figure 3.10.

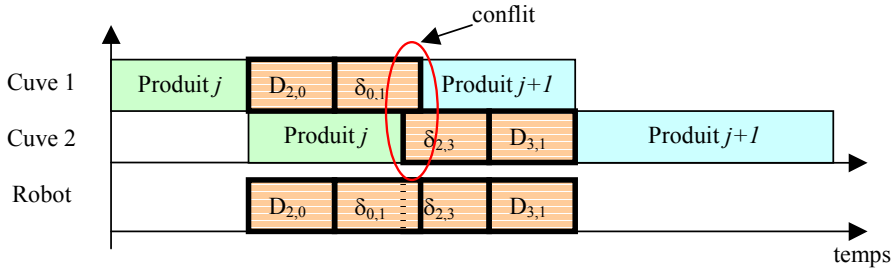

Figure 3.10 – Opérations les plus courtes en vis-à-vis

Comme le montre la figure 3.11 lorsque $u_{2,j} < D_{2,0} + \delta_{0,1} + D_{1,2}$ et $u_{1,j} < D_{1,2} + \delta_{2,3} + D_{3,1}$ alors la perte au niveau du temps de cycle et de $\min(l_{2,j} + \delta_{2,3} + D_{3,0} - D_{3,1}; D_{3,0} - D_{3,1} + \delta_{0,1} + l_{2,j})$. Dans l'hypothèse où les deux opérations ne sont pas mises en vis-à-vis, la perte au niveau du cycle est la somme des deux décalages : $(D_{3,0} - D_{3,1} + \delta_{1,2} + l_{1,j}) + (l_{2,j} + \delta_{2,3} + D_{3,0} - D_{3,1})$.

Supposons que l'algorithme de Gilmore et Gomory met comme prédécesseur d'un produit dont l'opération sur la cuve T_1 est inférieure à $D_{1,2} + \delta_{2,3} + D_{3,1}$ un produit dont l'opération sur la cuve T_2 est inférieure à $D_{2,0} + \delta_{0,1} + D_{1,2}$. Alors l'algorithme donne la solution optimale.

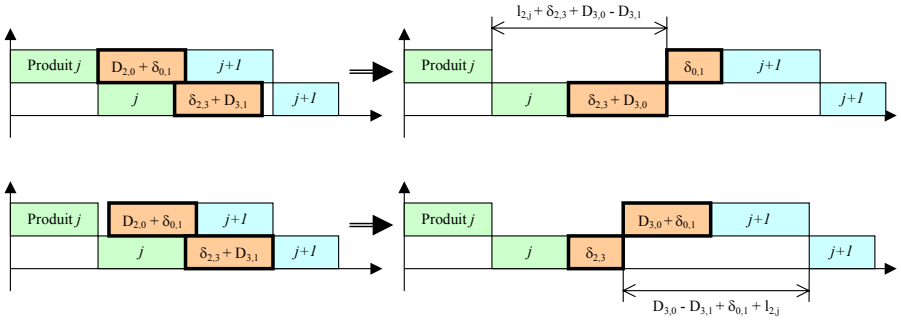

Figure 3.11 – Décalage pour les opérations les plus courtes en vis-à-vis

Par contre si les deux marges sont suffisamment grandes alors il faut augmenter les deux durées de trempe comme le montre la figure 3.12.

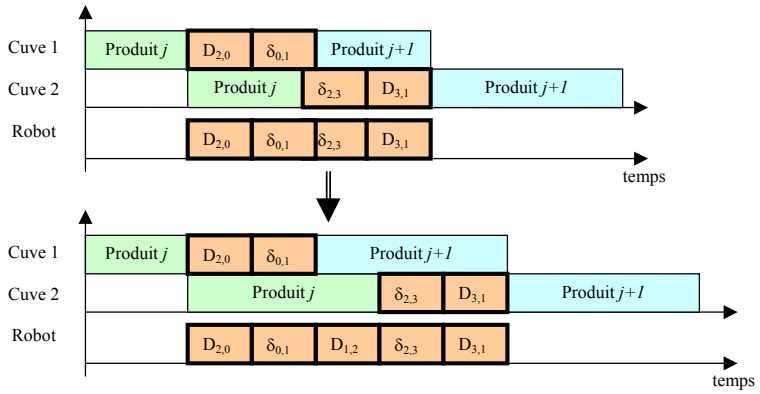

Figure 3.12 – Augmentation des durées de trempe pour les opérations les plus courtes en vis-à-vis

Augmenter les durées de trempe sur les deux opérations à la fois augmente donc le temps de cycle de :
$(D_{2,0} + \delta_{0,1} + D_{1,2} + \delta_{2,3} + D_{3,1}) - \max(D_{2,0} + \delta_{0,1} + l_{1,j+1} - D_{3,1}; l_{2,j} + \delta_{2,3} + D_{3,1})$

Dans ce cas il faut donc comparer cette perte avec la perte que l'on obtiendrait en proposant une séquence différente de celle obtenue par l'algorithme. Nous n'avons pas réussi, durant nos travaux à trouver un contre-exemple, tel que la séquence proposée par l'algorithme de Gilmore et Gomory ne soit pas optimale pour le *HSP*, mais nous n'avons pas de certitude sur l'optimalité de la séquence obtenue par l'algorithme de Gilmore et Gomory.

3.2.6 Application de l'algorithme sur un exemple

Pour résumer cette section, nous allons appliquer la méthode sur un exemple.

Nous considérerons les temps de transport comme additifs et indépendants de la charge. Dans cet exemple, le déplacement d'une cuve à la suivante vaut 1, donc $\delta_{i,j} = D_{i,j} = |i - j|$.

Soient les cinq produits suivants à ordonnancer.

Produit	Cuve 1		Cuve 2	
	min	max	min	max
1	3	3	4	6
2	3	5	6	8
3	10	12	4	6
4	12	12	10	10
5	10	10	8	8

- Si $u_{1,j} \geq D_{1,2}+\delta_{2,3}+D_{3,1}$ prendre $p_{1,j} = \max(l_{1,j}; D_{1,2}+\delta_{2,3}+D_{3,1})$ sinon $p_{1,j} = l_{1,j}$;
- Si $u_{2,j} \geq D_{2,0}+\delta_{0,1}+D_{1,2}$ prendre $p_{2,j} = \max(l_{2,j}; D_{2,0}+\delta_{0,1}+D_{1,2})$ sinon $p_{2,j} = l_{2,j}$.
  Dans notre exemple, nous aurons donc les temps effectifs suivants :

Produit (j)	1	2	3	4	5
$p_{1,j}$	3	4	10	12	10
$p_{2,j}$	4	6	4	10	8

- Utiliser l'algorithme de Gilmore et Gomory avec des temps opératoires de $p_{1,j} + D_{2,0} + \delta_{0,1}$ sur la cuve T_1 et $p_{2,j} + D_{3,1} + \delta_{2,3}$ sur la cuve T_2 pour tous les produits. Dans le cas de notre exemple, nous aurons les valeurs suivantes :

Produit (j)	1	2	3	4	5
$p_{1,j}$	6	7	13	15	13
$p_{2,j}$	7	9	7	13	11

  Dans cet exemple, si l'on applique l'algorithme de Gilmore et Gomory, on obtient le cycle (1,2,3,4,5). Si l'on dessine le résultat sous forme de diagramme de Gantt on obtient le résultat figure 3.13

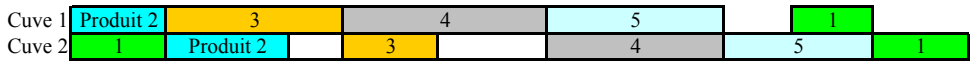

Figure 3.13 – Diagramme de Gantt du résultat obtenu par l'algorithme de Gilmore et Gomory

– Décaler tout produit dont $p_{1,j} < D_{1,2} + \delta_{2,3} + D_{3,1}$ de $D_{3,0} - D_{3,1} + \delta_{1,2} + l_{1,j}$. Sur notre exemple, il faut donc décaler le produit 1 comme le montre la figure 3.14. Sur cette figure nous avons ajouté l'occupation du robot.

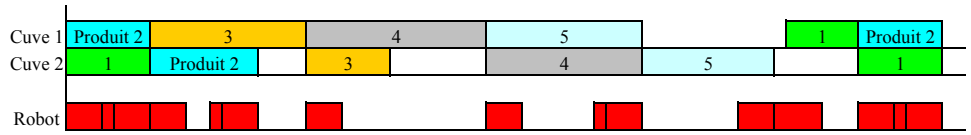

Figure 3.14 – Décalage des opérations pour notre exemple

– Décaler toute opération $j+1$ de $l_{2,j} + \delta_{2,3} + D_{3,0} - D_{3,1}$ lorsque pour l'opération précédente : $p_{2,j} \geq D_{2,0} + \delta_{0,1} + D_{1,2}$.

Dans notre exemple, nous obtenons donc le cycle optimal (1,2,3,4,5) dont le temps de cycle est de 69 unités de temps. Si nous regardons, dans cet exemple, les mouvements du robot en terme d'activités, nous obtenons le cycle 0, 2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 2, 0, 1. Ce cycle est donc un 6-cycle, ce qui montre que la conjecture 1 (dominance des 1, 2... $(m-1)$-cycles) ne peut pas être élargie au cas multi-produit.

3.3 Conclusions

Nous avons vu, dans la section 3.1, comment obtenir le cycle optimal pour le problème du HSP mono-produit. Pour cela, nous avons utiliser le *line-graph* et ses propriétés. Ces résultats confirment la conjecture 1 qui dit que les cycles optimaux sont à chercher parmi les 1,2... $(m-1)$-cycles.

Ensuite, nous avons montré, dans la section 3.2, comment utiliser l'algorithme de Gilmore et Gomory pour obtenir un cycle réalisable pour le problème multi-produit. Nous avons prouvé, pour cette méthode, les conditions qui garantissent l'optimalité du cycle, sans toutefois prouver que l'algorithme de Gilmore et Gomory donne la séquence optimale. De plus, cette étude a permis de montrer que la conjecture 1 ne s'applique pas au cas multi-produit.

Chapitre 4

Ligne équilibrée à trois cuves

Dans ce chapitre, nous considérons les lignes équilibrées (temps de trempe minimaux égaux dans chaque cuve), à trois cuves. Nous nous limiterons à une production mono-produit avec des fenêtres de temps de largeur nulle (temps de trempe minimal égal au temps de trempe maximal) ou infinie (temps de trempe maximal infini).

Après avoir montré le tableau des cycles optimaux (tableau 4.1), nous prouverons, par une étude exhaustive, l'optimalité de ces cycles[1]. Nous vérifierons, durant cette étude, que les cycles optimaux vérifient la conjecture 1 (pour une ligne à m cuves les cycles optimaux sont à chercher parmi les 1,2,... $(m-1)$-cycles).

Enfin nous verrons comment généraliser l'obtention des cycles optimaux au HSP équilibré à trois machines.

4.1 Présentation des cycles optimaux

Dans ce chapitre, nous étudions une ligne comportant trois cuves de traitement de surface (figure 4.1). Chaque produit doit rester, au minimum, une durée l dans chaque cuve. Les traitements étudiés sont sans marge (temps passé dans la cuve exactement égal à l) ou avec une marge infinie (durée dans la cuve devant simplement être supérieure à l).

[1] Ces travaux ont été présentés lors de la conférence PMS'02 [52] et sont à paraître dans la revue JESA [53]

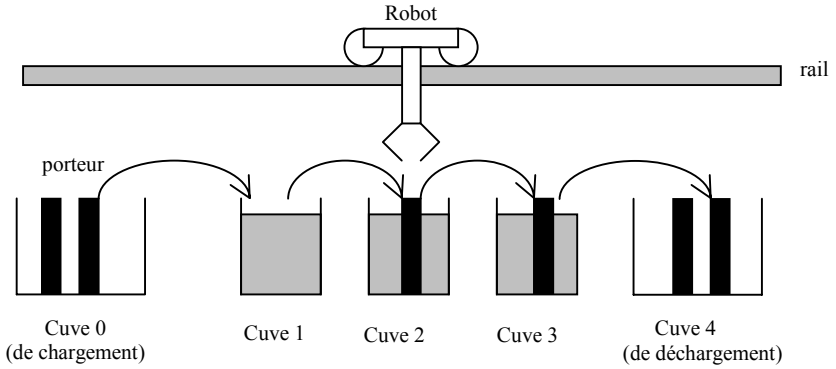

Figure 4.1 – Ligne comportant 3 cuves de traitement

Le but est de trouver les cycles de degré minimal qui minimisent le temps de cycle en fonction des données du problème :

- configuration de l'atelier : taille des fenêtres de temps dans chaque cuve,
- temps de trempe : l (égal dans toutes les cuves),
- temps de transport : δ (temps de déplacement du robot pour passer d'une cuve à une cuve voisine).

Le tableau 4.1 donne les cycles optimaux pour toutes les instances de notre étude. Les colonnes indiquent les intervalles dans lesquels se trouve le temps de trempe minimal en fonction de δ. Chaque ligne représente une configuration de la ligne. Par exemple, la configuration "$0, \infty, \infty$" indique que le temps de trempe dans la cuve T_1 est fixe (égal à l) et les temps de trempe dans les cuves T_2 et T_3 peuvent se trouver dans l'intervalle $[l, +\infty[$.

Tableau 4.1 – Cycles optimaux pour une ligne équilibrée à trois cuves

<table>
<tr><th></th><th>$[0, \delta[$</th><th>$[\delta, 2\delta[$</th><th>$[2\delta, 4\delta[$</th><th>$[4\delta, 6\delta[$</th><th>$[6\delta, 8\delta[$</th><th>$\geq 8\delta$</th></tr>
<tr><td>$0, 0, 0$</td><td rowspan="9"></td><td colspan="2" rowspan="4">0,1,2,3</td><td>0,2,1,3,2,3,0,1</td><td rowspan="4">0,2,1,3,2,0,3,1</td><td rowspan="9">$0, 3, 2, 1$</td></tr>
<tr><td>$\infty, 0, 0$</td><td>0,2,3,1</td></tr>
<tr><td>$0, 0, \infty$</td><td>0,1,3,2</td></tr>
<tr><td>$0, \infty, 0$</td><td colspan="2" rowspan="4">0,2,1,3</td></tr>
<tr><td>$0, \infty, \infty$</td><td colspan="2">0,1,3,2</td></tr>
<tr><td>$\infty, \infty, 0$</td><td colspan="2">0,2,3,1</td></tr>
<tr><td>$\infty, 0, \infty$</td><td></td><td>$0,3,1,2$</td><td colspan="2">0,2,1,3,2,0,3,1</td></tr>
<tr><td>∞, ∞, ∞</td><td>0,1,3,2 ou
0,2,3,1</td><td colspan="4">0,3,2,1</td></tr>
</table>

Pour limiter les calculs, nous allons utiliser les propriétés de symétrie de la ligne. En effet, si nous étudions un k-cycle $C_k = \alpha_1, \alpha_2, \dots \alpha_{k(m+1)}$ le k-cycle symétrique C_k^S est défini par $C_k^S = (m - \alpha_{k(m+1)}), (m - \alpha_{k(m+1)-1}), \dots (m - \alpha_1)$. Par exemple le cycle symétrique de (0,1,3,2) est le cycle (1,0,2,3). La figure 4.2 met en évidence cette symétrie.

Toutes les propriétés utilisées pour un cycle peuvent ainsi être utilisées pour la configuration symétrique (marges sur les durées de trempe, temps de process...) et pour le cycle symétrique (temps de cycle, conditions de faisabilité...).

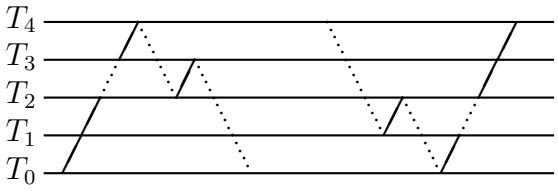

Figure 4.2 – Exemple de symétrie sur les cycles

Dans notre exemple le cycle (0,1,3,2) est réalisable si p_2 peut être supérieur à 4δ et si p_3 peut être supérieur à 6δ. Le cycle symétrique (1,0,2,3) est réalisable si p_2 peut être supérieur à 4δ et si p_1 peut être supérieur à 6δ.

Étant donnée la symétrie de la ligne entre T_1 et T_3, pour prouver les résultats, nous considérerons uniquement les cas suivants :

- Sans attente sur T_1 et $l < 8\delta$ (Section 4.2) (les résultats sont symétriques pour une configuration ou il n'y a pas d'attente sur T_3) ;
- Sans attente sur T_2 et $l < 8\delta$ (Section 4.4) ;
- $l \geq 8\delta$ (Section 4.5) ;
- Marge infinie sur toutes les cuves (Section 4.6).

En traitant uniquement tous ces cas, nous couvrons bien toutes les configurations possibles.

Le *line-graph* (figure 4.3) est construit pour une ligne dont les temps opératoires ne sont pas bornés.

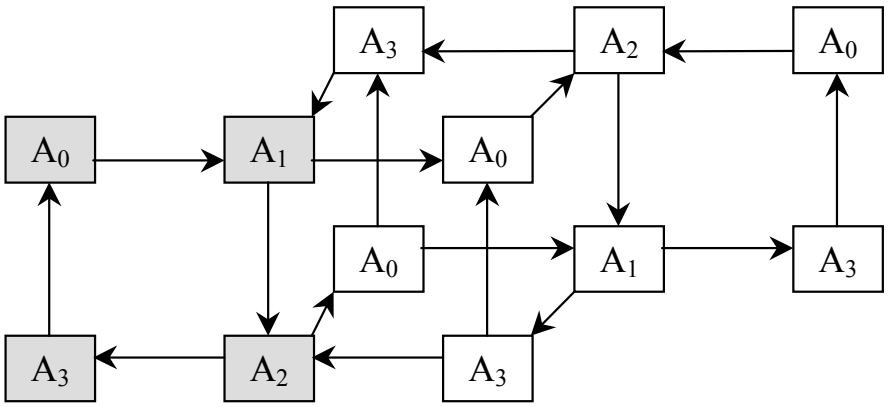

Figure 4.3 – Line-Graph LG_3

Or, dans notre étude, les temps opératoires peuvent être fixes (égal à l). Il se peut donc que certains circuits nécessitent des temps opératoires plus importants. Par exemple, si nous regardons la séquence (A_1, A_3, A_2), en gras sur la figure 4.4, le robot va déposer une pièce dans la cuve T_2 (A_1), puis va faire sortir un produit de la ligne (A_3), avant de venir récupérer la pièce dans la cuve T_2 pour la transporter dans la cuve T_3 (A_2). Si nous comptabilisons les temps de transport entre le dépôt de la pièce dans la cuve T_2 et sa sortie, il faut au moins 4δ.

Donc, pour que l'arc entre A_1 et A_3 soit réalisable, il faut que le temps de trempe dans T_2 (p_2) puisse être supérieur à 4δ. Donc il faut que le temps de process minimal l soit supérieur à 4δ ou que la marge sur la cuve T_2 soit infinie. Nous pouvons faire le même raisonnement avec tous les arcs et nous obtenons alors les conditions de faisabilité données dans la figure 4.4.

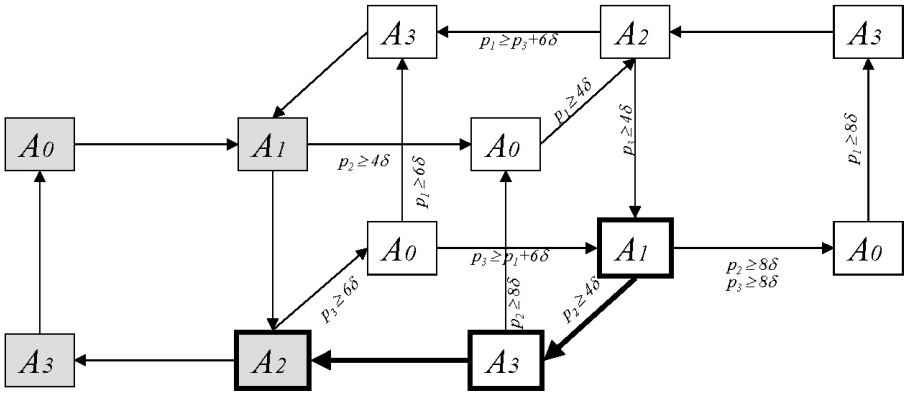

Figure 4.4 – Line-graph LG_3 avec les conditions de faisabilité

4.2 Sans attente sur T_1 et $l < 8\delta$

4.2.1 $l < 4\delta$

Quand $l < 4\delta$ et qu'il n'y a pas d'attente possible dans la cuve T_1, cela signifie que p_1 est strictement inférieur à 4δ. Tous les arcs de la figure 4.4 nécessitant un temps de trempe dans la cuve T_1 supérieur à 4δ peuvent donc être retirés. Si nous retirons ensuite les sommets qui sont inaccessibles ou dont on ne peut pas ressortir, on obtient alors le *line-graph* de la figure 4.5.

Il est possible de supprimer ces arcs d'une autre manière. En effet, dans cette configuration, le robot n'a pas assez de temps pour faire d'autres activités entre A_0 et A_1. Donc, dans tout cycle réalisable, toutes les activités A_0 doivent être immédiatement suivies par une activité A_1. Donc, tous les arcs qui débutent par une activité A_0 et qui aboutissent sur une activité autre que A_1 peuvent être retirés du *line-graph*. En enlevant tous les arcs impossibles nous obtenons également le *line-graph* de la figure 4.5.

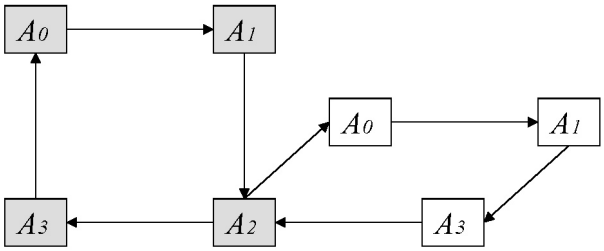

Figure 4.5 – *Line-graph* LG_3 pour $l < 4\delta$ et un traitement sans attente dans la cuve T_1

Tout k-cycle passe donc k fois par le nœud gris A_2 de l'ensemble E_3. La propriété 1 (un k-cycle passant plus d'une fois par un nœud de E_m est dominé par le meilleur sous-cycle), indique que le meilleur des sous-cycles entre $(0,1,2,3)$ et $(0,1,3,2)$ est dominant. Or le cycle $Id = (0,1,2,3)$ a un temps de cycle de $3l + 8\delta$ alors que le cycle $(0,1,3,2)$ a un temps de cycle de $l + 10\delta$. Ce qui veut dire que Id est dominant pour $l < \delta$ et le cycle $(0,1,3,2)$ est dominant pour $l \in [\delta; 4\delta[$ lorsqu'il est réalisable (configuration $(0,\infty,\infty)$).

4.2.2 $l \in [4\delta; 8\delta[$

Lorsque $l \in [4\delta; 8\delta[$, alors le temps de trempe dans la cuve T_1 est strictement inférieur à 8δ et donc, $p_1 < 8\delta$. Si nous retirons, selon le même principe que précédemment, tous les arcs impossibles dans cette configuration, nous obtenons le *line-graph* de la figure 4.6.

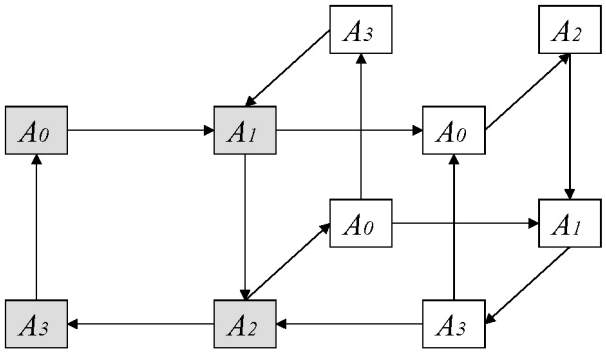

Figure 4.6 – *Line-graph* LG_3 pour $l < 8\delta$ et un traitement sans attente dans la cuve T_1

Tous les circuits associés à un k-cycle doivent traverser exactement k fois un nœud A_2. Si nous considérons la forme du graphe, alors les k-cycles, pour lesquels nous ne pouvons pas utiliser la propriété 1, ont un circuit associé qui passe $k-1$ fois par le nœud A_2 blanc et une fois par le nœud A_2 gris. Les différents cycles ainsi obtenus sont donnés dans le tableau 4.2. La troisième colonne indique les conditions de faisabilité des cycles.

Tableau 4.2 – Les k-cycles ($k > 1$) réalisables pour $l \in [4\delta; 8\delta[$ et sans attente sur la cuve T_1

	C_{ik}	Conditions de faisabilité $k \geq 3$
C_{1k}	$(\underbrace{0,2,1,3}_{k-1 \text{ fois}},2,0,1,3)$	$u_2 \geq 8\delta$, $u_3 = \infty$
C_{2k}	$(\underbrace{0,2,1,3}_{k-1 \text{ fois}},2,3,0,1)$	$u_2 \geq 8\delta$
C_{3k}	$(\underbrace{0,2,1,3}_{\text{k−1 fois}},2,0,3,1)$	$u_1 \geq 6\delta$, $u_2 \geq 8\delta$, $u_3 \geq 6\delta$

Nous allons maintenant calculer les temps de cycle pour tous les C_{ik} avec $i = 1, 2, 3$ et montrer qu'ils sont plus grands que $l + 8\delta$ (temps de cycle du cycle $(0, 2, 1, 3)$ pour $l < 8\delta$).

– $C_{1k} = (\underbrace{0,2,1,3}_{k-1 \text{ fois}},2,0,1,3)$ avec $k \geq 2$

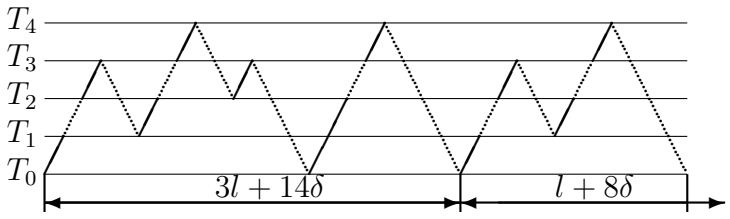

Figure 4.7 – Représentation du cycle C_{1k} pour $k = 3$

Le cycle C_{1k} est représenté sur la figure 4.7 pour $k = 3$. Pour $4\delta \leq l < 8\delta$, sa durée est la longueur de la première flèche $(3l + 14\delta)$ plus $(k - 2)$ fois la durée de la seconde $(l + 8\delta)$.

Donc pour $l \geq 4\delta$, nous obtenons :

$$\begin{aligned} T(C_{1k}) &= \frac{3l + 14\delta + (k-2)(l+8\delta)}{k} \\ &= (l+8\delta) + \frac{l-2\delta}{k} \text{ avec } l \in [4\delta; 8\delta[\\ &\geq (l+8\delta) \end{aligned}$$

Si $k = 2$ le cycle $C_{12} = (0, 2, 1, 3, 2, 0, 1, 3)$ a un temps de cycle égal à $\frac{3}{2}l + 7\delta$, ce qui est supérieur au temps de cycle du cycle (0,2,1,3). Il est donc dominé.

– $C_{2k} = (\underbrace{0, 2, 1, 3}_{k-1 \text{ fois}}, 2, 3, 0, 1)$ avec $k \geq 3$

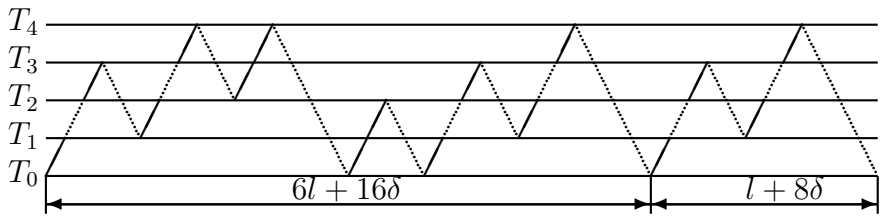

Figure 4.8 – Représentation du cycle C_{2k} pour $k = 4$

Le cycle C_{2k} est représenté sur la figure 4.8 pour $k = 4$. Pour $4\delta \leq l < 8\delta$, sa durée est la longueur de la première flèche $(6l + 16\delta)$ plus $(k-3)$ fois la durée de la seconde $(l + 8\delta)$.

Donc, nous obtenons :

$$\begin{aligned} T(C_{2k}) &= \frac{6l + 16\delta + (k-3)(l+8\delta)}{k} \\ &= (l+8\delta) + \frac{3l - 8\delta}{k} \text{ avec } l \in [4\delta; 8\delta[\\ &\geq (l+8\delta) \end{aligned}$$

Si $k = 2$ le cycle $C_{22} = (0, 2, 1, 3, 2, 3, 0, 1)$ a un temps de cycle égal à $2l + 6\delta$, ce qui est supérieur au temps de cycle du cycle (0,2,1,3). Il est donc dominé.

– $C_{3k} = (\underbrace{0, 2, 1, 3}_{\text{k-1 fois}}, 2, 0, 3, 1)$ avec $k \geq 3$

Le cycle C_{3k} est représenté sur la figure 4.9 pour $k = 4$. Pour $4\delta \leq l < 8\delta$, sa durée est la longueur de la première flèche $(5l + 14\delta)$ plus $(k-3)$ fois la durée de la seconde $(l + 8\delta)$.

Donc, nous obtenons :

$$\begin{aligned} T(C_{3k}) &= \frac{5l + 14\delta + (k-3)(l+8\delta)}{k} \\ &= (l + 8\delta) + \frac{2l - 10\delta}{k} \end{aligned}$$

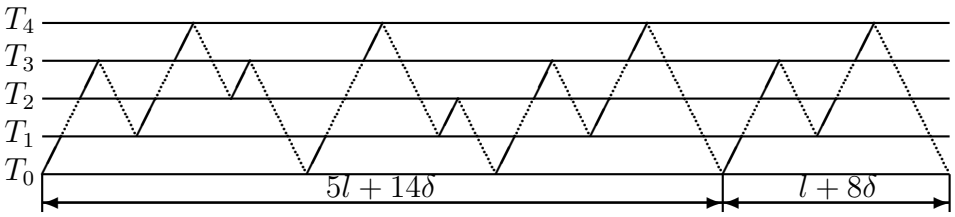

Figure 4.9 – Représentation du cycle C_{3k} pour $k = 4$

Le cycle C_{3k} est réalisable pour $l \geq 6\delta$ car il contient la séquence 031 et T_1 est une cuve sans-attente. Cette condition entraîne que $2l - 10\delta > 0$ et donc $T(C_{3k}) > l + 8\delta$. Ce qui montre que C_{3k} est dominé par le cycle $0, 2, 1, 3$.

Nous avons prouvé que pour $l \geq 4\delta$, les temps de cycle des C_{ik} $(i = 1, 2, 3)$ sont plus grands que $l + 8\delta$ pour $k \geq 3$. De plus, ces cycles ne sont réalisables que pour $u_2 \geq 8\delta$. Comme, de plus, ils sont dominés par le 1-cycle $(0, 2, 1, 3)$, dont la contrainte est $u_2 \geq 8\delta$. Cela prouve la conjecture 1 pour une configuration où le traitement sur T_1 est sans-attente et où $l < 8\delta$. Les temps de cycle des cycles dominants sont donnés dans le tableau 4.3.

Tableau 4.3 – k-cycles $(k \leq 2)$ réalisables pour $4\delta \leq l < 8\delta$ et un traitement sans attente dans la cuve T_1

Cycle	Temps de cycle	Conditions de faisabilité
$(0, 1, 2, 3)$	$3l + 8\delta$	
$(0, 3, 1, 2)$	$2l + 6\delta$	$u_1 \geq 6\delta$; $u_3 \geq 6\delta$
$(0, 1, 3, 2)$	$2l + 6\delta$	$u_3 = \infty$
$(0, 2, 1, 3)$	$l + 8\delta$	$u_2 \geq 8\delta \Rightarrow u_2 = \infty$
$C_{22} = (0, 2, 1, 3, 2, 3, 0, 1)$	$2l + 6\delta$	
$C_{32} = (0, 2, 1, 3, 2, 0, 3, 1)$	$\frac{3}{2}l + 5\delta$	$u_1 \geq 6\delta$; $u_3 \geq 6\delta$

A partir de ce tableau nous pouvons déduire les cycles optimaux suivant :

– 0,2,1,3,2,3,0,1 pour le cas sans attente sur toutes les cuves et $l < 6\delta$,
– 0,1,3,2 lorsque la marge est infinie dans la cuve T_3 et $l < 6\delta$.

– 0,2,1,3,2,0,3,1 pour le cas sans attente sur la cuve T_2 et $l \geq 6\delta$,
– 0,2,1,3 lorsque la marge est infinie dans la cuve T_2.

4.3 Sans attente sur T_3 et $l < 8\delta$

Le cas, où la cuve T_3 est sans attente est obtenu par symétrie. D'après la remarque faite dans la section 4.1, le calcul est identique en remplaçant C_{1k} par $C_{4k} = (\underbrace{0,2,1,3}_{\text{k−1 fois}},0,2,3,1)$.

4.4 Sans attente sur T_2 et $l < 8\delta$

Comme le temps de trempe dans la cuve T_2 doit être inférieur à 8δ, alors, au plus une opération peut être effectuée entre deux activités A_1 et A_2 consécutives. Une fois les arcs non permis retirés du *line-graph*, nous obtenons la figure 4.10.

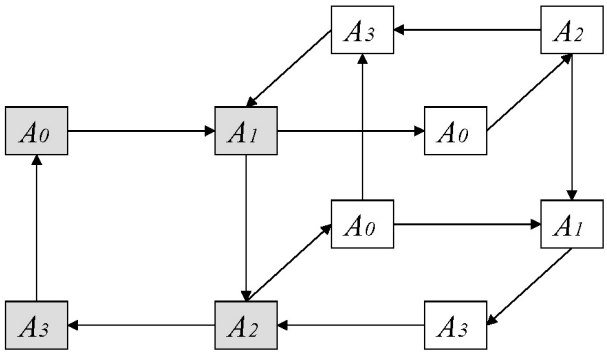

Figure 4.10 – Line-graph LG_3 pour $l < 8\delta$ et sans attente dans la cuve T_2

Pour tout k-cycle, les circuits associés doivent traverser exactement k fois un nœud A_2. Si nous considérons la forme du graphe, alors les k-cycles pour lesquels nous ne pouvons pas utiliser la propriété 1, ont un circuit qui passent $k-1$ fois par le nœud A_2 blanc et une fois par le nœud A_2 gris. Or, dans ce cas d'étude, les circuits qui passent par le nœud A_2 blanc, passent aussi par le nœud A_1 gris. Donc pour que les cycles ne soient pas dominés, il faut que $k-1$ soit inférieur ou égal à 1.

Si nous énumérons tous les 1 et 2-cycles réalisables, nous obtenons le tableau 4.4.
A partir de ce tableau, nous pouvons remarquer que le 1-cycle Id =$(0,1,2,3)$ est dominant pour $l < 2\delta$ quelles que soient les conditions sur les trempes dans les cuves T_1 et T_3.

Tableau 4.4 – Cycles de degré strictement inférieur à 3 réalisables, pour $l < 8\delta$ et un traitement sans attente dans la cuve T_2

Cycle	Temps de cycle	Conditions de faisabilité
$(0,1,2,3)$	$3l + 8\delta$	
$(0,1,3,2)$	$2l + 6\delta$	$u_2 \geq 4\delta$ et $u_3 = \infty$
$(0,2,3,1)$	$2l + 6\delta$	$u_2 \geq 4\delta$ et $u_1 = \infty$
$(0,3,1,2)$	$l + \max(l; 6\delta) + 6\delta$	$u_1 \geq 6\delta$ et $u_3 \geq 6\delta$
$C_{22} = (0,2,1,3,2,3,0,1)$	$2l + 6\delta$	$u_i \geq 4\delta\ \forall i$
$C_{32} = (0,2,1,3,2,0,3,1)$	$l + \frac{1}{2}\max(l; 6\delta) + 5\delta$	$u_1 \geq 6\delta$, $u_2 \geq 4\delta$ et $u_3 \geq 6\delta$

Lorsque $l \in [2\delta, 4\delta[$, le cycle Id est toujours dominant si les marges sur T_1 et T_3 ne sont pas infinies. Si elles sont infinies, le cycle $(0,3,1,2)$ est dominant.

Lorsque $l \in [4\delta, 6\delta[$, le 2-cycle $(0,2,1,3,2,0,3,1)$ est dominant quand les marges sur les cuves T_1 et T_3 sont infinies. Si seule la marge sur T_1 (resp T_3) est infinie alors, parmi les cycles réalisables, le cycle dominant est le cycle $(0,2,3,1)$ (resp $(0,1,3,2)$).

Lorsque $l \in [6\delta, 8\delta[$, le 2-cycle $(0,2,1,3,2,0,3,1)$ est dominant car il possède le plus petit temps de cycle parmi les cycles réalisables, et qu'il est réalisable quelles que soient les marges sur T_1 et T_3.

4.5 Temps de trempe minimal supérieur à 8δ

Pour $l \geq 8\delta$, le cycle $(0,3,2,1)$ est réalisable quelle que soit la configuration de la ligne. Crama et van de Klundert [20] ont montré que son temps de cycle $(l + 4\delta)$ est une borne minimale. Il est donc optimal.

4.6 Tolérance infinie sur toutes les cuves

Quand toutes les tolérances sont infinies, le problème équivaut au problème des cellules robotisées. Crama et van de Klundert [22] ont prouvé que, même pour des lignes non-équilibrées, les 1-cycles sont dominants. Donc la conjecture 1 est valide pour le cas (∞, ∞, ∞).

Les cycles optimaux sont alors les suivants :

– (0,1,2,3) pour $l \in [0; \delta[$,
– (0,1,3,2) ou (0,2,3,1) pour $l \in [\delta; 2\delta[$,
– (0,3,2,1) pour $l \geq 2\delta$.

4.7 Détermination des cycles optimaux par intervalle

Dans cette section, nous détaillons, sur un exemple, comment trouver les cycles optimaux pour une ligne équilibrée à trois cuves en utilisant la validité de la conjecture 1 et le tableau 4.3.

Nous considérons le cas $6\delta \leq l \leq 8\delta$ et pas d'attente permise sur les cuves T_1 et T_3. Dans la section 4.2, nous avons prouvé que le cycle optimal est soit un 1-cycle, soit $C_{22} = (0, 2, 1, 3, 2, 3, 0, 1)$ soit $C_{32} = (0, 2, 1, 3, 2, 0, 3, 1)$. Le tableau 4.3 indique que $0, 2, 1, 3$ n'est pas réalisable puisque $l \leq 8\delta$. Donc le cycle dominant est le cycle dont le temps de cycle est le plus petit parmi les suivants : $(0, 1, 2, 3)$, $(0, 3, 1, 2)$, $(0, 1, 3, 2)$, $(0, 2, 1, 3, 2, 3, 0, 1)$ et $(0, 2, 1, 3, 2, 0, 3, 1)$.

Dans notre cas, nous avons les inégalités suivantes : $8\delta + 3l \geq 6\delta + 2l \geq 5\delta + \frac{3}{2}l$. Donc, pour $6\delta \leq l \leq 8\delta$ et la configuration $(0, 0, \infty)$ ou (0,0,0), le cycle optimal est $C_{32} = (0, 2, 1, 3, 2, 0, 3, 1)$ dont le temps de cycle vaut $5\delta + \frac{3}{2}l$.

De la même manière, nous obtenons tous les cycles optimaux (tableau 4.5). Le tableau 4.5 donne les temps de cycle, en plus des cycles qui étaient déjà donnés dans le tableau 4.1 au début du chapitre.

Nous pouvons remarquer que, à la différence du problème des cellules robotisées, nous ne pouvons pas nous restreindre aux 1-cycles. En effet, il existe des configurations où le cycle dominant de degré minimum est un 2-cycle. Si nous prenons, par exemple, la configuration $(\infty, 0, \infty)$, un temps de transport $\delta = 1$ unité de temps et un temps de trempe $l = 5$ unités de temps, les temps de cycles des 1-cycles réalisables sont donnés dans le tableau 4.6.

Dans cette configuration le 2-cycle $(0, 2, 1, 3, 2, 0, 3, 1)$ possède un temps de cycle de 13 unités de temps, ce qui est meilleur que tous les 1-cycles réalisables (*i.e.* 16 unités de temps).

Tableau 4.5 – Cycles optimaux pour une ligne comportant trois cuves de traitement

<table>
<tr><th>p</th><th>$[0,\delta[$</th><th>$[\delta,2\delta[$</th><th>$[2\delta,4\delta[$</th><th>$[4\delta,6\delta[$</th><th>$[6\delta,8\delta[$</th><th>$\geq 8\delta$</th></tr>
<tr><td>$0,0,0$</td><td rowspan="8"></td><td rowspan="3">0, 1, 2, 3
$3l+8\delta$</td><td rowspan="3"></td><td>0, 2, 1, 3, 2, 3, 0, 1
$2l+6\delta$</td><td rowspan="3">0, 2, 1, 3, 2, 0, 3, 1
$\frac{3}{2}l+5\delta$</td><td rowspan="8">0, 3, 2, 1
$l+4\delta$</td></tr>
<tr><td>$\infty,0,0$</td><td>0, 2, 3, 1
$2l+6\delta$</td></tr>
<tr><td>$0,0,\infty$</td><td>0, 1, 3, 2
$2l+6\delta$</td></tr>
<tr><td>$0,\infty,0$</td><td colspan="3" rowspan="3">0, 2, 1, 3
$l+8\delta$</td></tr>
<tr><td>$0,\infty,\infty$</td><td colspan="2">0, 1, 3, 2
$l+10\delta$</td></tr>
<tr><td>$\infty,\infty,0$</td><td colspan="2">0, 2, 3, 1
$l+10\delta$</td></tr>
<tr><td>$\infty,0,\infty$</td><td colspan="2"></td><td>0, 3, 1, 2
$l+12\delta$</td><td>0, 2, 1, 3, 2, 0, 3, 1
$l+8\delta$</td><td>0, 2, 1, 3, 2, 0, 3, 1
$\frac{3}{2}l+5\delta$</td></tr>
<tr><td>∞,∞,∞</td><td colspan="2">0, 1, 3, 2
0, 2, 3, 1
$l+10\delta$</td><td colspan="3">0, 3, 2, 1
12δ</td></tr>
</table>

Tableau 4.6 – Temps de cycle pour $l = 5$ unités de temps, $\delta = 1$ unité de temps et $(\infty, 0, \infty)$

Cycle	temps de cycle
$(0,1,2,3)$	23
$(0,2,3,1)$	16
$(0,1,3,2)$	16
$(0,3,1,2)$	17

4.8 Extension au HSP équilibré

Comme pour le cas à deux cuves, nous pouvons étendre la méthode pour le *hoist scheduling problem* équilibré, c'est à dire lorsque tous les temps de trempe dans les cuves T_i sont dans un intervalle $[l; u_i]$.

Lorsque le cycle (0,3,2,1) est réalisable, on sait que, pour un temps de trempe minimum l supérieur à 2δ, il est optimal (tableau 4.5). Sinon ce sont parmi les 1-cycles (0,1,2,3), (0,1,3,2) et (0,2,3,1) qu'il faut chercher le cycle optimal en fonction des configurations (faisabilité et temps de cycle). Si il n'est pas réalisable on peut réduire le *line-graph* comme le montre la figure 4.11.

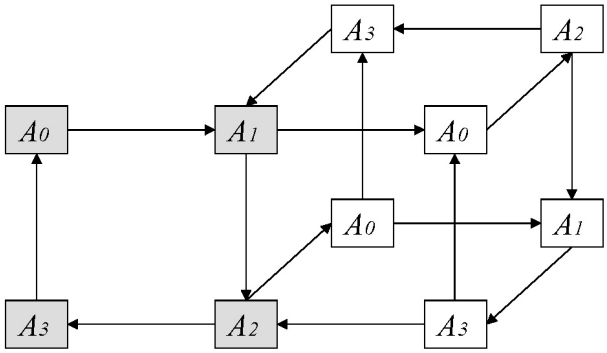

Figure 4.11 – Line-graph réduit lorsque le cycle (0,3,2,1) n'est pas réalisable

Si nous étudions ce *line-graph*, nous nous apercevons qu'il y a seulement deux sommets A_2 dont un qui fait partie de E_3 (ensemble des sommets gris).

Si l'on regarde les k-cycles ($k \geq 3$) qui ne sont pas concernés par la propriété 1 (cycles qui passent plus d'une fois par un même sommet de E_3 sont dominés), nous obtenons les cycles C_{ik} ($i = 1$ à 4) définis précédemment et le cycle $C_{5k} = (\underbrace{0,2,1,3}_{\text{k−2 fois}},0,2,3,1,2,0,1,3)$.

Dans les sections précédentes, nous avons montré que les k-cycles C_{ik} sont dominés par le cycle $0,2,1,3$. Le cycle C_{5k} est représenté sur la figure 4.12 pour $k = 3$.

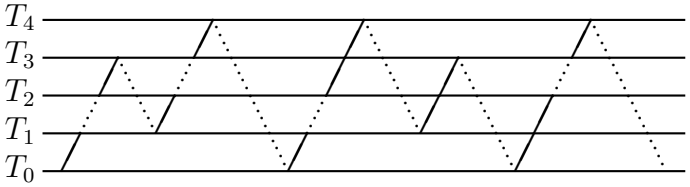

Figure 4.12 – Cycle C_{5k} pour $k = 3$

Le cycle C_{5k} n'est réalisable que si $u_1 \geq l + 6\delta$, $u_2 \geq 8\delta$ et $u_3 \geq l + 6\delta$. Dans ces conditions, le cycle (0,3,2,1) est aussi réalisable et donc il domine le cycle C_{5k}.

Nous pouvons donc en déduire que les k-cycles ($k \geq 3$) sont dominés par les 1- et 2-cycles. Pour trouver les cycles optimaux il suffit donc de regarder ce type de cycles et de comparer les cycles réalisables suivant les conditions de faisabilité et les temps de trempe.

Le tableau 4.7 récapitule tous les cycles optimaux précédemment trouvés. Pour chacun de ces cycles, nous donnons les conditions de faisabilité sur les temps de trempe effectifs (p_i).

Tableau 4.7 – 1 et 2-cycles optimaux pour le HSP à 3 cuves

Nom	Cycle	Conditions de faisabilité	Temps de cycle
C_{11}	$0,1,2,3$	pas de condition	$\sum_{i=1}^{3} l_i + 8\delta$
C_{21}	$0,1,3,2$	pas de condition sur p_1 $p_2 \geq 4\delta$ $p_3 \geq p_1 + 6\delta$	$\max(l_1 + 10\delta; l_1 + l_2 + 6\delta; l_3 + 4\delta)$
C_{31}	$0,2,1,3$	$p_1 \geq 4\delta$ $p_2 \geq 8\delta$ $p_3 \geq 4\delta$	$\max(0; l_1 - 4\delta; l_2 - 8\delta; l_3 - 4\delta; \frac{1}{2}\sum_{i=1}^{3} l_i - 8\delta) + 12\delta$
C_{41}	$0,2,3,1$	$p_1 \geq p_3 + 6\delta$ $p_2 \geq 4\delta$ pas de condition sur p_3	$\max(l_1 + 4\delta; l_2 + l_3 + 6\delta; l_3 + 10\delta)$
C_{51}	$0,3,1,2$	$p_1 \geq 6\delta$ pas de condition sur p_2 $p_3 \geq 6\delta$	$\max(l_1; l_3; 6\delta) + l_2 + 6\delta$
C_{61}	$0,3,2,1$	$p_i \geq 8\delta\ \forall i$	$\max(l_i; 8\delta) + 4\delta$
C_{22}	$0,2,1,3,2,3,0,1$	$p_i \geq 4\delta\ \forall i$	$1/2[l_1 + l_3 + \max(l_1 + l_2 + 4\delta; l_2 + l_3 + 4\delta; 12\delta) + 4\delta]$
C_{32}	$0,2,1,3,2,0,3,1$	$p_1 \geq 6\delta$ $p_2 \geq 4\delta$	$1/2[\max(l_1 + l_2 + 6\delta; l_2 + l_3 + 6\delta; 14\delta) + \max(l_1; l_3; 4\delta)]$

L'étude complète est très longue et n'apporte pas de résultats supplémentaires à ce que nous avons déjà montré. Ainsi nous ne développerons la méthode que pour le seul intervalle où $l \in [4\delta; 6\delta[$. Lorsque les marges sont infinies sur toutes les cuves, le tableau 4.5 (section 4.7) donne le cycle $C_{61} = (0, 3, 2, 1)$ comme optimal. Ce cycle est réalisable et optimal, si et seulement si, pour toutes les cuves, u_i est supérieur à 8δ. Si u_1 ou u_3 est strictement inférieur à 8δ, C_{61} n'est plus réalisable. Nous regardons ensuite les cycles réalisables et comparons leur temps de cycle pour obtenir le cycle optimal.

Par exemple, si u_2 et u_3 sont supérieures 8δ et $u_1 < 6\delta$ alors les cycles réalisables sont les cycles :

– C_{11}, $T(C_{11}) = 3l + 8\delta$;
– C_{21} (si $u_3 \geq l + 6\delta$), $T(C_{21}) = 2l + 6\delta$;
– C_{31}, $T(C_{31}) = 1.5l + 4\delta$;
– C_{22}, $T(C_{22}) = 2l + 6\delta$.

On peut donc en déduire que les cycles C_{21} et C_{22} sont optimaux. Comme le cycle C_{21} nécessite, en plus, que la borne supérieure sur la cuve T_3 soit supérieure à $l + 6\delta$ alors pour tout $u_3 \geq 8\delta$ le cycle optimal est le cycle C_{22}.

En tenant le même raisonnement pour chaque valeur de l, u_1, u_2 et u_3 ; nous pouvons obtenir les cycles optimaux donnés dans les tableaux 4.8 à 4.11.

Tableau 4.8 – HSP pour une ligne à trois cuves et $l < 2\delta$

l	$[0, \delta[$	$[\delta, 2\delta[$				
u_1		qcq	$< l + 6\delta$		$\geq l + 6\delta$	
u_2		$< 4\delta$	$\geq 4\delta$			
u_3		qcq	$< l + 6\delta$	$\geq l + 6\delta$		$< l + 6\delta$
cycle	C_{11}			C_{21}	C_{21} ou C_{41}	C_{41}

Tableau 4.9 – HSP pour une ligne à trois cuves et $2\delta < l < 4\delta$

l	$[2\delta, 4\delta[$								
u_1	$< 6\delta$	$\geq 6\delta$		$< l + 6\delta$			$\geq l + 6\delta$		$\geq 8\delta$
				$< 6\delta$	$\geq 6\delta$				
u_2	$< 4\delta$			$\geq 4\delta$					$\geq 8\delta$
u_3	$\geq 6\delta$	$< 6\delta$	$\geq 6\delta$	$\geq l + 6\delta$	$< l + 6\delta$			$\geq l + 6\delta$	$\geq 8\delta$
					$\geq 6\delta$	$< 6\delta$	$\geq 6\delta$		
cycle	C_{11}		C_{51}	C_{21}	C_{11}		C_{51}	C_{41}	C_{21} ou C_{41}	C_{61}

Tableau 4.10 – HSP pour une ligne à trois cuves et $4\delta < l < 6\delta$

l	$[4\delta, 6\delta[$								
u_1	$< 6\delta$			$\geq 6\delta$		$< 8\delta$		$\geq 8\delta$	
				$< l + 6\delta$	$\geq l + 6\delta$				
u_2	$< 8\delta$					$\geq 8\delta$			
u_3	$< 6\delta$	$\geq 6\delta$		$< 6\delta$		$\geq 8\delta$	$< 8\delta$	$\geq 8\delta$	
		$< l + 6\delta$	$\geq l + 6\delta$						
cycle	C_{22}		C_{21}	C_{32}	C_{22}	C_{41}	C_{22}		C_{61}

Tableau 4.11 – HSP pour une ligne à trois cuves et $l \geq 6\delta$

l	$[6\delta, 8\delta[$			$\geq 8\delta$
u_1	$< 8\delta$		$\geq 8\delta$	
u_2	$< 8\delta$	$\geq 8\delta$		
u_3	$< 8\delta$	$\geq 8\delta$	$< 8\delta$	$\geq 8\delta$
cycle	C_{32}	C_{31}		C_{61}

4.9 Conclusion

Dans ce chapitre, nous avons généralisé le travail réalisé pour le cas deux cuves au cas 3 cuves. Nous avons montré, dans le cas équilibré, que les cycles optimaux, lorsque les marges sont nulles ou infinies, sont des 1-cycles ou des 2-cycles. Nous avons utilisé, comme dans le chapitre 3, les propriétés des *line-graphs*. Nous avons travaillé, cette fois, sur des *line-graphs* réduits obtenus en supprimant les arcs impossibles, ces arcs nécessitant des durées de trempe trop grandes par rapport aux marges disponibles.

Cette recherche des cycles optimaux élargit la plage de validité de la conjecture 1, proposée par Agnetis dans le cas *sans-attente*, au cas où les temps de trempe sont bornés (HSP équilibré à trois cuves).

Chapitre 5

Ligne équilibrée à quatre cuves

Dans le chapitre précédent, nous avons montré que pour une ligne à trois cuves les 1- et 2-cycles sont dominants et ce quelles que soient les marges sur les durées de trempe. Dans ce chapitre, nous montrerons que la conjecture d'Agnetis (les cycles dominants sont à chercher parmi les 1, 2 ... (m-1)-cycles) est valide pour une ligne équilibrée à quatre cuves sans attente (la figure 5.1 représente une autre conception possible de ligne).

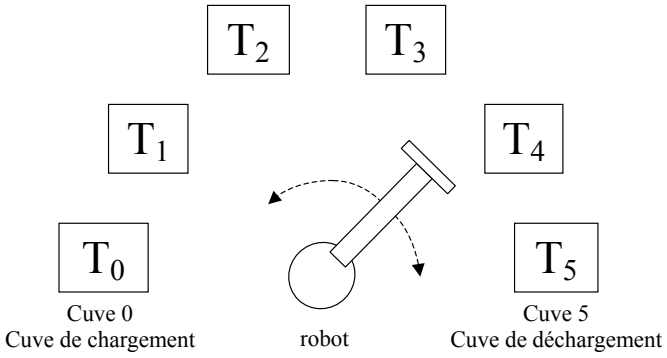

Figure 5.1 – Ligne à quatre cuves

Pour cela, nous allons montrer qu'il existe des 1-cycles dominants (section 5.2), des 2-cycles dominants (section 5.3) et des 3-cycles dominants (section 5.4). Notre étude portera sur le *flowshop* équilibré, sans attente. Cela signifie que les temps de trempe minimaux sont égaux ($l_i = l$) et que les marges sont nulles ($u_i = l_i = l$)[1]. Nous utiliserons donc, dans la suite du chapitre, le temps de trempe effectif p car nous avons $p = l = u$. Dans ce chapitre, nous utiliserons les propriétés des *line-graphs*. Enfin, nous verrons les limites de la méthode qui utilise les graphes.

[1]Ces travaux ont été présentés lors de la conférence MOSIM'03 [54]

5.1 Construction des graphes

La figure 5.2 représente le graphe d'état d'une ligne à quatre cuves.

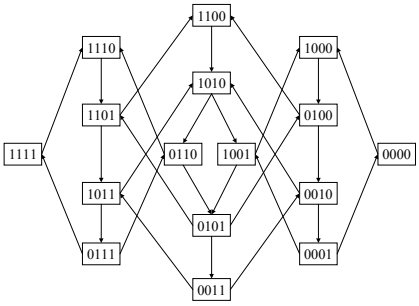

Figure 5.2 – Graphe d'état pour une ligne à quatre cuves

La figure 5.3 représente le *line-graph* LG_4 du graphe d'état G_4. Sur cette figure le nœud X_Y signifie que l'activité X est réalisé et Y permet de distinguer les nœuds représentant une même activité.

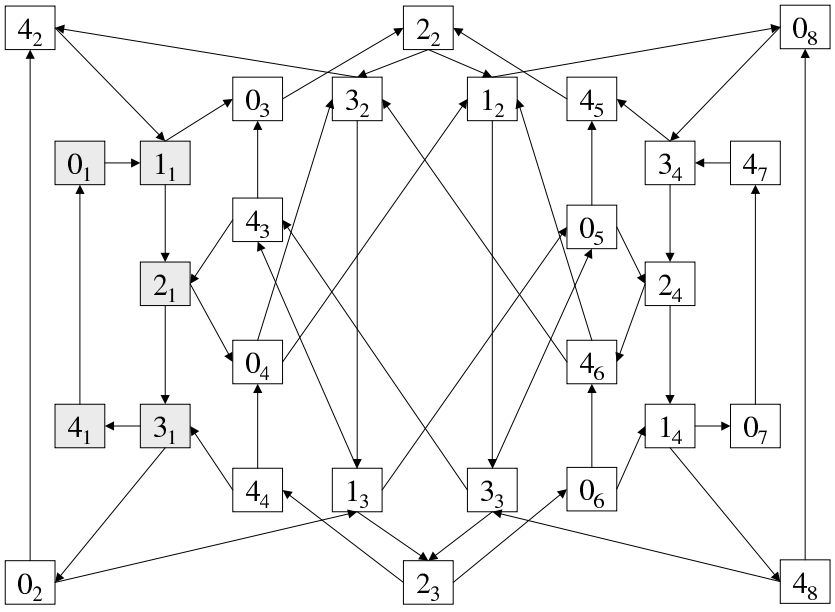

Figure 5.3 – *Line-graph* LG_4 du graphe d'état pour une ligne à quatre cuves

5.2 Optimalité d'un 1-cycle

Dans cette section nous montrons que, lorsque p se trouve dans l'intervalle $[0; 4\delta[$ ou $[6\delta; 10\delta[$ ou lorsque p est supérieur ou égal à 12δ, il existe des 1-cycles dominants.

5.2.1 p dans l'intervalle $[0; 4\delta[$

Pour les instances qui vérifient $p \in [0; 4\delta[$, le robot n'a pas le temps de réaliser une activité entre deux activités successives. Donc, les activités α et $\alpha + 1$ doivent être consécutives. Les seuls k-cycles (pour tout k) réalisables sont alors les répétitions k fois du cycle Id=(01234). Le cycle Id=(01234) est donc optimal et son temps de cycle est de $4p + 10\delta$.

5.2.2 p dans l'intervalle $[6\delta; 8\delta[$

Le *line-graph* figure 5.3 est construit pour une ligne dont les temps de trempe ne sont pas bornés. Si tous les arcs qui ne sont pas possibles sont retirés, le *line-graph* figure 5.4 est obtenu.

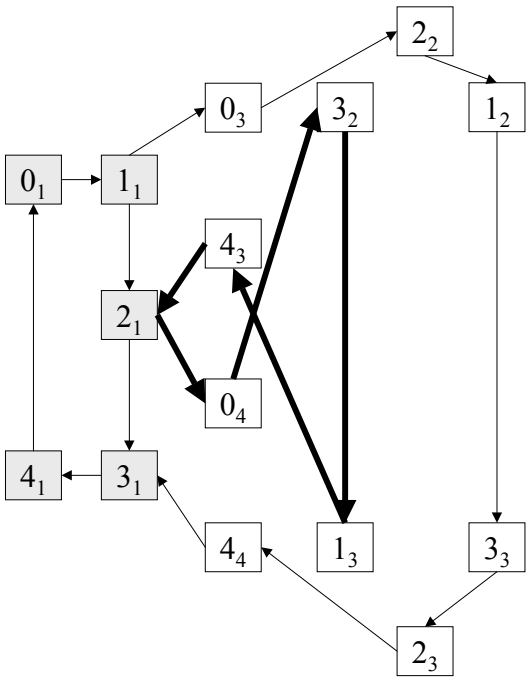

Figure 5.4 – Line-graph quand $p \in [6\delta; 8\delta[$

A partir de ce *line-graph* réduit, nous nous apercevons que tout circuit qui passe par le nœud 0_3 provient forcément du nœud 1_1. Or ce nœud appartient à E_4 et donc les cycles, dont les circuits associés passent plus d'une fois par 0_3, sont dominés. Le même raisonnement peut être appliqué aux nœuds 2_2, 1_2, 3_3, 2_3, et 4_4. Nous pouvons remarquer que chaque nœud étiqueté par l'activité 2 ne peut être traversé plus d'une fois. Or il y a trois nœuds de ce type. Donc le degré du cycle optimal pour $p \in [6\delta; 8\delta[$ est inférieur à 3.

Il est possible, maintenant, de construire facilement tous les cycles réalisables et calculer leur temps de cycle :

– $(01234) : 4p + 10\delta$,
– $(0102132434) : 5p/2 + 7\delta$,
– $(03142) : 2p + 6\delta$.

En comparant tous ces temps de cycle, nous nous apercevons que le cycle qui possède le plus petit temps de cycle sur l'intervalle $[6\delta; 8\delta[$ est le cycle (03142) (figure 5.5). Ce cycle est donc le cycle optimal.

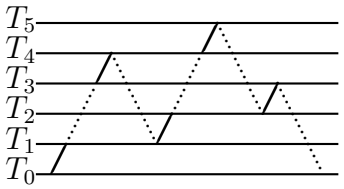

Figure 5.5 – Cycle (03142)

5.2.3 p dans l'intervalle $[8\delta, 10\delta[$

Si nous retirons les arcs qui entraînent des séquences non réalisables, nous obtenons le *line-graph* réduit représenté sur la figure 5.6.

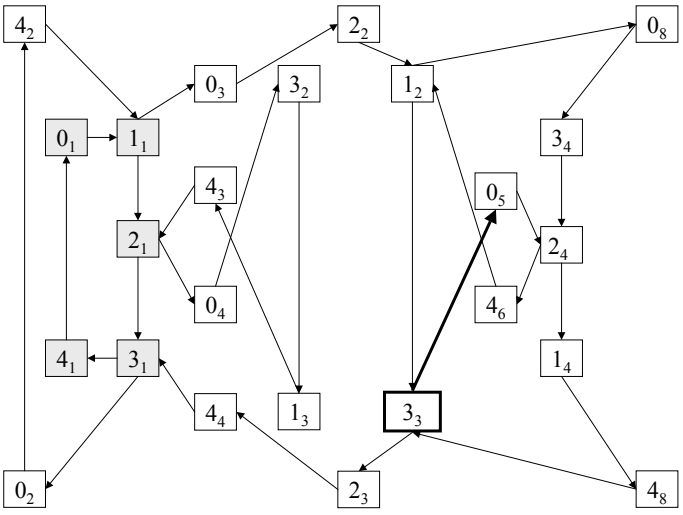

Figure 5.6 – *Line-graph* LG_4 pour $p \in [8\delta, 10\delta[$

Ce *line-graph* peut être encore simplifié, ce qui limitera encore le domaine de recherche. En effet, intéressons nous au nœud 3_3 et à la faisabilité de l'arc $3_3, 0_5$ (en

gras sur la figure 5.6). Nous nous apercevons que la séquence $1_4, 4_8, 3_3, 0_5, 2_4$ n'est pas réalisable. Le produit qui entre dans la cuve 2 avec l'activité 1_4 n'en ressort qu'après l'exécution des activités 4,3,0. Cette séquence dure un temps minimum de 12δ. Par contre la séquence $1_2, 3_3, 0_5, 2_4$ est réalisable.

Ceci veut dire que l'arc $3_3 0_5$ est réalisable si l'on vient du nœud 1_2, mais qu'il n'est pas réalisable si l'on vient du nœud 4_8. Enfin la séquence $2_2, 1_2, 3_3, 2_3$ ne pose pas de problème. Pour tenir compte de la provenance pour un nœud, nous l'avons séparé en deux (3_3 et $3_{3'}$). Ces sommets sont alors joints par un arc unidirectionnel qui entraîne que la séquence $2_2, 1_2, 3_3, 2_3$ est possible. Nous pouvons réaliser la même chose pour le nœud 1_2 qui sera séparé en 1_2 et $1_{2'}$.

Le même raisonnement peut aussi être tenu pour le nœud 2_4 qui sera partagé en 2_4 et $2_{4'}$. Dans ce cas, les deux arcs ne sont pas liés par un arc. En effet le chemin $1_2, 0_8, 3_4, 2_4, 4_6$ n'est pas réalisable et le chemin $0_5, 2_4, 1_4$ non plus. Nous obtenons ainsi un *line-graph* qui a certes plus de sommets, mais dont le nombre de circuits possibles est plus faible (figure 5.7).

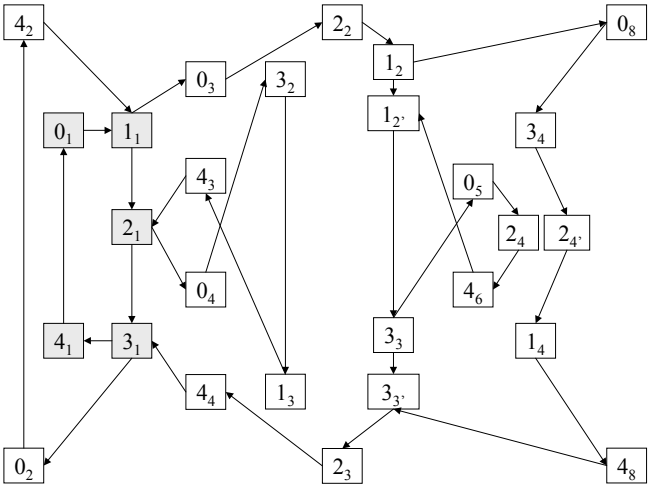

Figure 5.7 – *Line-graph* LG_4 pour $p \in [8\delta, 10\delta[$ avec nœuds séparés

A partir de ce graphe, il est plus facile d'énumérer tous les cycles réalisables et pour lesquels la propriété 1 n'est pas applicable.

– (02413), temps de cycle : $3p/2 + 4\delta$,
– (2430410213 $\underbrace{02413}_{k-2 \text{ fois}}$), temps de cycle : $(5p + 16\delta + (k-2)(3p/2 + 4\delta))/k$

Comparons ce temps de cycle avec le temps de cycle de (02413). Si le cycle est dominant, alors nous avons l'inégalité suivante sur les temps de cycle :

$$(5p + 16\delta + (k-2)(3p/2 + 4\delta))/k > 3p/2 + 4\delta$$

Si nous simplifions cette équation nous obtenons l'équation suivante :

$$5p + 16\delta - (3p + 8\delta) > 0$$

Comme cet inégalité est toujours vérifiée, le cycle (02413) domine le k-cycle.

– (021032143243041), le temps de cycle est $5p/3 + 16/3$. Ce temps étant supérieur au temps de cycle de (02413), ce dernier est donc dominant.

Le cycle (02413) domine aussi les cycles étudiés dans la section précédente, il est donc optimal et son temps de cycle est de $3p/2 + 4\delta$ (figure 5.8).

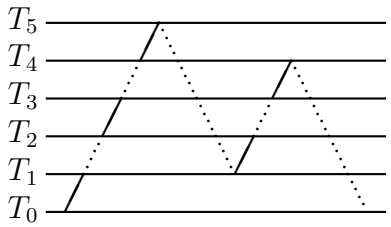

Figure 5.8 – Cycle (02413), optimal pour $p \in [8\delta; 10\delta[$

5.2.4 $p \geq 12\delta$

Dans cette configuration, le cycle (04321) est réalisable. Comme il a été prouvé que son temps de cycle atteint une borne minimale [20], alors il est dominant. Son temps de cycle est de $p + 4\delta$.

5.3 Optimalité d'un 2-cycle pour $p \in [4\delta; 6\delta[$

Nous pouvons remarquer que si $p \in [4\delta; 6\delta[$, alors entre une activité $\alpha-1$ et l'activité α suivante, il peut y avoir soit l'activité $\alpha-2$, soit l'activité $\alpha+1$, soit aucune activité.

En effet, soit le moment où un porteur est déposé dans la cuve T_α par l'activité $\alpha-1$. Si un autre porteur est traité dans une cuve T_β le robot doit aller le récupérer. Il doit donc aller au dessus de la cuve T_β ce qui lui prend $|\alpha - \beta|\delta$ unités de temps. Il doit ensuite déplacer le porteur de la cuve T_β à la cuve $T_{\beta+1}$ (δ unités temps). Enfin il doit retourner au dessus de la cuve T_α pour sortir le porteur ($|\alpha - (\beta+1)|\delta$ unités de temps).

– Si $\alpha > \beta$:
Le temps entre le dépôt d'un porteur et le retour du robot pour le ressortir est de : $(\alpha - \beta + 1 + \alpha - (\beta + 1))\delta = 2(\alpha - \beta)\delta$. Les conditions de faisabilité impliquent donc que le temps de trempe doit être supérieur à cette valeur. Donc $p \geq 2(\alpha - \beta)\delta$. Or l'intervalle d'étude impose que p soit inférieur strictement à 6δ. Donc, nous en déduisons que $6\delta > 2(\alpha - \beta)\delta$. Comme α et β sont des entiers alors $\beta \geq \alpha - 2$. Or β est forcément différent de $\alpha - 1$. Donc si $\alpha > \beta$ alors $\beta = \alpha - 2$.

– Si $\beta > \alpha$:
De la même manière, il faut que p soit supérieur à $2(\beta + 1 - \alpha)\delta$. Comme p est inférieur strictement à 6δ, on peut en déduire que $\beta + 1 - \alpha < 3\delta$ donc que $\beta \leq \alpha + 1$. Comme α et β sont différents alors si $\alpha < \beta$ alors $\beta = \alpha + 1$.

Cette propriété implique que les seuls cycles réalisables sont des concaténations de $C_1(2) = (0102132434)$ et Id. Si un tel cycle est de degré supérieur à 2, alors il passe plus de deux fois par un même nœud de E_4 dans le *line-graph*. Quand $p \in [4\delta; 6\delta[$ alors le 2-cycle $C_1(2) = (0102132434)$ (figure 5.9) est optimal et domine strictement tous les autres cycles. Le temps de cycle relatif de $C_1(2)$ est $5p/2 + 7\delta$.

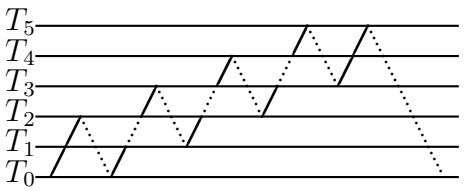

Figure 5.9 – Cycle (0102132434)

5.4 Optimalité d'un 3-cycle pour p dans l'intervalle $[10\delta, 12\delta[$

Dans cette section nous montrons qu'il existe un 3-cycles optimal, et donc, que si l'on doit rechercher des cycles optimaux, trouver les meilleurs 1- et 2-cycles ne suffit pas.

Nous allons prouver que dans cette configuration, le 3-cycle $C_3(3)$=(043104210321432) est optimal. Le temps de cycle relatif de $C_3(3)$ est $4p/3 + 14\delta/3$. Sa représentation pyramidale est donnée dans la figure 5.10

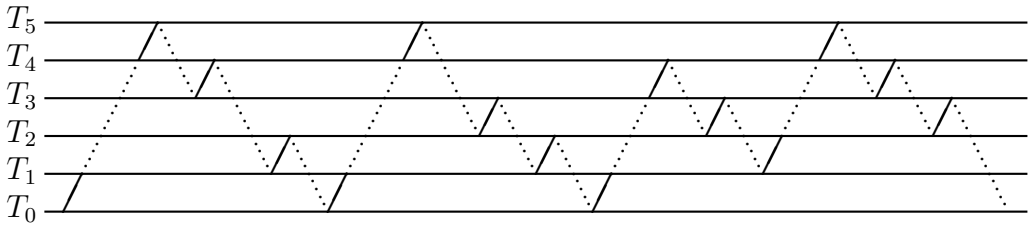

Figure 5.10 – Cycle (043104210321432) optimal pour $p \in [10\delta; 12\delta[$

Si on retire les arcs non réalisables, pour $p \in [10\delta, 12\delta[$, nous obtenons le *line-graph* de la figure 5.11. Les traits en gras représentent le circuit correspondant au cycle $C_3(3)$.

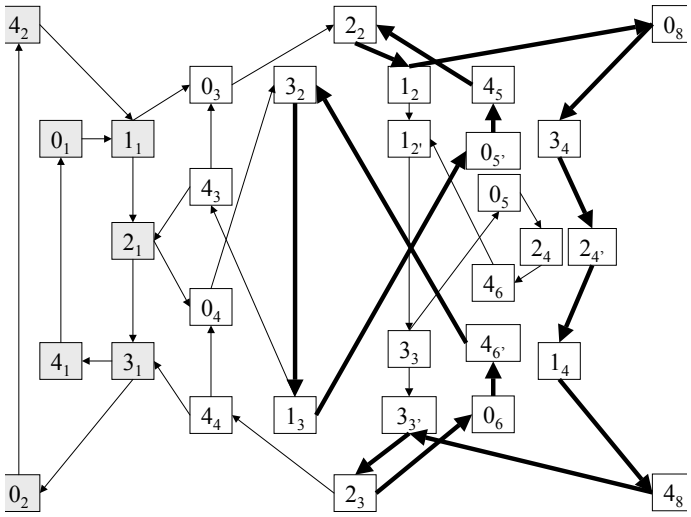

Figure 5.11 – Line-graph LG_4 pour $p \in [10\delta, 12\delta[$

Pour prouver la dominance de $C_3(3)$, nous démontrons que son temps de cycle relatif est inférieur à celui des 1-cycles (paragraphe 5.4.1), puis des 2-cycles (paragraphe 5.4.2) et enfin des 3-cycles (paragraphe 5.4.3). Pour $k > 3$ nous n'avons as pu prouver le résultat.

5.4.1 Dominance de $C_3(3)$ sur tous les 1-cycles

Etant données les conditions sur les temps de trempe, il ne reste que quatre 1- cycles réalisables. Le tableau 5.1 donne ces cycles ainsi que leur temps de cycle.

Cycle	temps de cycle
01234	$4p + 10\delta$
02413	$3p/2 + 4\delta$
04123	$3p + 8\delta$
03142	$2p + 6\delta$

Tableau 5.1 – 1-cycles réalisables et leur temps de cycle

Tous ces temps de cycle sont supérieurs à $4p/3+14\delta/3$ dans l'intervalle étudié, donc $C_3(3)$ domine strictement tous ces 1-cycles.

5.4.2 Dominance de $C_3(3)$ sur tous les 2-cycles

Pour montrer la dominance de $C_3(3)$ sur les 2-cycles, nous allons regarder les séquences possibles entre une activité A_1 et l'activité A_2 suivante. Pour chaque séquence, nous regardons si il est possible de reboucler sur l'activité A_1 par un 2-cycle. En effet si l'on regarde la séquence 1032 ($1_2 0_8 3_4 2_4$ sur le graphe), il est impossible de terminer le circuit en effectuant un 2-cycle. Pour les séquences où un 2-cycle est réalisable, il suffit de comparer le temps de cycle avec $4p/3 + 14\delta/3$ (temps de cycle de $C_3(3)$).

Les cycles obtenus sur le *line-graph* sont alors :

- (0102132434), temps de cycle : $5p/2 + 7\delta$
- (0310421324). Ce cycle est a priori réalisable car tous les temps de transport entre deux activités α et $\alpha+1$ sont inférieurs à 12δ. Soient X, Y et Z les temps d'attente. La (figure 5.12) représente ce cycle où nous avons rajouté les temps d'attente.

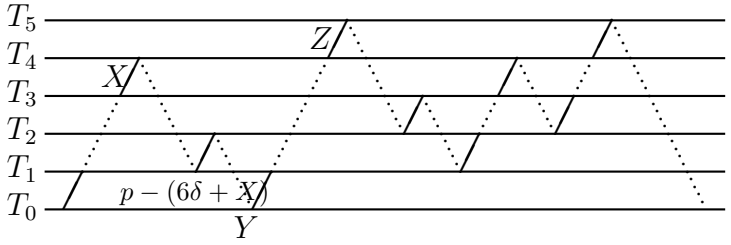

Figure 5.12 – Cycle (0310421324)

Entre le moment où le produit entre dans la cuve T_4 et le moment où il en sort,

il doit se passer exactement p unités de temps. Nous pouvons en déduire que :

$$\begin{array}{lcl} Z & = & p-(10\delta+p-(6\delta+X)+Y) \\ & = & X-Y-4\delta \text{ or } Y \geq 0 \\ & \text{donc} & \\ Z & \leq & X-4\delta \text{ or } X \leq p-10\delta \\ & \leq & p-14\delta \\ Z & < & 0 \end{array}$$

Comme les temps d'attente sont forcément positifs, ce cycle n'est pas réalisable dans l'intervalle considéré.

– (0213240314), temps de cycle : $4p+10\delta$

Tous ces temps de cycle sont supérieurs à $4p/3+14\delta/3$ dans l'intervalle étudié, donc $C_3(3)$ domine strictement tous les 2-cycles réalisables.

5.4.3 Dominance de $C_3(3)$ sur tous les 3-cycles

Si l'on utilise un algorithme de parcours de graphe sur le *line-graph* on peut rapidement obtenir les 21 3-cycles qui ne passent pas plus d'une fois par un sommet de E_4 (ensemble des sommets où la ligne contient au plus un seul produit). Le tableau 5.2 donne cet ensemble de cycles ainsi que leurs temps de cycles.

Si nous comparons tous les temps de cycle, nous pouvons déduire que le cycle $C_3(3)$=(0, 3, 2, 1, 4, 3, 2, 0, 4, 3, 1, 0, 4, 2, 1) en gris dans le tableau 5.2 est le cycle dominant.

Pour les cycles de degré supérieur, une méthode identique peut difficilement être envisagé vu la combinatoire du problème. En effet le nombre total de 4-cycles est de 60648 [12], même si nous supprimons les cycles qui ne sont pas faisables le nombre restera trés élevé pour le faire de manière non-automatisée.

Tableau 5.2 – 3-cycles pour $m = 4$ et $p \in [10\delta, 12\delta[$

Cycle	temps de cycle
0, 1, 2, 0, 3, 1, 4, 0, 2, 1, 3, 2, 4, 3, 4	$(8p + 20\delta)/3$
0, 1, 2, 0, 3, 1, 0, 4, 2, 1, 3, 2, 4, 3, 4	$(7p + 20\delta)/3$
0, 1, 0, 2, 1, 3, 2, 4, 0, 3, 1, 4, 2, 3, 4	$(8p + 20\delta)/3$
0, 1, 0, 2, 1, 3, 2, 0, 4, 3, 1, 4, 2, 3, 4	$(7p + 20\delta)/3$
0, 1, 0, 2, 1, 3, 0, 2, 4, 1, 3, 2, 4, 3, 4	$(7p + 18\delta)/3$
0, 1, 0, 2, 1, 0, 3, 2, 1, 4, 3, 2, 4, 3, 4	$(6p + 18\delta)/3$
0, 2, 1, 3, 2, 4, 3, 0, 4, 1, 2, 0, 3, 1, 4	$(7p + 18\delta)/3$
0, 2, 1, 3, 2, 4, 0, 3, 1, 4, 2, 3, 0, 4, 1	$(7p + 18\delta)/3$
0, 2, 1, 3, 2, 4, 0, 3, 1, 4, 2, 0, 3, 1, 4	$(6p + 16\delta)/3$
0, 2, 1, 3, 2, 0, 4, 3, 1, 4, 2, 3, 0, 4, 1	$(6p + 18\delta)/3$
0, 2, 1, 3, 2, 0, 4, 3, 1, 4, 2, 0, 3, 1, 4	$(6p + 16\delta)/3$
0, 2, 1, 3, 0, 2, 4, 1, 3, 2, 4, 0, 3, 1, 4	$(5p + 20\delta)/3$
0, 2, 1, 3, 0, 2, 4, 1, 3, 2, 0, 4, 3, 1, 4	$(5p + 18\delta)/3$
0, 2, 1, 0, 3, 2, 1, 4, 3, 2, 4, 3, 0, 4, 1	$(5p + 16\delta)/3$
0, 2, 1, 0, 3, 2, 1, 4, 3, 2, 4, 0, 3, 1, 4	$(5p + 14\delta)/3$
0, 2, 1, 0, 3, 2, 1, 4, 3, 2, 0, 4, 3, 1, 4	$(5p + 14\delta)/3$
0, 3, 1, 4, 2, 0, 3, 1, 0, 4, 2, 1, 3, 2, 4	$(5p + 16\delta)/3$
0, 3, 1, 0, 4, 2, 1, 3, 2, 4, 3, 0, 4, 1, 2	$(6p + 18\delta)/3$
0, 3, 1, 0, 4, 2, 1, 3, 2, 4, 0, 3, 1, 4, 2	$(6p + 16\delta)/3$
0, 3, 1, 0, 4, 2, 1, 3, 2, 0, 4, 3, 1, 4, 2	$(5p + 16\delta)/3$
0, 3, 2, 1, 4, 3, 2, 4, 0, 3, 1, 0, 4, 2, 1	$(5p + 14\delta)/3$
0, 3, 2, 1, 4, 3, 2, 0, 4, 3, 1, 0, 4, 2, 1	$(4p + 14\delta)/3$
0, 4, 2, 1, 3, 2, 0, 4, 3, 1, 0, 4, 2, 1, 3	$(5p + 16\delta)/3$
0, 2, 4, 1, 3, 2, 0, 4, 3, 1, 0, 4, 2, 1, 3	$(5p + 14\delta)/3$

5.5 Récapitulatif

Le tableau 5.4 indique les cycles optimaux, pour une ligne équilibrée sans-attente à quatre cuves.

Tableau 5.4 – Cycles optimaux pour $m = 4$ et sans attente sur toutes les cuves

p	$[0, 4\delta[$	$[4\delta, 6\delta[$	$[6\delta, 8\delta[$	$[8\delta, 10\delta[$	$[10\delta, 12\delta[$	$\geq 12\delta$
Cycle optimal	01234	0102132434	03142	02413	043104210321432	04321
Temps de cycle	$4p + 10\delta$	$5p/2 + 7\delta$	$2p + 6\delta$	$3p/2 + 4\delta$	$4p/3 + 14\delta/3$	$p + 4\delta$
Degré du cycle	1	2	1	1	3	1

Pour résumer, on peut tracer l'évolution des temps de cycle pour des lignes à deux, trois et quatre cuves en fonction des temps de trempe (figure 5.13). Pour la représentation nous avons pris $\delta = 1$.

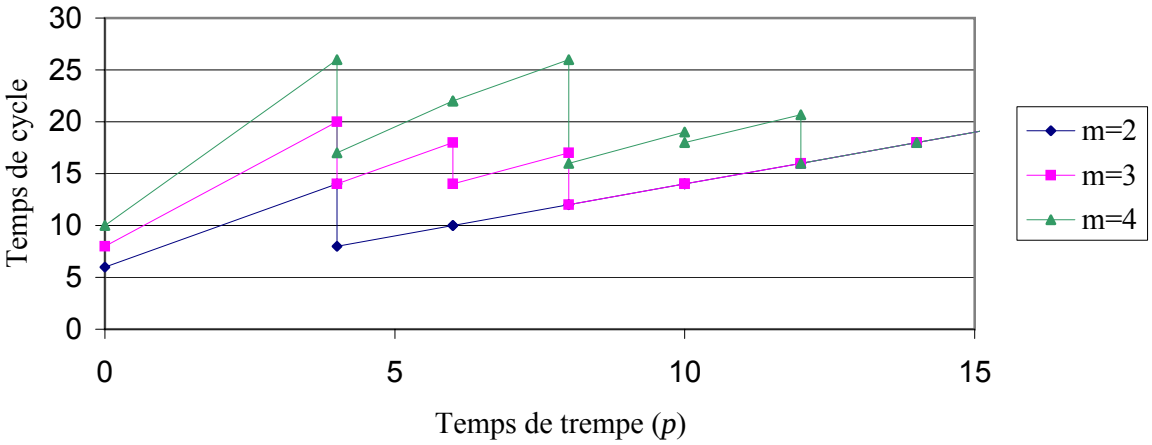

Figure 5.13 – Temps de cycle des cycles optimaux pour 2,3 et 4 cuves

5.6 Limites de l'approche par les graphes

Dans ce chapitre, nous avons vu que dans le cas sans attente équilibré, pour une ligne à quatre cuves, la conjecture d'Agnetis semble encore valide. Pour cela nous avons utilisé une nouvelle fois une approche par les *line-graphs*.

La méthode utilisant les *line-graphs* convient assez bien pour les lignes contenant peu de cuves, mais on peut logiquement penser que pour un nombre plus important de cuves, la taille du graphe deviendra trop importante pour que l'on puisse correctement l'utiliser, surtout pour les grandes valeurs de p pour lesquels le nombre d'arcs retirés est faible.

Pour calculer le nombre d'arcs dans un graphe, nous avons développé un algorithme qui retourne la variable `nbarc` (nombre d'arcs total dans le graphe d'état) et une matrice `etat[i][j]` telle que pour tout arc `j` créé, `etat[1][j]` représente l'activité qui fait passer le système de l'état "`etat[2][j]`" à l'état "`etat[3][j]`".

Algorithme [Construction du graphe d'état]

```
donnée m          //nombre de cuves
nbarc=0;          //nombre d'arcs dans le graphe
/*-------------------------------------------------*/
Début
    pour p=0 à p<2^m faire     //p : état de départ en base 10
        k=p;
/*-------------------------------------------------*/
    /*Création du vecteur d'état*/
        //etat : matrice 1*m+2 telle que
           etat[0]=1;   //Cuve de chargement toujours remplie
           etat[m+1]=0; //Cuve de déchargement de capacité infinie
        //etat[i]=0 si la cuve i est vide sinon etat[i]=1
            pour i=1 à i<=m
                etat[i]=k % 2; // reste de la division de k par 2
                k=k/2;
            fin pour
/*-------------------------------------------------*/
    /*Recherche des arcs*/
        pour j=m à j>=0
        //si la cuve j contient un produit et la cuve est j+1 non, alors
        //l'activité j est possible pour transporter un produit de j à j+1
            si ((etat[j]==1) ET  (etat[j+1]==0)) alors
        //Il existe l'activité i (donc un arc) qui fait passer
        //de p à q (état d'arrivé en base 10)
                si (j==0) alors
                    q=p+1;             //etat[0]toujours égale à 1
                fin si
                sinon
                    si(j==m) alors q=p-2^(j-1);
                                       //etat[m+1]toujours égale à 0
                    sinon q=p-2^(j-1)+2^j;       //cas général
                fin sinon
            graphe[1][nbarc]=j;         //activité
            graphe[2][nbarc]=p;         //état de départ
            graphe[3][nbarc]=q;         //état d'arrivé
            incrémenter (nbarc);
        fin pour
    fin pour
    retourner graphe,nbarc;
Fin;
```

Cet algorithme a été programmé en C et permet d'obtenir rapidement le nombre d'arcs dans le graphe d'état pour une valeur de m donnée. Ces résultats sont donnés dans le tableau 5.5.

Nombre de cuves	2	3	4	5	6	7	8	9	10	11	12
Nombre d'arcs	5	12	28	64	144	320	704	1536	3328	7168	15360

Tableau 5.5 – Nombres d'arcs dans le graphe d'état

Comme le nombre d'états possibles dans le système vaut 2^m et qu'il faut au moins un arc qui arrive à chaque sommet (donc un sommet dans le *line-graph*), le nombre de sommet dans le *line-graph* est supérieur à 2^m. La figure 5.14 montre bien que le nombre d'arcs dans le *line-graph* est exponentiel en fonction du nombre de cuve.

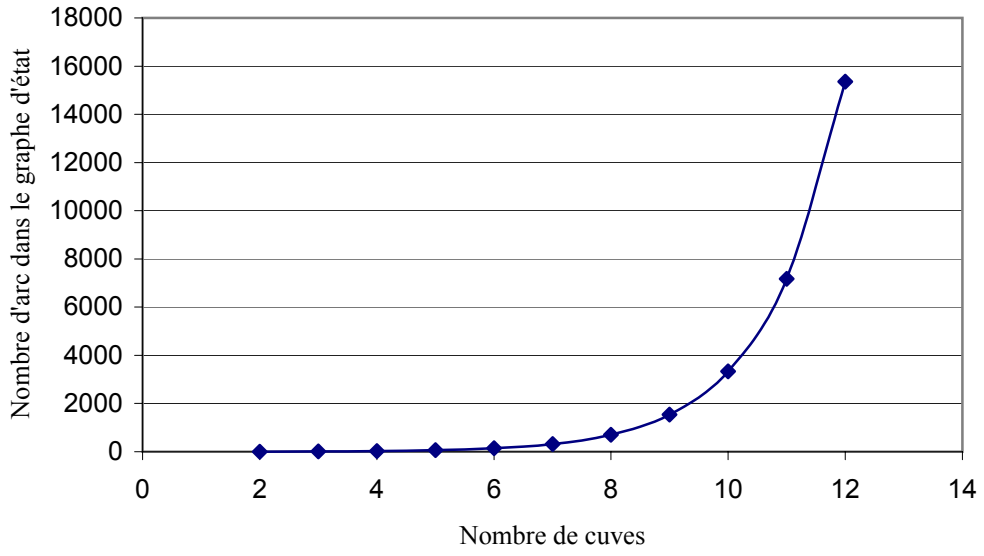

Figure 5.14 – Nombres d'arcs dans le graphe d'état

5.7 Conclusion

Dans ce chapitre, nous avons démontré partiellement que la conjecture sur l'optimalité des 1-, 2- ... $(m-1)$-cycles est encore valide pour 4 cuves ($m = 4$). Pour cela nous avons utiliser les *line-graphs*. Une fois les arcs non réalisables retirés et les séquences non réalisables prises en compte, nous avons pu déterminer les cycles et, en comparant leur temps de cycle, nous avons pu déterminer les cycles optimaux.

Même si l'approche par les *line-graphs* ne peut raisonnablement pas être généralisée à m machines, on peut néanmoins commencer à voir quelles sont les formes des cycles intéressants à considérer. En effet il semble que les degrés des cycles dominants soient croissants en fonction de p et qu'ils augmentent de 1 pour p augmentant de 4δ. Dans le chapitre suivant, nous proposerons une conjecture sur les cycles optimaux basée sur cette idée.

Chapitre 6

Ligne équilibrée, sans attente, à m cuves

Dans ce chapitre, nous étudierons le problème sans attente, équilibré et mono-produit pour une ligne à m cuves (m quelconque). Cette configuration est représentée figure 6.1.

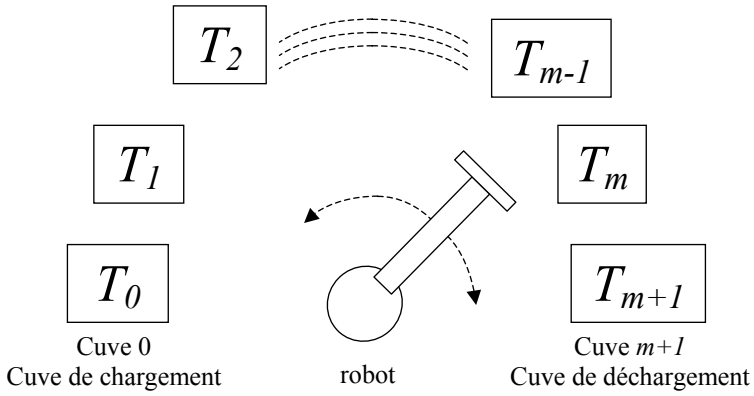

Figure 6.1 – Ligne à m cuves

Nous proposerons deux conjectures sur les cycles optimaux : une lorsque le nombre de cuves sur la ligne est impair et une autre pour le cas où le nombre de cuves est pair. Ces conjectures indiquent que les cycles optimaux peuvent être de cinq types différents. Avant de proposer ces conjectures, nous expliciterons en détails les cycles proposés. Enfin nous prouverons les conjectures pour certaines valeurs du temps de trempe[1].

[1]Ces travaux ont été présentés lors de la conférence IEPM'03 [55]

6.1 Définition des cycles

Nous définissons, dans cette partie, les cycles que nous proposons comme cycles optimaux. Nous définissons ici trois α-cycles $C_1(\alpha)$, $C_2(\alpha)$ et $C_3(\alpha)$ et deux 1-cycles C_4 et C_5.

6.1.1 Définition du α-cycle $C_1(\alpha)$

6.1.1.1 Définition

D'un point de vue des mouvements du robot le α-cycle $C_1(\alpha)$ est composé des trois séquences suivantes :

Séquence (1) A partir d'une ligne vide, le cycle $C_1(\alpha)$ dispose α porteurs, consécutivement, dans les cuves $T_1, T_2 \dots T_\alpha$. Ceci revient à réaliser la séquence suivante :
$\underbrace{0}\ \underbrace{10}\ \underbrace{210} \dots \underbrace{i(i-1)\dots 10} \dots \underbrace{(\alpha-1)(\alpha-2)\dots 10}$.

Si on utilise la notation $\prod$ introduite dans le chapitre 2, chaque accolade peut être écrite $\prod_{j=0}^{i}(i-j)$. La séquence entière s'écrit alors :

$\prod_{i=0}^{\alpha-1}\left[\prod_{j=0}^{i}(i-j)\right]$.

Séquence (2) Le robot transporte tous les porteurs jusqu'à la fin de la ligne. Ceci constitue la séquence suivante :
$\underbrace{\alpha(\alpha-1)\dots 1}\ \underbrace{(\alpha+2)(\alpha+1)\alpha\dots 2} \dots \underbrace{m(m-1)\dots(m-\alpha+1)}$.

Chaque accolade peut être aussi écrite $\prod_{j=0}^{\alpha-1}(i-j)$. Ce qui fait que la séquence complète peut être décrite par l'expression suivante :

$\prod_{i=\alpha}^{m}\left[\prod_{j=0}^{\alpha-1}(i-j)\right]$.

Séquence (3) Enfin, le robot vide entièrement la ligne en faisant sortir tous les produits. Les activités que réalise le robot sont donc :
$\underbrace{m(m-1)\dots(m-\alpha+2)} \dots \underbrace{m(m-1)\dots(m-j+1)} \dots \underbrace{m(m-1)}\ m$.

Les accolades peuvent donc s'écrire $\prod_{j=1}^{\alpha-i}(m-j+1)$. La séquence complète peut alors être définie de la manière suivante :

$\prod_{i=1}^{\alpha-1}\Big[\prod_{j=1}^{\alpha-i}(m-j+1)\Big]$.

Si l'on rassemble les différentes séquences ci-dessus, le α-cycle $C_1(\alpha)$ peut être défini par l'équation 6.1.

$$C_1(\alpha) = \underbrace{\prod_{i=0}^{\alpha-1}\left[\prod_{j=0}^{i}(i-j)\right]}_{(1)}\underbrace{\prod_{i=\alpha}^{m}\left[\prod_{j=0}^{\alpha-1}(i-j)\right]}_{(2)}\underbrace{\prod_{i=1}^{\alpha-1}\left[\prod_{j=1}^{\alpha-i}(m-j+1)\right]}_{(3)} \quad (6.1)$$

Par exemple, pour une ligne avec cinq cuves ($m = 5$), le cycle $C_1(2) = (010213243545)$ est représenté figure 6.2. Il consiste à faire entrer deux porteurs sur la ligne, les faire avancer à tour de rôle, puis à les faire sortir de la ligne.

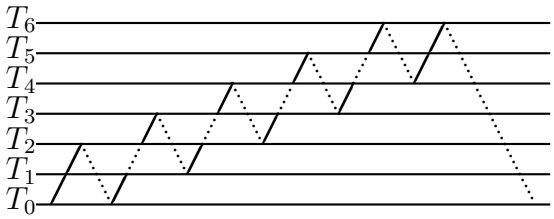

Figure 6.2 – $C_1(2)$ pour $m = 5$

6.1.1.2 Conditions de faisabilité

Regardons maintenant quelles sont les conditions sur le temps de trempe p pour que le cycle $C_1(\alpha)$ soit réalisable. Le nombre maximum d'activités entre une activité β et l'activité $\beta + 1$ consécutive est atteint durant la deuxième partie du cycle. La figure 6.3 donne un exemple des mouvements à réaliser durant cette partie du cycle.

Après le dépôt du porteur, le robot doit aller retirer le porteur dans la cuve $\beta - 1$. Ceci nécessite 3δ unités de temps (déplacements à vide et sous charge). Puis, le robot fait la même chose jusqu'au dernier porteur sur la ligne (cuve T_{min}). Ce qui fait un temps de $3(\beta - min - 1)\delta$ unités de temps.

Ensuite, le robot doit retirer le porteur le plus en avant sur la ligne (cuve T_{max}) soit $(max - min - 1)\delta$.

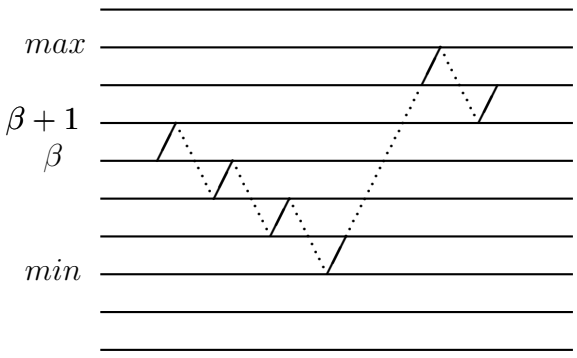

Figure 6.3 – Conditions de faisabilité du cycle $C_1(\alpha)$

Enfin, le robot doit retirer tous les porteurs les plus en avant sur la ligne jusqu'au retrait du produit dans la cuve $\beta + 1$. Ces déplacements prennent donc $3(max - \beta)\delta$. Donc le temps total, entre le dépôt et le retrait dans la cuve $\beta + 1$, est de $3(\beta - min - 1)\delta + (max - min - 1)\delta + 3(max - \beta)\delta$. Ce qui donne un temps de $4(max - min - 1)\delta$, or $max - min = \alpha$. Donc, pour que le cycle soit réalisable, il faut que le temps de trempe p soit supérieur à $4(\alpha - 1)\delta$.

6.1.1.3 Temps de cycle

Pour calculer la longueur du cycle nous regardons le premier produit qui rentre sur la ligne. Ce même produit va sortir de la ligne durant ce cycle. Ceci dure $mp+(m+1)\delta$ unités de temps. Un fois le produit sorti le robot doit transporter le second produit de la cuve T_m à la cuve T_{m+1} (3δ unités de temps). Ensuite il va déplacer les autres produits avant de sortir de la ligne le second produit $(p + \delta)$. En tenant le même raisonnement pour chaque produit, on peut en déduire qu'il faut $p + 4\delta$ unités de temps pour chacun des produits 2 à α. Si l'on rajoute le temps pour que le robot revienne à la cuve 0 en fin de cycle ($(m + 1)\delta$ unités de temps), nous obtenons une longueur de : $mp + (m + 1)\delta + (\alpha - 1)(p + 4\delta) + (m + 1)\delta$.

Pour obtenir le temps de cycle de $C_1(\alpha)$, il suffit de diviser par le degré α. Pour une ligne à m cuves, le temps de cycle de $C_1(\alpha)$ est donc :

$$T(C_1(\alpha)) = \frac{1}{\alpha}[(m + \alpha - 1)p + 2(m + 2\alpha - 1)\delta]$$

6.1.2 Définition du α-cycle $C_2(\alpha)$

6.1.2.1 Définition

Pour aider à la compréhension, le cycle $C_2(4)$ pour $m = 5$ sera pris comme exemple. Le cycle $C_2(\alpha)$ est composé des mouvements de robot suivants :

Séquence (1) Une série de α produits entre sur la ligne pendant que la série précédente est en train de sortir. La formulation générale est donnée par l'équation (6.2).

$$\prod_{i=0}^{2(\alpha-1)-m} [i(i-1)\dots 0\ m(m-1)\dots(m-(2(\alpha-1)-m)+i)] \tag{6.2}$$

Pour l'exemple $C_2(4)$ cela représente la séquence 054 105 (figure 6.4).

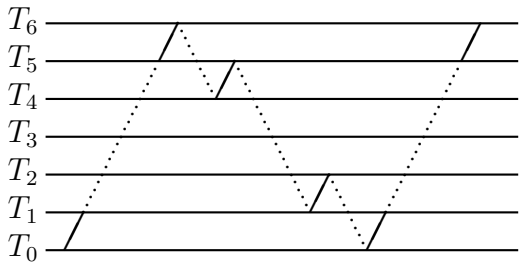

Figure 6.4 – Première partie du cycle $C_2(4)$ pour $m = 5$

Séquence (2) Le robot termine de faire rentrer la série de α produits alors que la série précédente est totalement sortie. L'équation (6.3) en est la formulation.

$$\prod_{i=2\alpha-m-1}^{\alpha-1} [i(i-1)\dots 0] \tag{6.3}$$

Dans l'exemple proposé cela revient à la séquence 210 3210 (figure 6.5).

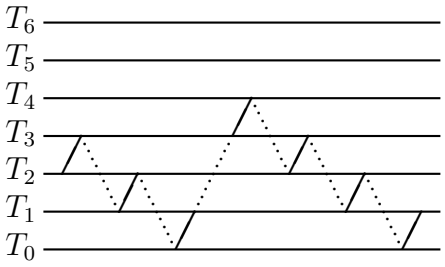

Figure 6.5 – Deuxième partie du cycle $C_2(4)$ pour $m = 5$

Séquence (3) La série des α produits est traitée jusqu'à ce que le premier produit arrive dans la dernière cuve. Cette séquence est définie par l'équation (6.4).

$$\prod_{i=\alpha}^{m}[i(i-1)\dots(i-\alpha+1)] \tag{6.4}$$

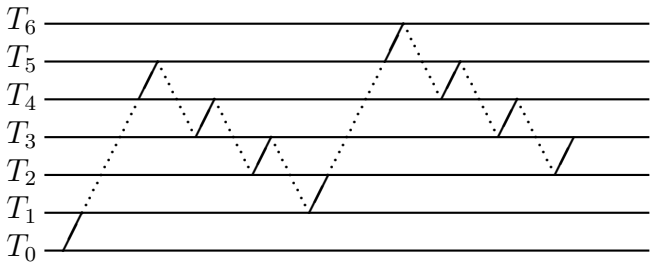

Figure 6.6 – Troisième partie du cycle $C_2(4)$ pour $m = 5$

Dans notre exemple, la séquence est alors : 4321 5432 qui est représentée figure 6.6.

Séquence (4) Les produits continuent de quitter la ligne jusqu'à ce qu'il y en ait suffisamment de sortie pour que le robot puisse en faire rentrer de nouveau. Pour le cycle $C_2(\alpha)$, une nouvelle série entre sur la ligne alors qu'il reste $2\alpha - m - 1$ produits. Les conditions de faisabilité qui seront développées dans la suite montrent que les séquences précédentes nécessitent que p soit supérieur à $4(\alpha - 1)\delta$. Dans ces conditions, une nouvelle série de pièce peut rentrer alors qu'il reste $2\alpha - m - 1$ produits. On obtient alors l'équation (6.5).

$$\prod_{i=\alpha-2}^{2\alpha-m-1}[m(m-1)\dots(m-i)] \tag{6.5}$$

Pour le cycle $C_2(4)$ et $m = 5$, nous obtenons la séquence 543, représentée figure 6.7. Après l'activité 3, il reste alors deux produits sur la ligne.

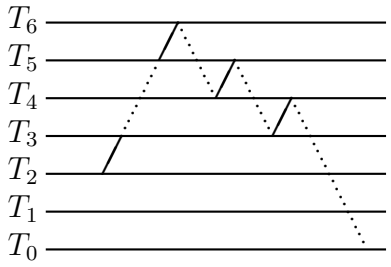

Figure 6.7 – Quatrième partie du cycle $C_2(4)$ pour $m = 5$

Après concaténation des quatre séquences, nous obtenons la formulation du cycle $C_2(\alpha)$:

$$\begin{aligned} C_2(\alpha) \;=\; & \prod_{i=0}^{2(\alpha-1)-m} [i(i-1)\dots 0\; m(m-1)\dots(m-(2(\alpha-1)-m)+i)] \\ & \prod_{i=2\alpha-m-1}^{\alpha-1} [i(i-1)\dots 0] \\ & \prod_{i=\alpha}^{m} [i(i-1)\dots(i-\alpha+1)] \\ & \prod_{i=\alpha-2}^{2\alpha-m-1} [m(m-1)\dots(m-i)] \end{aligned}$$

Une manière plus condensée d'écrire le cycle $C_2(\alpha)$ est donné par l'expression (6.6).

$$\begin{aligned} C_2(\alpha) \;=\; & \prod_{i=0}^{2(\alpha-1)-m} \left[\prod_{j=0}^{i}(i-j) \prod_{j=0}^{2(\alpha-1)-m-i}(m-j)\right] \prod_{i=2\alpha-m-1}^{\alpha-1} \left[\prod_{j=0}^{i}(i-j)\right] * \\ & \prod_{i=\alpha}^{m} \left[\prod_{j=0}^{\alpha-1}(i-j)\right] \prod_{i=\alpha-2}^{2\alpha-m-1} \left[\prod_{j=0}^{i}(m-j)\right] \end{aligned} \tag{6.6}$$

Dans l'exemple le 4-cycle $C_2(4)$, pour une ligne à cinq cuves, est composé des activités 0541052103 21043215432543. Il est représenté figure 6.8.

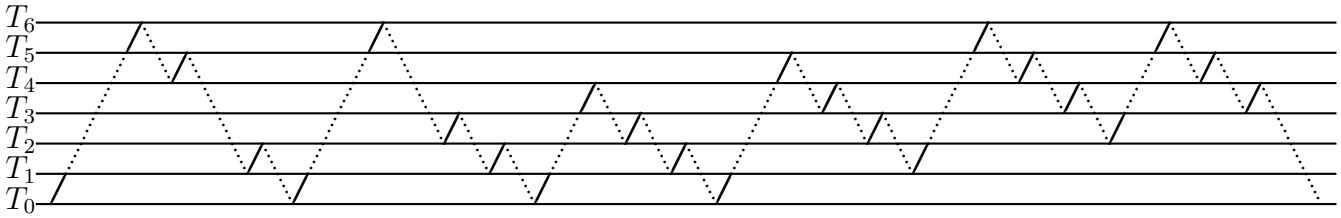

Figure 6.8 – $C_2(4)$ pour $m = 5$

6.1.2.2 Conditions de faisabilité

Nous allons étudier la première partie du cycle :

$\prod_{i=0}^{2(\alpha-1)-m}[i(i-1)\dots 0\ m(m-1)\dots(m-(2(\alpha-1)-m)+i)]$.

Considérons un produit qui, faisant partie de la série en train de sortir, sort de la cuve T_k par l'activité k. Regardons tous les mouvements réalisés jusqu'à l'activité $k+1$ suivante.

- Déplacement des produits de la cuve $k-1$ à la cuve $(m-(2(\alpha-1)-m)+i) = (2(m+\alpha-1)+i) : 3(k-2(m-\alpha+1)-i)\delta$ unités de temps.

- Déplacement à vide de la cuve $2(m+\alpha-1)+i$ à la cuve $i : 2(m-\alpha+1)\delta$ unités de temps.

- Déplacement des produits, de celui qui se trouve dans la cuve i jusqu'au produit qui se trouve dans la cuve $0 : 3i\delta$ unités de temps.

- Remontée à vide jusqu'à $m : m\delta$ unités de temps.

- Déplacement des produits de celui qui se trouve dans la cuve m jusqu'au produit qui se trouve dans la cuve $k : (3(m-k)-1)\delta$ unités de temps.

Si l'on somme tous les temps de transport, on obtient la condition de faisabilité suivante :

$[3(k-2(m-\alpha+1)-i)+2(m-\alpha+1)+3i+m+3(m-k)-1]\delta = 4(\alpha-1)\delta$.

Considérons, maintenant, un produit qui, faisant partie de la série en train de rentrer, sort de la cuve k' par l'activité k'. Regardons tous les mouvements réalisés jusqu'à l'activité $k'+1$ suivante.

- Déplacement des produits de la cuve $k'-1$ à la cuve $0 : 3(k'-1)\delta$ unités de temps.

– Remontée à vide jusqu'à m : $m\delta$ unités de temps.

– Déplacement des produits, de celui qui se trouve dans la cuve m jusqu'au produit qui se trouve dans la cuve $2(m+\alpha-1)+i$: $3(2\alpha-m-i-2)\delta$ unités de temps.

– Déplacement à vide de la cuve $2(m+\alpha-1)+i$ à la cuve i : $2(m-\alpha+1)\delta$ unités de temps.

– Déplacement des produits de celui qui se trouve dans la cuve i jusqu'au produit qui se trouve dans la cuve k : $3(i-k')\delta$ unités de temps.

Si l'on somme tous les temps de transport, on obtient la borne inférieure sur p suivante :

$$[3(k'-1)+m+3(2\alpha-m-i-2)+2(m-\alpha+1)+3(i-k')]\delta = 4(\alpha-1)\delta.$$

Les conditions de faisabilité des deuxième, troisième et quatrième parties peuvent être calculées de la même façon que pour les cycles $C_1(\alpha)$. Donc pour que ces parties soient réalisables, il faut que p soit supérieur à $4(\alpha-1)\delta$.

Nous pouvons donc en déduire que, pour que le cycle $C_2(\alpha)$ soit réalisable, il faut que p soit supérieur à $4(\alpha-1)\delta$ unités de temps.

6.1.2.3 Temps de cycle

Le calcul du temps de cycle se réalise de la manière suivante. Pour que la périodicité au niveau des temps d'attente soit d'un cycle, il suffit de mettre tous les temps d'attente sur la dernière cuve. Pour la première séquence cela consiste à mettre un temps d'attente sur la dernière cuve de :

– $2m\delta$: Descente jusqu'à la cuve T_0
– $2(2\alpha-m-2)\delta$: 2δ pour chaque produit sorti durant la descente.

Ce qui fait un total de $4(\alpha-1)\delta$. Si l'on fait de même pour toutes les séquences et que l'on fait la somme des temps d'attente et des temps de transport, on obtient un temps de cycle de $C_2(\alpha)$:

$$T(C_2(\alpha)) = \tfrac{1}{\alpha}[(2m-\alpha)p + 4m\delta].$$

6.1.3 Définition du α-cycle $C_3(\alpha)$

6.1.3.1 Définition

Le α-cycle $C_3(\alpha)$ fonctionne sur le même principe que le α-cycle $C_2(\alpha)$. La différence vient du fait qu'une nouvelle série de α produits entre sur la ligne alors qu'il reste un produit de plus. C'est à dire, quand nous avons défini le cycle $C_2(\alpha)$, nous avons posé qu'une nouvelle série débutait alors qu'il restait $2\alpha-m-1$ produits sur la ligne. Pour le cycle $C_3(\alpha)$, une nouvelle série rentre sur la ligne alors qu'il reste $2\alpha - m$ produits. La notation des cycles $C_3(\alpha)$ diffère donc très peu. La partie encadrée dans l'équation qui suit indique la différence entre les deux cycles.

Ceci fait que les α-cycle $C_3(\alpha)$ sont définis de la manière suivante :

$$\begin{aligned} C_3(\alpha) \quad = \quad & \prod_{i=0}^{|\overline{2\alpha-m-1}|} [i(i-1)\dots 0\ m(m-1)\dots(m-(2\alpha-m-1)+i))] \\ & \prod_{|\overline{i=2\alpha-m}|}^{\alpha-1} [i(i-1)\dots 0] \\ & \prod_{i=\alpha}^{m} [i(i-1)\dots(i-\alpha+1)] \\ & \prod_{i=\alpha-2}^{|\overline{2\alpha-m}|} [m(m-1)\dots(m-i)] \text{ (lorsque } \alpha \neq m-1) \end{aligned}$$

$$C_3(\alpha) \quad = \quad \prod_{i=0}^{2\alpha-m-1} \left[\prod_{j=0}^{i} (i-j) * \prod_{j=0}^{2\alpha-m-1-i} (m-j) \right] \prod_{i=2\alpha-m}^{\alpha-1} \left[\prod_{j=0}^{i} (i-j) \right] \prod_{i=\alpha}^{m} \left[\prod_{j=0}^{\alpha-1} (i-j) \right] \underbrace{\prod_{i=\alpha-2}^{2\alpha-m} \left[\prod_{j=0}^{i} (m-j) \right]}_{\text{(lorsque } \alpha \neq m-1)}$$

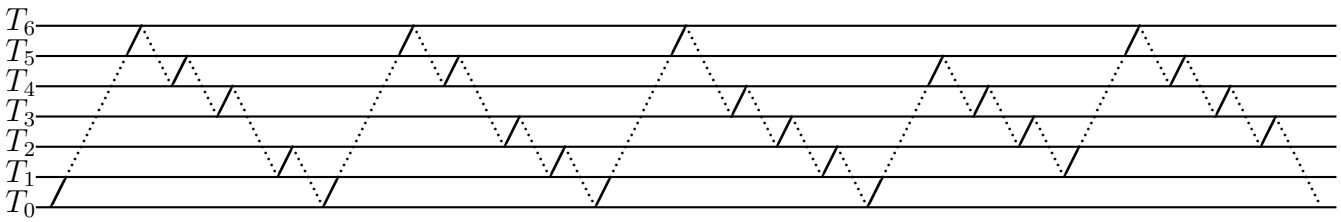

Figure 6.9 – $C_3(4)$ pour $m = 5$

6.1.3.2 Conditions de faisabilité

Le même raisonnement que pour le α-cycle $C_2(\alpha)$, sur la première partie du cycle, peut être tenu. Il montre que, pour que le α-cycle $C_3(\alpha)$ soit réalisable, il faut que p soit supérieur à $4(\alpha-1)\delta + 2\delta$.

6.1.3.3 Temps de cycle

Là encore, le même raisonnement que pour les cycles $C_2(\alpha)$ peut être tenu. Le temps de cycle de $C_3(\alpha)$ est donc :

$$T(C_3(\alpha)) = \frac{1}{\alpha}[(2m - \alpha - 1)p + (4m - 2)\delta]$$

6.1.4 Définition du 1-cycles C_4

6.1.4.1 Définition

Le 1-cycle C_4 est défini, pour un nombre pair de cuves m, de la manière suivante :

$$C_4 = \prod_{i=0}^{m/2} 2i \prod_{i=0}^{m/2-1} (2i + 1)$$

Le cycle C_4 peut être décrit à partir de l'état où une cuve sur deux contient un porteur (0,1,0,1...0,1). De cet état, le robot fait rentrer un nouveau porteur sur la ligne. Puis, le porteur suivant est déplacé, ainsi de suite jusqu'au dernier qui sort de la ligne. La ligne est alors dans l'état (1,0,1,0...1,0). Le robot déplace alors le porteur qui se trouve dans la cuve T_1 puis celui dans la cuve T_3 etc... La ligne revient alors dans l'état initial et le robot peut recommencer. La figure 6.10 donne un exemple de C_4 pour une ligne à quatre cuves.

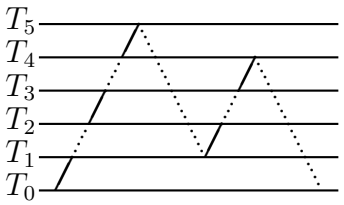

Figure 6.10 – Exemple de C_4 pour $m = 4$

6.1.4.2 Conditions de faisabilité

Pour étudier les conditions de faisabilité, nous allons regarder les déplacements du robot entre une activité β et l'activité $\beta + 1$ suivante.

Si β est pair, le robot doit aller jusqu'à la cuve T_m, afin de réaliser l'activité m. Ceci prend $(m - \beta)\delta$ unités de temps. Ensuite il doit déplacer les porteurs qui se trouvent dans les cuves impaires, de la cuve T_1 jusqu'à ce qu'il réalise l'activité $\beta+1$. Il se passe donc $m\delta$ unités de temps pour redescendre jusqu'à la cuve T_1, puis $\beta\delta$ unités de temps pour transporter tous les produits se trouvant en amont de la cuve

T_β. Le temps total, entre le dépôt d'une pièce dans la cuve T_β et sa sortie, est de $2m\delta$.

Si β est impair, le robot doit aller jusqu'à la cuve T_{m-1}, afin de réaliser l'activité $m-1$. Ce qui prend $(m-\beta-1)\delta$ unités de temps. Ensuite, il doit déplacer les porteurs qui se trouvent dans les cuves paires de la cuve T_0 jusqu'à ce qu'il réalise l'activité $\beta+1$. Il se passe donc $m\delta$ unités de temps pour redescendre jusqu'à la cuve T_0, puis $\beta\delta$ unités de temps pour transporter tous les produits se trouvant en amont de la cuve T_β. Le temps total, entre le dépôt d'une pièce dans la cuve T_β et sa sortie, est de $2m\delta$.

Nous pouvons donc en déduire que, pour que le cycle C_4 soit réalisable, il faut que p soit supérieur ou égal à $2m\delta$.

6.1.4.3 Temps de cycle

Pour calculer le temps de cycle de C_4, il est possible de faire attendre le robot $2\frac{p-2m\delta}{m}$ au dessus de chaque cuve à part pour les cuves T_0 et T_1. Cette disposition des temps d'attente permet de revenir à l'état initial sur la ligne au niveau des temps d'attente au bout d'un cycle. Le temps de cycle est alors la somme des temps de transport ($4m\delta$ unités de temps) plus la somme des temps d'attente ($2\frac{p-2m\delta}{m}(m-1)$).

Le temps de cycle de C_4 est donc :

$$T(C_4) = 2\frac{m-1}{m}p + 4\delta$$

6.1.5 Définition du 1-cycles C_5

6.1.5.1 Définition

Les 1-cycles C_5 sont les cycles où toutes les cuves contiennent un produit. Le robot sort alors le produit de la cuve T_m puis déplace le produit qui se trouve dans la cuve T_{m-1} et ainsi de suite jusqu'à faire rentrer un nouveau produit sur la ligne. L'expression de C_5 est donc $C_5 = \prod_{i=0}^{m}(m-i)$ ou, si l'on souhaite l'exprimer à partir d'une activité 0 :

$$C_5 = 0\prod_{i=0}^{m-1}(m-i)$$

La figure 6.11 représente deux occurrences du cycle C_5 pour $m = 5$.

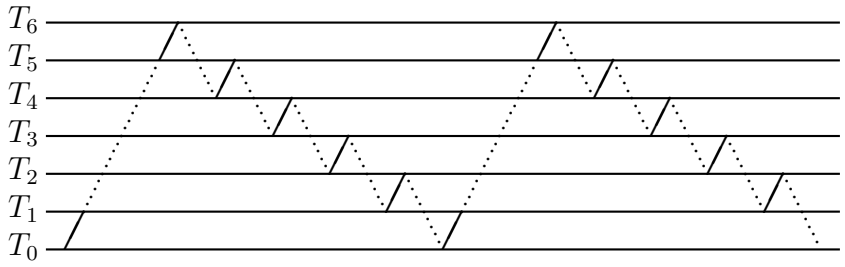

Figure 6.11 – Cycle C_5 pour $m = 5$

6.1.5.2 Conditions de faisabilité

Si nous regardons les déplacements du robot entre le moment où le robot dépose un produit dans une cuve T_X et le moment où il le ressort, le robot doit effectuer les activités suivantes :

- Il doit déplacer les produits qui se trouvent dans les cuves en aval sur la ligne. Ceci nécessite $3X\delta$ (2δ pour les déplacements à vide et δ pour l'activité elle même).
- Il doit retourner au dessus de la cuve T_m, ce qui prend $(m-1)\delta$ unités de temps.
- Il doit enfin déplacer tous les produits qui se situent entre la cuve T_m et la cuve T_X. Ces déplacements vont durer $3(m-X-1)\delta$.

Au total il se sera passé $4(m-1)\delta$ unités de temps entre le dépôt et la sortie du produit dans la cuve T_X. Il faut donc que le temps de trempe soit supérieur à cette valeur pour que le cycle soit réalisable.

6.1.5.3 Temps de cycle

Pour calculer le temps de cycle de C_5, il suffit de mettre un temps d'attente pour le robot de $p - 4(m-1)\delta$ au dessus de la cuve T_m. Tous les autres temps d'attente valent alors 0, le cycle se retrouve alors dans la situation initiale en début du cycle suivant. Le temps de cycle de C_5 est alors la somme des temps de transport ($4m\delta$) plus la somme des temps d'attente ($p - 4(m-1)\delta$). Le temps de cycle vaut alors

$$T(C_5) = p + 4\delta$$

.

6.2 Conjecture sur les cycles optimaux ($m \geq 5$)

6.2.1 Conjecture sur les cycles optimaux avec m impair

Conjecture 2 *Pour une ligne, sans attente, équilibrée à m cuves (m impair), k est défini tel que* $p \in [4(k-1)\delta, 4k\delta[$. *Les cycles optimaux sont les suivants :*

- $k \leq \frac{m-1}{2}$ *: le k-cycle* $C_1(k)$ *est optimal ;*

- $k = \frac{m+1}{2}$ *:*
  - *Si* $p \in [4(k-1)\delta, 4(k-1)\delta + 2\delta[$ *le k-cycle* $C_1(k)$ *est optimal ;*
  - *Si* $p \in [4(k-1)\delta + 2\delta, 4k\delta[$ *le k-cycle* $C_3(k)$ *est optimal ;*

- $\frac{m+1}{2} < k < m-1$ *:*
  - *Si* $p \in [4(k-1)\delta, 4(k-1)\delta + 2\delta[$, *le k-cycle* $C_2(k)$ *est optimal.*
  - *Si* $p \in [4(k-1)\delta + 2\delta, 4k\delta[$, *le k-cycle* $C_3(k)$ *est optimal.*

- *pour* $k \geq m-1$, *le 1-cycle* $C_5 = \prod_m^{i=0}(m-i)$ *est optimal.*

On remarque que pour tout $k \leq m-1$ il existe une configuration tel que le cycle optimal soit un k-cycle, ce qui confirmerait la conjecture d'Agnetis [1].

6.2.2 Conjecture sur les cycles optimaux avec m pair

Conjecture 3 *Pour une ligne, sans attente, équilibrée à m cuves (m pair), k est défini tel que* $p \in [4(k-1)\delta, 4k\delta[$. *Les cycles optimaux sont les suivants :*

- $k < \frac{m}{2}$ *: le k-cycle* $C_1(k)$ *est optimal ;*

- $\frac{m}{2} \leq k \leq m-1$ *:*
  - *Si* $p \in [4(k-1)\delta, 4(k-1)\delta + 2\delta[$ *, le meilleur cycle entre le* 1*-cycle* C_4 *et le k-cycle* $C_2(k)$ *est optimal.*
  - *pour* $p \in [4(k-1)\delta + 2\delta, 4k\delta[$, *le meilleur cycle entre le* 1*-cycle* C_4 *et le k-cycle* $C_3(k)$ *est optimal.*

- *pour* $k \geq m-1$, *le 1-cycle* $C_5 = \prod_m^{i=0}(m-i)$ *est optimal.*

6.2.3 Retour sur $m = 2$, $m = 3$ et $m = 4$

Les conjectures 2 et 3 sont valides dans les cas $m = 2$ et $m = 3$. En effet, si nous comparons les tableaux 6.1 et 6.2 obtenus par la conjecture et les premières lignes des tableaux 3.1 et 4.5 proposés précédemment, nous retrouvons des résultats identiques.

Tableau 6.1 – Cycles optimaux pour une ligne équilibrée, sans attente, à deux cuves

Temps de trempe	$[0, 4\delta[$	$\geq 4\delta$
Cycle	$C_1(1)$	C_5
	0,1,2	0, 2, 1

Tableau 6.2 – Cycles optimaux pour une ligne équilibrée, sans attente, à trois cuves

Temps de trempe	$[0, 4\delta[$	$[4\delta, 6\delta[$	$[6\delta, 8\delta[$	$\geq 8\delta$
Cycle	$C_1(1)$	$C_2(2)$	$C_3(2)$	C_5
	0,1,2,3	0,2,1,3,2,3,0,1	0,2,1,3,2,0,3,1	0, 3, 2, 1

En revanche, la conjecture pour m pair n'est pas valide pour $m = 4$ dans l'intervalle $[6\delta; 8\delta[$, c'est à dire dans l'intervalle $[4(\frac{m}{2} - 1) + 2\delta; 4\frac{m}{2}[$. En effet, le 1-cycle $(0 \prod_{i=1}^{m/2}[(m/2 + i)i])$, dont le temps de cycle est $\frac{m}{2}p + (m + 2)\delta$, est dominant pour $m = 4$. La figure représente ce 1-cycle pour $m = 6$.

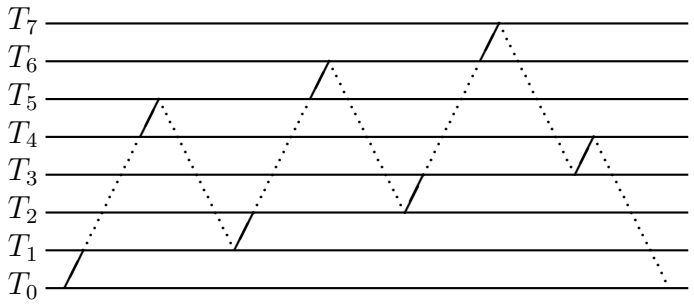

Figure 6.12 – 1-cycle $0 \prod_{i=1}^{m/2}[(m/2 + i)i]$ pour $m = 6$

Quand on compare ce temps de cycle avec le temps de cycle $T(C_1(\frac{m}{2})) = (3 - 2/m)p + (8 - 4/m)\delta$ comme $m/2 \geq 3 - 2/m$ et $m + 2 \geq 8 - 4/m$ pour $m \geq 6$, on obtient alors que $C_1(\frac{m}{2})$ domine ce cycle pour $m \geq 6$.

6.3 Preuve de la conjecture dans certaines configurations

Dans cette section, nous allons prouver la conjecture lorsque le temps de trempe est inférieur à $(m+2)\delta$ unités de temps. Nous allons aussi démontrer que le $(m-1)$-cycle proposé lorsque $p \in [(4(m-1)-2)\delta; 4(m-1)\delta[$ domine tous les k-cycles ($k \leq m-1$).

6.3.1 Cycles optimaux pour $k \leq \frac{m+2}{4}$

Dans cette configuration ($k \leq \frac{m+2}{4}$ ce qui revient à $p < (m+2)\delta$) la conjecture est la même quelle que soit la parité sur le nombre de cuves sur la ligne. Le théorème suivant s'applique donc pour m pair et impair.

Théorème 1 *Pour $p \in [4(k-1)\delta, 4k\delta[$ et $k \leq \frac{m+2}{4}$ le k-cycle $C_1(k)$ est optimal.*

Preuve : La preuve de l'optimalité va être basée sur la méthode suivante. Nous allons montrer, dans un premier temps, que pour $p < (m+2)\delta$ il est nécessaire de vider la ligne au moins une fois par cycle. Dans un second temps, nous allons montrer qu'avec cette condition, nous obtenons une borne inférieure du temps des cycles réalisables et que les temps de cycle des cycles $C_1(\alpha)$ atteignent cette borne.

Quand $p \in [4(k-1)\delta; 4k\delta[$, il y a au plus k produits qui sont traités en même temps sur la ligne. Regardons quelles sont les conditions qui permettent de ne pas vider la ligne. Soit T_X la cuve qui traite un produit entre le moment où un produit sort de la ligne et qu'un produit suivant entre sur la ligne.

Il faut que le temps de trempe soit suffisamment grand pour que le robot ait le temps d'aller sortir le produit de la dernière cuve et de revenir récupérer le produit dans la cuve T_X donc : $p \geq 2(m-X+1)\delta$. Il faut aussi le temps de trempe soit suffisamment grand pour que le robot puisse faire rentrer un produit : $p \geq 2(X+1)\delta$. Ces conditions sont représentées figure 6.13.

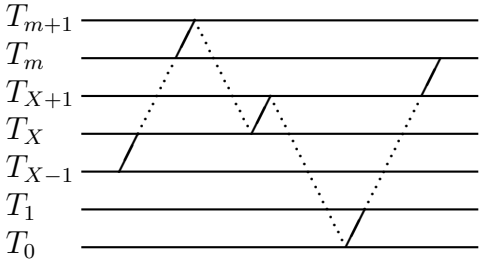

Figure 6.13 – Conditions de faisabilité pour ne pas vider la ligne

Or $p < 4k\delta$, donc nous pouvons en déduire que : $4k\delta > 2(m-X+1)\delta$.

Donc X doit vérifier l'inégalité suivante : $X > m+1-2k$. De même $4k\delta > 2(X+1)\delta$, ce qui implique que X doit vérifier l'inégalité suivante : $X < 2k-1$. Nous obtenons donc la double inégalité $2k-1 > X > m+1-2k$. Pour que la ligne ne nécessite pas d'être vidée, il faut donc que $2k-1 > m+1-2k$, c'est à dire $k > \frac{m+2}{4}$.

Donc, dans le cas où $k \leq \frac{m+2}{4}$, la ligne doit être vidée durant chaque cycle. Ceci implique que pour tout α-cycle, les α produits qui entrent sur la ligne sont ceux qui la quittent.

Ensuite quel que soit $\alpha > k$, tout α-cycle est dominé par un β-cycle avec $\beta \leq k$. En effet la propriété 1 dit que tout cycle qui vide la ligne plus d'une fois est dominé par le meilleur sous-cycle. La valeur de p et la remarque précédente imposent donc que la ligne soit vidée, au moins, tous les k produits ce qui signifie que pour un α-cycle ($\alpha > k$) la ligne doit être vidée au moins deux fois. Ainsi tout α-cycle ($\alpha > k$) est dominé par un β-cycle où $\beta \leq k$.

Si l'on considère un α-cycle $C(\alpha)$ avec $\alpha \leq k$. Les activités du robot durant le cycle déplacent au plus α produits en même temps. Ceci nous permet d'obtenir une borne minimale pour $T(C(\alpha))$:

$$\alpha T(C(\alpha)) \geq \underbrace{mp + 2(m+1)\delta}_{\text{premier produit qui entre sur la ligne}} + \underbrace{(p+4\delta)(\alpha-1)}_{\text{sortie des } (\alpha-1) \text{produits restants}}$$

Alors

$$T(C(\alpha)) \geq \frac{1}{\alpha}[(m+\alpha-1)p + 2(m+2\alpha-1)\delta] = f(\alpha)$$

$$f(\alpha) = p + 4\delta + (m-1)\frac{p+2\delta}{\alpha}$$

Comme f est une fonction décroissante de α, le minimum est atteint pour α maximum. Or pour $\alpha > k$, les α-cycles sont dominés, donc f est minimale pour $\alpha = k$. Comme $T(C_1(k))$ atteint cette borne, le k-cycle $C_1(k)$ est donc dominant.

□

6.3.2 Dominance du cycle $C_3(m-1)$ sur les k-cycles ($k \leq m-1$) pour $p \in [4(m-2)\delta + 2\delta; 4(m-1)\delta[$

Dans cette section, nous allons montrer que le $(m-1)$-cycle $C_3(m-1)$ domine les k-cycles ($k \leq m-1$) sur l'intervalle $[4(m-2)\delta + 2\delta; 4(m-1)\delta[$. Pour cela nous allons chercher une borne inférieure et nous allons montrer que $C_3(m-1)$ l'atteint.

Nous utiliserons de nouvelles notation proposées ci-dessous.
- α : degré des cycles ;
- β : nombre de cycles pour traiter entièrement un produit ;
- N_A : nombre d'activité entre l'entrée et la sortie d'un produit ;
- N_p et N_δ : tels que $T(cycle) = N_p p + N_\delta \delta$;
- N_M (*resp* N_m) : nombre maximal (*resp* minimal) de produits en même temps sur la ligne.
- X : Nombre de produit minimal entre la sortie de la ligne du produit considéré et l'entrée sur la ligne du produit équivalent suivant.

Si l'on considère le produit tel que juste avant sa sortie le nombre de produit sur la ligne vaut N_M, le temps de passage de ce produit sur la ligne est de $mp + (m+1)\delta$. Comme il faut réaliser β fois le α-cycle, nous pouvons en déduire que $\alpha\beta T \geq mp + (m+1)\delta$. Entre la sortie du produit et l'entrée du produit suivant équivalent, le nombre de produits sur la ligne va passer de $N_M - 1$ à X. Ceci entraîne qu'il y a aura $(N_M - 1 - X)$ activités 4 et 5 entre les deux moments. Nous pouvons en déduire que $\alpha\beta T \geq mp + (m+1)\delta + (N_M - 1 - X)(p + 4\delta)$. Ensuite le robot va devoir retourner au dessus de la cuve T_0 ce qui prend $(m+2)\delta$. De plus, durant ce retour, il va devoir déplacer les produits sur la ligne ce qui prend 2δ. Comme il doit le faire pour les X produits restants cela implique l'inégalité suivante :

$$\alpha\beta T \geq mp + (m+1)\delta + \underbrace{(N_M - X - 1)}_{(1)}(p + 4\delta) + \underbrace{(m+2)\delta}_{(2)} + \underbrace{2X\delta}_{(3)}$$

avec :
(1) temps d'attente sur les dernières cuves pour passer de N_M à X.
(2) retour à T_0
(3) pertes dues aux produits restant sur la ligne.

Comme il y a au plus N_M produits en même temps sur la ligne, au niveau des activités ceci signifie qu'entre une activité i et l'activité $i+1$ suivante il y a, au plus, $N_M - 1$ activités. Si l'on considère le nombre total d'activités entre l'entrée sur la ligne et la sortie du produit précédemment considéré alors :

Activités	0		1		2	...	m-1		m
Nombre d'activités		$\geq N_M - 1$		$\geq N_M - 1$		...		$\geq N_M - 1$	

On peut donc en déduire que $N_A \leq m(N_M - 1)$. Donc le nombre de cycles nécessaires

pour traiter un produit β est inférieur ou égal à N_A que divise le nombre d'activités d'un α-cycle plus 1. C'est à dire :

$$\begin{aligned}
\beta &\leq \frac{N_A}{\alpha(m+1)} + 1 \\
&\leq \frac{m(N_M - 1)}{\alpha(m+1)} + 1 = \frac{m(N_M - 1) + m + 1}{\alpha(m+1)} \\
&\leq \frac{N_M(m+1) + 1 - N_M}{\alpha(m+1)} \\
\beta &\leq \frac{N_M}{\alpha}
\end{aligned}$$

On obtient ainsi la borne inférieure suivante pour tout cycle :

$$T \geq \frac{m + N_M - X - 1}{N_M} p + \frac{2(m + 2N_M - X - 1)}{N_M} \delta$$

Or dans ce cas N_M est inférieur ou égal à $m-1$ et comme X est forcément inférieur ou égal à $N_M - 1$ on obtient la borne suivante :

$$\begin{aligned}
T &\geq \frac{m}{m-1} p + 2\left(\frac{m}{m-1} + 1\right)\delta \\
&\geq \frac{m}{m-1} p + \left(\frac{4m-2}{m-1}\right)\delta \\
T &\geq T(C_3(m-1))
\end{aligned}$$

Comme le temps de cycle de $C_3(m-1)$ est une borne inférieure, on peut donc en déduire que dans cet intervalle, le cycle $C_3(m-1)$ est dominant par rapport aux cycles de degré inférieur à $m-1$.

6.3.3 Cycles optimaux pour $p \geq 4(m-1)\delta$

Pour $p \geq 4(m-1)\delta$ le 1-cycle $C_5 = \prod_m^{i=0}(m-i)$ est réalisable et dominant [19], son temps de cycle est $p + 4\delta$ et représente une borne inférieure des temps de cycle.

6.4 Exemple pour $m = 5$

Pour illustrer les résultats précédents, considèrons l'exemple où $m = 5$. Les cycles dominants sont donnés dans le tableau 6.3 et illustrés par les figures 6.14 à 6.20. Les cellules grises sont les cas qu'il reste à démontrer. Lors de la section précédente nous avons démontré les preuves de la conjecture pour p dans les intervalles $[0; 4\delta[$, $[14\delta; 16\delta[$ et lorsque p est supérieur à 16δ.

Tableau 6.3 – Cycles optimaux pour $m = 5$

p	$[0, 4\delta[$	$[4\delta, 8\delta[$	$[8\delta, 10\delta[$
Cycle	$C_1(1)$	$C_1(2)$	$C_1(3)$
Temps de cycle	$5p + 12\delta$	$6p/2 + 8\delta$	$7p/3 + 20\delta/3$
k	1	2	3

p	$[10\delta, 12\delta[$	$[12\delta, 14\delta[$	$[14\delta, 16\delta[$	$\geq 16\delta$
Cycle	$C_3(3)$	$C_2(4)$	$C_3(4)$	C_d
Temps de cycle	$6p/3 + 18\delta/3$	$6p/4 + 20\delta/4$	$5p/4 + 18\delta/4$	$p + 4\delta$
k	3	4	4	1

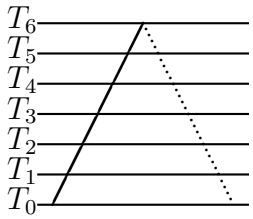

Figure 6.14 – $C_1(1)$ pour $m = 5$ et $p < 4\delta$

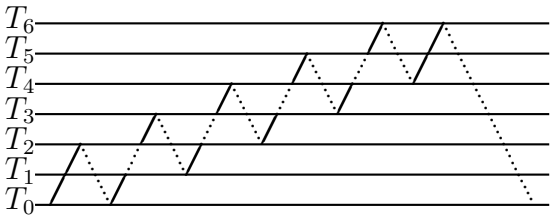

Figure 6.15 – $C_1(2)$ pour $m = 5$ et $p \in [4\delta; 8\delta[$

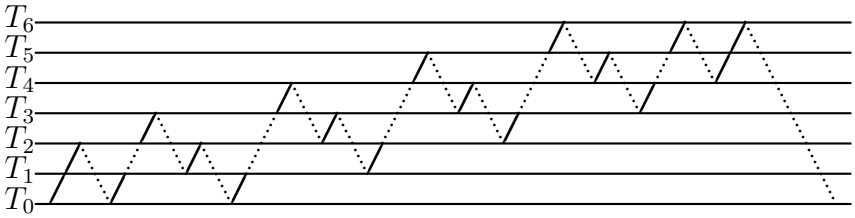

Figure 6.16 – $C_1(3)$ pour $m = 5$ et $p \in [8\delta, 10\delta[$

6.5 Conclusion

Dans ce chapitre, nous avons proposé deux conjectures sur les cycles optimaux dans le cas d'un nombre de cuve impair pour l'une et pair pour l'autre. Nous avons ensuite prouvé l'optimalité des cycles proposés dans la conjecture pour certaines configurations de la ligne ($p < (m + 2)\delta$ et $p \geq 4(m - 2)\delta + 2\delta$).

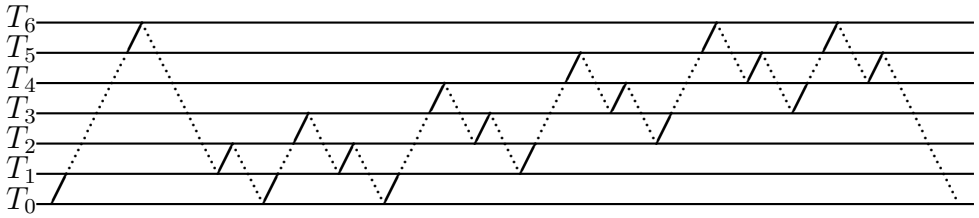

Figure 6.17 – $C_3(3)$ pour $m = 5$ et $p \in [10\delta; 12\delta[$

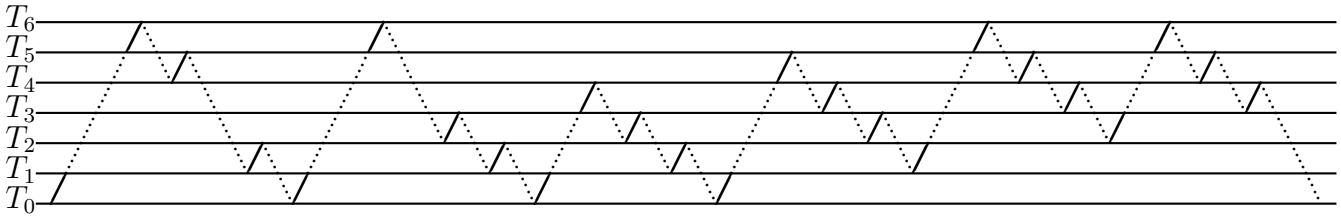

Figure 6.18 – $C_2(4)$ pour $m = 5$ et $p \in [12\delta; 14\delta[$

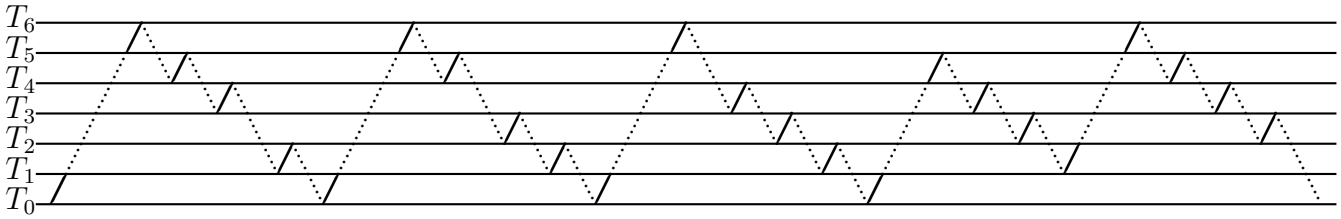

Figure 6.19 – $C_3(4)$ pour $m = 5$ et $p \in [14\delta; 16\delta[$

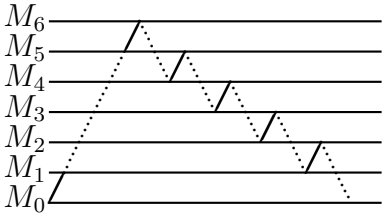

Figure 6.20 – C_5 pour $m = 5$ et $p \geq 16\delta$

Pour prouver les conjectures sur certains intervalles nous avons utilisé cette fois-ci des bornes obtenues avec les conditions sur les durées de trempe. Ces preuves confirment, un peu plus, la conjecture d'Agnetis (optimalité des $1, 2, \ldots, (m-1)$-cycles). En effet, nous pouvons en déduire que, pour tout entier k, il existe une ligne à $k+1$ cuves où un k-cycle est optimal. Nous pouvons surtout conclure que se limiter aux ℓ-cycles (avec ℓ fixe) n'est pas suffisant. Il faut au moins vérifier pour tout ℓ dans l'intervalle $[1, m-1]$.

Conclusions et perspectives

Conclusions

Ce travail de thèse se situe dans le cadre de l'ordonnancement des ateliers de production. L'ordonnancement consiste à trouver les séquences des produits à traiter et des ressources utilisées pour optimiser un critère. Il faut, de plus, que l'ordonnancement tienne compte des spécificités des lignes étudiées (contraintes de précédence, contraintes de ressources...). Les critères possibles peuvent engendrer des différences fondamentales dans l'ordonnancement. Les critères peuvent être la minimisation des encours ($\min(\sum C_i)$), la minimisation de la date de sortie du dernier produit ($\min C_{max}$), la minimisation des retards ou encore la minimisation du temps de cycle...

Cette étude traite des ateliers de traitement de surface. Sur ces lignes, les produits doivent être immergés successivement dans une série de cuves. Les durées de chaque opération ont la particularité de devoir être comprises dans un intervalle $[l_i, u_i]$ appelé fenêtre de temps. De plus, les produits sont transportés d'une cuve à l'autre par un robot. Il faut donc qu'au moment du calcul de l'ordonnancement, ces contraintes soient prises en compte explicitement.

Ce mémoire s'attache à une production par campagne, c'est à dire lorsque l'atelier ne produit qu'un seul type de pièces. Dans cette configuration, il n'est pas nécessaire de s'occuper de l'ordre de passage des produits sur la ligne. La solution du problème étant cyclique, le calcul de l'ordonnancement optimal (appelé *Hoist Scheduling Problem*) consiste alors à trouver les mouvements du robot qui minimisent la durée du cycle, ce qui revient à maximiser le taux de production.

Nous nous sommes intéressé aux caractéristiques des cycles optimaux. En effet, les méthodes et les heuristiques de la littérature permettent de calculer les k-cycles (cycles durant lesquels k produits entrent et k produits sortent de la ligne) optimaux pour des valeurs de k égales à 1 ou 2. Les méthodes qui permettent de calculer les cycles pour des valeurs de k supérieures nécessitent des temps de calcul très importants. Une conjecture, qui limite le degré k du cycle à $(m-1)$, a été proposée par Agnetis [1] pour le problème où les fenêtres de temps sont de largeurs nulles.

Notre objectif a été de montrer que cette conjecture peut s'appliquer au *Hoist Scheduling Problem* cyclique mono-produit.

Dans le chapitre 1, nous avons présenté les lignes de traitement de surface, leurs structures physiques et leurs caractéristiques. Nous avons montré ensuite comment ces caractéristiques interviennent pour le pilotage et les problèmes liés à l'ordonnancement. Les travaux existants sur le problème et sur les problèmes proches sont présentés dans un état de l'art.

Le chapitre 2 présente en détail le problème traité durant l'étude. Ce problème est le *Hoist Scheduling Problem* cyclique mono-produit avec comme critère d'optimisation la minimisation du temps de cycle. Les notations et les outils utilisés sont expliqués dans ce chapitre.

Nous avons étudié, dans le chapitre 3, le cas d'une ligne à deux cuves. Nous traitons, dans un premier temps, le problème mono-produit. Nous montré, à l'aide des *line-graphs* que les cycles optimaux sont des 1-cycles. Nous avons aussi donné dans le tableau 3.2 les cycles optimaux en fonction des paramètres du système. Nous avons proposé, dans un second temps, une méthode utilisant l'algorithme de Gilmore et Gomory permettant de trouver une solution réalisable au problème multi-produit. Nous n'avons pas complètement prouvé l'optimalité du cycle obtenu bien que, expérimentalement, nous n'ayons pas trouvé de contre exemple.

Le cas d'une ligne à trois cuves est étudié dans le chapitre 4. Toujours en utilisant les *line-graphs*, nous avons montré que les cycles optimaux sont des 1-cycles ou des 2-cycles, pour le *HSP* mono-produit équilibré avec marges nulles ou infinie. Ensuite, après avoir prouvé que les cycles optimaux sont des 1-cycles ou des 2-cycles pour des marges quelconques, nous avons proposé un tableau récapitulant les cycles optimaux en fonction des instances du problème.

Dans le chapitre 5, nous nous sommes attaché particulièrement au problème du *flowshop* robotisé sans attente, pour une ligne équilibrée à quatre cuves (durées opératoires fixes et égales). Dans ce chapitre nous avons montré que les cycles optimaux (cycles minimisant le temps de cycle) du problème mono-produit sont des 1-cycles, ou des 2-cycles, ou des 3-cycles. Ce qui confirme la conjecture proposée par Agnetis. Pour montrer ceci, nous avons utilisé une nouvelle fois les *line-graphs* en faisant une étude exhaustive des configurations.

Enfin dans le chapitre 6, nous avons généralisé au problème équilibré à m cuves. Nous avons proposé deux conjectures qui donnent les cycles optimaux en fonction des instances (nombre de cuves, durée de trempe et temps de transport) pour le problème du *flowshop* cyclique mono-produit, équilibré, sans attente. La première

conjecture est proposée pour le cas où le nombre de cuves est impair tandis que la seconde est proposée lorsque le nombre de cuves est pair. Nous avons prouvé, ensuite, ces conjectures pour certaines configurations. Nous avons prouvé surtout la dominance d'un $(m-1)$-cycle par rapport à tout k-cycle ($k \leq m-1$). Cette dernière remarque permet donc de montrer que si l'on recherche le cycle optimal, il est nécessaire de chercher, au moins, jusqu'à ($k = m-1$).

Perspectives

Les perspectives ouvertes par ces travaux sont multiples. Dans le chapitre 3, nous nous sommes attachés au cas multi-produit. Il faudrait dans la méthode proposée soit prouver l'optimalité de la solution obtenue soit trouver un contre-exemple et une autre méthode. Il reste aussi à prouver les conjectures sur les cycles optimaux pour le cas à m cuves pour toutes les configurations. Mais avant une preuve exacte, il serait possible de la prouver expérimentalement. En effet, la méthode, utilisée dans les chapitres 3 à 5, consistant à parcourir le *line-graph* et à comparer les cycles, peut, vraisemblablement, être informatisée. Même si il est raisonnable de croire que l'on ne pourra pas obtenir de calcul pour un nombre élevé de cuves, cela permettrait de valider les conjectures pour des valeurs de m supérieures à 4.

Une autre ouverture possible serait de généraliser au problème du *HSP*, c'est à dire avec fenêtres de temps de longueur non nulles, au moins pour le cas équilibré. Puis, pour le cas sans attente il serait bon de généraliser au cas non équilibré ou au moins de prouver que les cycles optimaux sont des 1- ou 2- ... ou $(m-1)$-cycles comme l'a proposé Agnetis.

Bibliographie

[1] Agnetis, A. Scheduling no-wait robotic cells with two and three machines. *European Journal of Operational Research 123*, 2 (2000), 303–314.

[2] Armstrong, R., Gu, S., and Lei, L. A greedy algorithm to determine the number of transporters in a cyclic electroplating process. *IIE Transactions 28* (1996), 347–355.

[3] Artigues, C., and Roubelat, F. An operation insertion procedure in a multi-ressource schedule based on a dominance rules. Tech. rep., LAAS, 01 1997.

[4] Baptiste, P., Bloch, C., and Varnier, C. *Ordonnancement de la production.* P. Lopez and F. Roubellat Eds. Hermès, 2001, ch. 9 : Ordonnancement des lignes de traitement de surface.

[5] Baptiste, P., Legeard, B., Manier, M.-A., and Varnier, C. Résolution d'un problème d'ordonnancement avec la PLC. *Journal Européen des Systèmes Automatisés 30*, 2-3 (1996), 201–30.

[6] Blażewicz, J., Ecker, K.-H., Pesch, E., Schmidt, G., and Węglarz, J. *Scheduling Computer an Manufacturing Processes.* Springer, Berlin, 1994.

[7] Bloch, C. *Contribution à l'ordonnancement dynamique de lignes de traitement de surface.* PhD thesis, Université de Franche-Comté, 1999.

[8] Brauner, N., and Finke, G. On a conjecture about robotic cells : new simplified proof for the three-machine case. *Journal of Information Systems and Operational Research - INFOR : Scheduling in Computer and Manufacturing Systems 37*, 1 (1999), 20–36.

[9] Brauner, N., and Finke, G. On cycles and permutations in robotic cells. *Mathematical and Computer Modelling 34* (2001), 565–591.

[10] Brauner, N., Finke, G., and Kubiak, W. A proof of the Lei and Wang claim. Tech. rep., Laboratoire Leibniz, Institut IMAG, 1997.

[11] Brauner, N., Finke, G., and Kubiak, W. Complexity of one-cycle robotic flowshops. Tech. rep., G.E.M.M.E, 2000.

[12] Brauner-Vettier, N. *Ordonnancement dans des cellules robotisées.* PhD thesis, Université Joseph Fourier de Grenoble, 1999. (in french).

[13] Chauvet, F., Levner, E., Meyzin, L., and Proth, J.-M. On-line scheduling in a surface treatment system. *European Journal of Operational Research 120*, 2 (2000), 382–392.

[14] Che, A., Chu, C., and Levner, E. A polynomial algorithm for 2-degree cyclic robotic scheduling. *European Journal of Operational Research 145* (2001), 31–44.

[15] Chen, H., Chu, C., and Proth, J.-M. Cyclic scheduling of a hoist with the time window constraint. Tech. Rep. 2307, INRIA, 1994.

[16] Chen, H., Chu, C., and Proth, J.-M. Cyclic hoist scheduling based on graph theory. In *Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation* (Los Alamitos, CA, USA, oct 1995), I. C. S. Press, Ed., vol. 1, Emerging Technologies and Factory Automation (INRIA/IEEE), pp. 451–459.

[17] Chen, H., Chu, C., and Proth, J.-M. Sequencing of parts in robotic cells. *International Journal of Flexible Manufacturing Systems 9* (1997).

[18] Collart-Dutilleul, S. *Commande robuste d'ateliers à contraintes de temps de séjour : application à la galvanoplastie.* PhD thesis, Ecole supérieure d'ingénieur d'Annecy, 1997.

[19] Crama, Y., Kats, V., van de Klundert, J., and Levner, E. Cyclic scheduling in robotic flowshops. *Annals of Operation Research : Mathematics of Industrial Systems 96* (2000), 97–124.

[20] Crama, Y., and van de Klundert, J. Cyclic scheduling of identical parts in a robotic cell. *Operations Research 45*, 6 (1997), 952–965.

[21] Crama, Y., and van de Klundert, J. Robotic flowshop scheduling is strongly NP-complete. Tech. rep., Research memoreandum, Maastricht, 1997.

[22] Crama, Y., and van de Klundert, J. Cyclic scheduling in 3-machine robotic flow shops. *Journal of Scheduling 2* (1999), 35–54.

[23] Fleury, G., Gourgand, M., and Lacomme, P. Coupling meta-heuristics and discrete event simulation models for the stochastic hoist scheduling problem. *ACS'99* (1999).

[24] Fleury, G., Gourgand, M., and Lacomme, P. Meta-heuristics for the stochastic hoist scheduling problem. *International Journal of Production Research 39*, 15 (2001), 3419–3457.

[25] Garey, M.-R., Johnson, D.-S., and Sethi, R. The comlexity of flow shop and job shop scheduling. *Mathematical and Operation research 1* (1976), 117–129.

[26] Ge, Y., and Yih, Y. Crane scheduling with time window in circuit board production lines. *International Journal of Production Research 33*, 5 (1995), 1187–1189.

[27] Gilmore, P., and Gomory, P. Sequencing a one state-variable machine : a solvable case of the travelling salesman problem. *Operations Research* (1964), 665–679.

[28] Goldberg, D. *Genetic Algorithm in Search.* Addison Wesley, 1989.

[29] Goujon, J.-Y., and Lacomme, P. Hoist scheduling problem : Modèle objets. Rapport interne, Laboratoire d'informatique (LIMOS), janvier 1997.

[30] Grunder, O., Baptiste, P., and Chappe, D. The relationship between the physical layout of the work stations and the productivity of a saturated single-hoist production line. *International Journal of Production Research 35*, 8 (1997), 2189–2211.

[31] Hall, N., Kamoun, H., and Sriskandarajah, C. Scheduling in robotic cells : Complexity and steady state analysis. Tech. rep., College of Business, The Ohio University, 1995.

[32] Hall, N., and Sriskandarajah, C. A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research 3*, 44 (1996), 510–525.

[33] Hanen, C., and Munier, A. Ordonnancement cyclique d'un robot sur une ligne de galvanoplastie : modèles et algorithmes. Tech. Rep. 93-30, LITP, May 1993.

[34] Holland, J.-H. *Adaptation in Natural and Artificial System.* The University of Michigan Press, 1975.

[35] Johnson, S. Discussion : Sequencing n jobs on two machines with arbitrary time lags. *Management Science 5* (1959), 293–298.

[36] Jullien, C. Ordonnancement d'une chaine de galvanoplastie. *DEA de Génie industriel (ENSGI-INPG)* (1999).

[37] Kamoun, H., Hall, N., and Sriskandarajah, C. Scheduling in robotic cells : Heuristics an cell design. *Operations Research 47*, 821-835 (1999).

[38] Kashyrskikh, K., Potts, C., and Sevastianov, S. A 3/2-approximation algorithm for two-machine flow-shop sequencing subject to release dates. *Discrete Applied Mathematics 14* (2001).

[39] Kats, V., and Levner, E. A strongly polynomial algortithm for no-wait cyclic robotic flow shop scheduling. *Operations Research Letters 21* (1997), 171–179.

[40] Kats, V., and Levner, E. Cyclic scheduling in a robotic production line. *Journal of Scheduling 5* (2002), 23–41.

[41] Khansa, W. *Réseaux de Petri p-temporels : contribution à l'étude des systèmes à événements discrets.* PhD thesis, Ecole supérieure d'ingénieur d'Annecy, 1997.

[42] Lamothe, J. *Une approche pour l'ordonnancement dynamique d'un atelier de traitement de surface.* PhD thesis, Ecole Supérieure de l'Aéronautique et de l'Espace Toulouse, 1996.

[43] Lamothe, J., and Delmas, J. Ordonnancement dynamique d'un atelier de traitement de surface avec cuves de capacité multiple. In *Deuxième Congrés International Franco-Québécois de Génie Industriel* (ALBI, 1997).

[44] Lei, L. Determining the optimal starting time in a cyclic schedule with a given route. *Computers and Operations Research 20*, 8 (1993), 807–816.

[45] Lei, L., and Wang, T.-J. A proof : the cyclic HSP is NP-complete. Tech. Rep. 89-0016, Graduate School of Management, Rutgers Univ 1989, 1989.

[46] Lei, L., and Wang, T.-J. The minimum common-cycle algorithm for cyclic scheduling of two material handling hoists with time window constraints. *Management Science 37*, 12 (1991), 1629–1639.

[47] Levner, E., and Kats, V.-B. A parametric critical path problem and an application for cyclic scheduling. *Discrete Applied Mathematics 87* (1998), 149–158.

[48] Lim, J.-M. A genetic algorithm for a hoist scheduling in the printed-circuit-board electroplating line. *Computers and Industrial Engineering 33*, 3-4 (1997), 789–792.

[49] Liu, J., Jiang, Y., and Zhou, Z. Cyclic scheduling of a single hoist in extended electroplating lines : a comprehensive integer programming solution. *IIE Transactions 34* (2002), 905–914.

[50] Lopez, P., and Roubelat, F. *Ordonnancement de la production.* Lavoisier, Hermes, 2000.

[51] Mak, R., Lam, K., and Gupta, S. A practical algorithm for cyclic hoist in a PCB manufacturing facility. *Journal of Electronic Manufacturing 8*, 3-4 (1998), 193–207.

[52] Mangione, F., Brauner, N., and Penz, B. Three tank hoist scheduling problem with unbounded or zero-width processing windows. In *8th International Workshop on Project Management and Scheduling - PMS* (Valencia, Spain, April 2002), EURO, Ed., pp. 253–256.

[53] Mangione, F., Brauner, N., and Penz, B. Balanced hoist scheduling problem with unbounded or zero-width processing windows. *Journal Européen des Systèmes Automatisés à paraître* (2003).

[54] Mangione, F., Brauner, N., and Penz, B. Flowshop robotisé à quatre machines sans attente. In *4ème Conférence Francophone de Modélisation et de Simulation (MOSIM)* (Toulouse, France, avril 2003), MOSIM, pp. 542–545.

[55] Mangione, F., Brauner, N., and Penz, B. Optimal cycles for the robotic balanced no-wait flow shop. In *International Conference of Industrial Engineering and Production Management - IEPM'03* (Porto, Portugal, May 2003), IEPM.

[56] Manier, M.-A., and Baptiste, P. Etat de l'art : Ordonnancement de robots de manutention en galvanoplastie. *APII 28*, 1 (1994), 7–35.

[57] Manier, M.-A., Varnier, C., and Baptiste, P. Constraint-based model for the cyclic multi-hoist scheduling problem. *PPC 11*, 3 (2000), 244–257.

[58] Phillips, L., and Hunger, P. Mathematical programming solution of a hoist scheduling problem. *AIIE Transactions* (1976), 219–225.

[59] Pinedo, M. *Scheduling Theory, Algorithms and Systems.* Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[60] Potts, C. Analysis of heuristics for two-machine flow-shop sequencing subject to release dates. *Mathematical and Operation Research 10* (1985), 576–584.

[61] Pujo, P., and Kieffer, J.-P. *Méthodes de pilotage des systèmes de production.* Lavoisier, Hermes, 2002.

[62] Reddi, S.-S., and Ramamoorthy, C.-V. On the flow-shop sequencing problem with no-wait in process. *Operationnal Research Quaterly 23* (1972), 323–331.

[63] Röck, H. The three-machine no-wait flow shop is NP-complete. *Journal of the Association for Computing Machinery 31*, 2 (1984), 336–345.

[64] Sethi, S.-P., Sriskandarajah, C., Sorger, G., Blazewicz, J., and Kubiak, W. Sequencing of parts and robots moves in a robotic cell. *International Journal of Flexible Manufacturing Systems 4* (1992), 331–358.

[65] Shapiro, G.-W. *Hoist scheduling for a PCB Electroplating Facility.* Program in operation research, Graduate Faculty of North Carolina State University, 1985.

[66] Shapiro, G.-W., and Nuttle, H. Hoist scheduling for a pcb electroplating facility. *IIE Transactions 20*, 2 (June 1988), 157–167.

[67] Song, W., Zabinsky, Z., and Storch, R. An algorithm for scheduling a chemical processing tank line. *Production Planning and Control 35* (1997), 277–284.

[68] Sriskandarajah, C., and Wagneur, E. Lot streaming and scheduling multiple products in two-machine no-wait flowshops. *IIE Transactions 35* (1999), 695–707.

[69] Subaï, C., Niel, E., and Baptiste, P. Vers un pilotage propre des lignes de traitement de surface. In *4ème Conférence Francophone de Modélisation et de Simulation (MOSIM)* (Toulouse, France, avril 2003), MOSIM.

[70] Varnier, C. *Extension du hoist scheduling problem cyclique. Résolution basée sur un traitement des contraintes disjonctives en programmation logique avec contraintes.* PhD thesis, Université de Franche-Comté, 1996.

[71] Varnier, C., Bachelu, A., and Baptiste, P. Resolution of the cyclic multi-hoists scheduling problem with overlapping partitions. *INFOR 35*, 4 (1997).

[72] Varnier, C., and Jeunehomme, N. A cyclic approach for the multi-product hoist scheduling problem. In *7th International Workshop on Project Management and Scheduling - PMS* (Osnabrueck, Germany, 2000), EURO, Ed.

[73] Yih, Y. Trace driven knowledge acquisition (tdka) for rule based real time scheduling systems. *Journal of Intelligent Manufacturing 1*, 4 (1988), 217–229.

[74] Yih, Y. An algorithm for hoist scheduling problems. *International Journal of Production Research 32*, 3 (1994), 501–516.

[75] Yin, N., and Yih, Y. Crane scheduling in a flexible electroplating line : a tolerance based approach. *Journal of Electronic Manufacturing 2* (1992), 137–144.

Résumé : Cette thèse traite des lignes de traitement de surface qui sont des lignes dans lesquelles les pièces sont immergées dans une succession de cuves. Chaque cuve contient des bains qui affectent les propriétés mécaniques ou électriques des pièces. Ce type de ligne est utilisé, par exemple, pour la galvanoplastie. Les pièces sont montées sur des porteurs et transportées d'une cuve à l'autre par un robot. Le temps opératoire (ou temps pendant lequel la pièce reste dans la cuve) est borné. La borne inférieure est le temps minimum qui permet le traitement et la borne supérieure dépend du type de traitement (attaque acide, rinçage...).
Un objectif classique est de trouver les mouvements du robot qui maximisent la productivité, ce problème est communément appelé "*hoist scheduling problem*" (*HSP*). Lors de ce travail nous nous sommes attachés à une production cyclique. Nous avons proposé dans le cas d'une ligne à deux cuves une méthode permettant d'obtenir les cycles optimaux. Nous avons démontré, pour le cas d'une ligne équilibrée à trois cuves pour une production mono-produit, les caractéristiques des cycles optimaux ainsi qu'une méthode pour les obtenir. Ensuite, nous avons étudié le problème sur quatre machines dans le cas où les temps de trempe sont égaux et sans attente. Nous avons proposé les cycles optimaux dans le cas d'une production mono-produit. Enfin nous avons proposé une conjecture sur les cycles optimaux et en avons démontré certaines parties, dans le cas d'une ligne équilibrée avec un nombre de cuves quelconque et où les marges sur les temps de process sont nulles.
Mots clés : Ateliers de traitement de surface, Hoist Scheduling Problem, ordonnancement, cycles, flowshop.

Abstract : In this thesis we study the automated electroplating lines. In these lines, the products are immerged in different tanks. Each tank contains baths, which affect the mechanical or electrical properties of the products. The parts are transferred from a tank to another one by a hoist. The processing times are bounded. The lower bound represents the minimum time to treat the product while the upper bound depends on the treatment.

A classical objective is to find the robot moves which maximize the throughput rate, this is called "hoist scheduling problem" (HSP). In this thesis, we study the cyclic case. We proposed for a two-tanks line a method to obtain the optimal cycles. We proved, for a balanced three-tanks line and a single-part production, the caracteristics of the optimal cycles and a way to find them. Then we proposed the optimal cycles for the no-wait case in a four-tanks line. Finally we gave, for a balanced m-tanks line and a single part production, a conjecture which gives the structure of the optimal production cycles and proved it for several cases.
Keywords : Printed circuit board, Hoist Scheduling Problem, scheduling, cycles, flowshop