Ordonnancement dans des cellules robotisées

Nadia Brauner

UNIVERSITÉ JOSEPH FOURIER-GRENOBLE I
SCIENCES & GÉOGRAPHIE

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER

Formation doctorale : Recherche Opérationnelle, Combinatoire et Optimisation

Discipline : Informatique

Présentée et soutenue publiquement

par

Nadia BRAUNER-VETTIER

le 24 septembre 1999

Ordonnancement dans des cellules robotisées

Directeur de Thèse : Gerd FINKE

Jury :

M. Michel BURLET,	Examinateur
M. Yves CRAMA,	Rapporteur
M. Gerd FINKE,	Directeur de thèse
M. Christian PROUST,	Rapporteur
Mme Catherine ROUCAIROL,	Présidente
M. András SEBÖ,	Examinateur

À Raja et Michel Brauner, mes parents,

À Benjamin Vettier, mon mari,

Avec tendresse,
Avec amour.

Remerciements

Je remercie Gerd Finke qui m'a accueillie dans son équipe, qui m'a guidée, qui a dirigé cette thèse et qui m'a donné les possibilités de faire une thèse dans des conditions excellentes. Je le remercie également pour sa confiance, sa disponibilité, ses encouragements, ses conseils, son soutien de tous les jours, l'équipe et l'environnement de travail qu'il a créés et surtout pour tout ce qu'il m'a appris et transmis sur le travail d'enseignant et de chercheur.

Je remercie Philippe Jorrand, directeur du laboratoire Leibniz-IMAG, au sein duquel ce travail a été effectué, pour son accueil.

Mes remerciements vont aux membres du jury :

à Catherine Roucairol qui m'a fait l'honneur d'accepter de présider le jury de cette thèse et qui m'a encouragée ;

à Yves Crama qui a accepté d'être rapporteur de cette thèse. Je le remercie pour ses encouragements, ses commentaires constructifs, ses remarques et ses suggestions qui ont permis d'améliorer ce document ;

à Christian Proust qui, en tant que coordonnateur du groupe de travail Bermudes, m'a permis de rencontrer la communauté française d'ordonnancement et qui a accepté d'être rapporteur de cette thèse. Je le remercie pour le temps qu'il m'a consacré, pour ses remarques et pour ses encouragements ;

à Michel Burlet pour sa présence, ses précieux conseils, son accueil et pour avoir accepté d'être dans mon jury ;

à András Sebő, pour l'intérêt qu'il a porté à ce travail, pour ses conseils et pour avoir accepté d'être examinateur de ma thèse.

Je remercie Wojciech Bienia et Denis Naddef qui m'ont fait découvrir et aimer la Recherche Opérationnelle et qui m'ont encouragée à devenir chercheur.

Je remercie tout particulièrement Marie-Laure Espinouse et Clarisse Dhaenens-Flipo qui m'ont guidée dans la vie d'enseignant chercheur et qui ont patiemment relu, corrigé et critiqué ce document. Je n'oublierai pas l'équipe que nous avons formée.

Je remercie les thésards du département de mathématiques discrètes, pour l'esprit de soutien et d'entraide qu'ils ont entretenu, en particulier Mouna Sadli qui a partagé mon bureau pendant ces trois années, Ammar Oulamara, Haris Gavranović, Yann Kieffer, Samia Ould-Ali et Sylvain Durand. Je remercie aussi mes collègues et amis du LMC, notamment Alfredo Goldman, Olivier Briant, Ekbel Bouzgarrou et Gregory Mounié.

Je remercie ceux avec qui j'ai eu la chance de travailler et qui m'ont guidée, en particulier Jacek Błażewicz, Lionel Dupont, Wieslaw Kubiak, Bernard Penz et Denis Trystram.

Je remercie enfin mes amis, ma famille de toujours et celle de mon mari, pour avoir cru en moi, m'avoir encouragée et soutenue.

Je remercie encore une fois tous ceux qui ont contribué a faire de ces trois années de thèse des années productives et heureuses.

Contents

List of Figures

List of Tables

Introduction

L'automatisation de la production a créé de nouveaux problèmes d'ordonnancement. Ainsi, dans les cellules de production robotisées, un robot est chargé du transport des pièces entre les machines. Ce robot est contrôlé par un ordinateur qui doit fournir rapidement des instructions sur les pièces à transférer. Nous étudions la production cyclique de pièces : une même séquence de mouvements est répétée pour la production d'un lot de pièces.

Un flow-shop (ou atelier en ligne) robotisé est une cellule de production dont les machines sont disposées en arc de cercle et dont le robot central bidirectionnel est chargé du transport des pièces entre les machines. Ce type de cellule de production a été introduit en 1985 par Asfahl [5]. Dans ce livre, l'auteur décrit une cellule d'usinage de pièces pour l'assemblage de différentiels de camions. Ce problème est présenté Figure 1. On remarque qu'il n'y a pas de zone de stockage dans la cellule et que l'entrée et la sortie des pièces sont dissociées. L'objectif est d'usiner des pièces en maximisant le taux de production.

Les problèmes d'ordonnancement dans des flow-shops robotisés sont NP-difficiles s'il y a $m \geq 3$ machines et différents types de pièces [39]. Il reste le cas d'une cellule robotisée à m machines dans laquelle on veut produire un seul type de pièces. Il s'agit alors de trouver une stratégie pour les mouvements du robot afin d'obtenir le taux maximal de production. Nous supposons que le processus de production est périodique : un certain nombre k de pièces est produit selon une séquence qui est ensuite répétée. Ce type de production cyclique facilite la programmation du robot. Une conjecture intéressante proposée par Sethi, Sriskandarajah, Sorger, Błażewicz et Kubiak [66], appelée Conjecture des 1-cycles, prétend que le taux maximum de production peut être atteint en répétant un cycle particulier qui produit à chaque fois une seule pièce.

L'objectif de ce travail est d'étudier la validité de cette conjecture pour différents types de cellules robotisées. Ce mémoire est divisé en trois parties. La première partie permet de définir le cadre pour les parties suivantes. Les parties II et III sont indépendantes. Les chapitres numérotés de 3 à 8 peuvent être lus indépendamment les uns des autres. Le contenu des chapitres est décrit au début de chaque partie. Les preuves les plus techniques sont présentées dans les annexes A et B.

Figure 1: Cellule robotisée à $m = 3$ machines (reproduit de [5])

La première partie a pour objectif de présenter le problème et ses propriétés, des outils de résolution et un module de test et de calcul (Chapitres 1 et 2). La deuxième partie concerne l'analyse de la production cyclique dans des flow-shops robotisés simples, sans espace de stockage avec des durées de transfert additives. La troisième partie concerne d'autres types de cellules robotisées. Nous étudions, pour chacune des variantes, la validité de la Conjecture des 1-cycles et le rôle des cycles de production d'une pièce. Dans certains cas, nous analysons également la complexité de la recherche du meilleur cycle de production d'une pièce.

Le problème du flow-shop robotisé a été proposé en 1985 et a commencé à être étudié il y a une dizaine d'années [66]. La Conjecture des 1-cycles a été posée en 1989 et publiée en 1992 [66]. Les auteurs de ce papier n'ont prouvé la conjecture que pour des cellules à deux machines. Ils pensaient que trouver le meilleur cycle de production d'une pièce était déjà un problème difficile (*¡¡ quite tedious although not impossible ¿¿*). Depuis Crama et van de Klundert ont prouvé que ce problème pouvait être résolu en temps polynomial [27]. En 1993, Hall et al. [38] ont démontré que la conjecture est vraie pour des cellules à trois machines lorsqu'on se restreint aux cycles de production de une ou de deux pièces. Puis, en 1996, Crama et van de Klundert ont donné une preuve assez longue de la validité de la conjecture pour des cellules à trois machines [74, 29].

Nous proposons (Chapitre 3) de nouvelles preuves de la conjecture, pour des cellules à deux et trois machines, avec des approches plus simples et plus générales qui unifient également d'autres résultats déjà connus. En 1997, nous avons prouvé que la Conjecture des 1-cycles était fausse à partir de quatre machines [14].

La conjecture étant fausse pour le cas général, nous considérons des restrictions sur les paramètres de la cellule (Chapitre 4). Ainsi, lorsque les machines sont équidistantes et les temps d'usinage sont identiques sur toutes les machines, la conjecture est vraie pour des cellules à quatre machines. De plus, le meilleur cycle de production d'une pièce peut être trouvé en temps constant quel que soit le nombre de machines.

Nous étudions également des variantes de la cellule robotisée de base (Partie II). Nous proposons d'abord quelques remarques lorsque la cellule est circulaire (Chapitre 5) : l'entrée et la sortie de la cellule sont au même endroit. Cette configuration a été très largement étudiée pour le problème proche du *Hoist Scheduling Problem* ou HSP (voir [11]) mais très peu pour les cellules robotisées.

Nous proposons également quelques remarques sur le HSP (Chapitre 6). Pour ce problème, les pièces doivent rester un temps limité sur les machines, pour un traitement chimique par exemple. Ce problème est beaucoup plus connu et étudié que le problème des cellules robotisées. Nous montrons que des propriétés intéressantes des cellules robotisées ne peuvent pas être étendues au HSP.

L'absence d'espace de stockage limite beaucoup les mouvements du robot. Nous levons cette contrainte (Chapitre 7) et nous prouvons que, si l'on rajoute un espace de stockage unitaire derrière chaque machine, la conjecture est vraie quel que soit le nombre de machines. Nous étudions également le gain entraîné par l'ajout d'espaces de stockage unitaires derrière les machines.

Enfin, nous supposons que les dures de transfert sont quelconques (Chapitre 8). Nous prouvons que, dans ce cas, la conjecture est fausse pour des cellules à trois machines et plus et que trouver le meilleur cycle de production d'une seule pièce est un problème NP-difficile.

Nous concluons en résumant les résultats présentés dans ce mémoire et en proposant des perspectives à ce travail.

Part I

Flow-shop robotisé

Introduction de la première partie

Dans cette partie, nous présentons les cellules robotisées et nous décrivons les notations, définitions et propriétés utilisées tout au long de ce travail.

Nous commençons par situer notre problème parmi les problèmes d'ordonnancement en décrivant le chemin qui, de l'ordonnancement en général, mène à l'ordonnancement des mouvements du robot dans un flow-shop mono-produit (Chapitre 1).

Dans le chapitre suivant nous introduisons les cycles de production et leurs propriétés (Chapitre 2). Nous décrivons d'abord une représentation générique des cycles. Puis, nous discutons la stabilité des cycles de production ce qui nous permet de définir le temps de cycle en fonction des paramètres de la cellule. Nous présentons ensuite les graphes d'état qui permettent de trouver des propriété intéressantes des cycles lorsque le nombre de machines est petit. Nous concluons ce chapitre avec la description d'un module de test et de calcul qui accompagne ce travail.

Chapter 1

De l'ordonnancement aux flow-shops robotisés

Ce chapitre décrit le chemin qui, à travers les problèmes d'ordonnancement, mène aux cellules robotisées. Pour chaque étape, nous donnons quelques références bibliographiques. Notre propos n'est pas de décrire toute la littérature sur les sujets mentionnés : plus le sujet est vaste et moins nous citons de références le concernant. À l'inverse, en approchant du problèmes précis qui nous concerne, l'état de l'art devient plus détaillé.

Nous commençons par définir les problèmes d'ordonnancement. Puis, nous nous limitons aux problèmes de flow-shop. L'étape suivante est le flow-shop avec gestion des ressources qui se spécialise en flow-shop avec gestion des ressources de transport.

1.1 Brève introduction à l'ordonnancement

Un problème d'ordonnancement est généralement décrit à partir de quatre éléments : les tâches, les processeurs, les contraintes et le critère de performance.

Les *tâches* sont des ensembles de travaux à exécuter. Les paramètres d'une tâche sont le mode d'exécution (préemptif ou non...), la durée d'exécution (p_j), la date de disponibilité (r_j), la date au plus tard (d_j)...

Le terme *processeur* désigne en fait ce qui traite et transforme les tâches (*to process*, en anglais).

Les *contraintes* peuvent être des contraintes de précédence entre les tâches, des contraintes sur la disponibilité des ressources (outil, transport)...

Le *critère de performance* peut être la durée totale de l'ordonnancement (C_{max}),

la durée d'un cycle de production (C_t), le retard maximum (T_{max})...

Ordonnancer signifie affecter des processeurs aux tâches et définir des dates d'exécutions des tâches en respectant les contraintes afin d'optimiser le critère de performance.

Les grandes classes de problèmes d'ordonnancement sont les suivantes :
- processeurs parallèles,
- processeurs dédiés
  - *open-shop* : gamme non fixée,
  - *job-shop* : gamme fixée,
  - *flow-shop* : gamme identique pour toutes les tâches,

où la *gamme* d'une tâche est l'ordre de passage de cette tâche sur les processeurs.

Nous citons quelques livres sur l'ordonnancement. Błażewicz et al. [8] proposent une étude et un schéma de classification (hérité de Graham et al. [36]) des problèmes d'ordonnancement dans les systèmes manufacturiers. Garey et Johnson [33] décrivent la complexité des problèmes combinatoires dont certains sont des problèmes d'ordonnancement. Tanaev et al. énumèrent de nombreux résultats d'ordonnancement pour des systèmes à un étage (processeurs parallèles) [73] et pour des systèmes à plusieurs étages (processeurs dédiés) [72]. Pinedo [65] propose un livre assez didactique sur l'ordonnancement. Le livre de Chrétienne et al. [22] rassemble un certain nombre d'articles sur l'ordonnancement. Enfin, la page web de Brucker et Knust [19] permet d'avoir un aperçu relativement à jour de la complexité de nombreux problèmes d'ordonnancement.

1.2 Flow-shop

Le problème du flow-shop se rencontre en particulier dans les ateliers en ligne : toutes les tâches doivent passer sur les machines dans le même ordre. Considérons le problème du flow-shop non préemptif avec, comme critère, la date d'achèvement (C_{max}) ou la durée d'un cycle de production. Ces deux critères sont assez proches. Par exemple, pour un flow-shop à deux machines sans attente, l'algorithme de Gilmore et Gomory [35] minimise le temps de cycle. Puis en rajoutant une certaine tâche fictive, cet algorithme minimise également le C_{max}. McCormick et Rao [60] étudient la relation entre ces deux critères.

Pour le flow-shop à deux machines, trouver l'ordre des pièces qui minimise le C_{max} est un problème polynomial si le buffer intermédiaire est de capacité illimitée [47] ou nulle [35, 41]. Il devient NP-complet si la capacité du buffer entre les deux machines est finie ([63] si le buffer respecte la règle "FIFO", [70] sinon). À partir de trois machines le problème est NP-complet quelle que soit la capacité des buffers (voir [34] pour des capacités illimitées et [41] pour des capacités nulles).

Citons quelques extensions du flow-shop. Hall et Sriskandarajah [41] présentent un

état de l'art sur le flow-shop sans attente et le flow-shop sans encours. Vignier et al. [75] exposent un état de l'art sur le problème du flow-shop hybride (plusieurs machines en parallèle à chaque étage, au lieu d'une seule machine). Espinouse [30] étudie quelques extensions du flow-shop comme le chevauchement des tâches (lots, préparation, remise en état...) ou l'indisponibilité des machines (maintenance...).

1.3 Flow-shop avec ressources

La production des pièces dans les ateliers en ligne requiert souvent des ressources additionnelles (outils, opérateurs, espaces de stockage...). Lorsque ces ressources sont en quantités limitées, la résolution du problème d'ordonnancement doit tenir compte de leur allocation.

Dans un ordonnancement admissible, les bonnes ressources doivent être à la bonne place, au bon moment. Les ressources peuvent être discrètes (outils) ou continues (énergie). Elles peuvent également être renouvelables (outils, opérateurs) ou non (matière brute).

Crama [24] et Błażewicz et Finke [9] proposent des états de l'art sur la gestion des ressources dans les systèmes manufacturiers. Ils distinguent les ressources de traitement (*processing resources*) et les ressources d'entrée/sortie (*I/O ressources*). Les ressources de traitement [7] sont bloquées pendant tout l'usinage de la pièce sur la machine. Les ressources d'entrée/sortie sont requises avant (ou au début de) ou après (ou à la fin de) l'usinage des pièces. Ces ressources sont, par exemple, des ressources de transport (véhicules filoguidés, robots), des espaces de stockage, des serveurs...

Concentrons nous sur les cellules avec ressources de transport [25]. Considérons d'abord le problème du *design* de la cellule de production. Certains auteurs se sont intéressés à la configuration (*layout*) de la cellule afin de minimiser les déplacements des ressources de transport [32, 48, 37, 61]. Chu et al. [23] et Błażewicz et al. [10] ont étudié le nombre minimum de ressources de transport nécessaires pour réaliser un plan de production. Divers problèmes d'ordonnancement dans des cellules robotisées ont été abordés : ordonnancement par batch [3], ordonnancement en environnement flou [58], cas multi-robot [46, 54, 10, 6, 57].

1.4 Ordonnancement avec système de transport

Nous considérons maintenant les problèmes d'ordonnancement dans des ateliers en ligne munis d'un unique robot chargé du transfert des pièces entre les machines. Le robot est une ressource bloquante. Le problème est donc à la fois d'ordonnancer

les mouvements du robot et de trouver l'ordre des pièces.

Certains auteurs considèrent des cellules robotisées où les machines sont équipées d'espaces de stockage (ou buffers). Kise [51] étudie la minimisation du C_{max} dans une cellule à deux machines avec des buffers infinis et un robot chargé du transfert des pièces. Il montre que ce problème est équivalent à un problème de flow-shop à trois machines et qu'il est NP-complet. Ce résultat a été repris et étendu dans [45]. Pour des buffers infinis et un nombre de machines arbitraire, King et al. [50] proposent un algorithme de séparation et évaluation.

Dans la suite de ce chapitre, nous considérons les flow-shops robotisés sans espace de stockage. Les problèmes d'ordonnancement dans les cellules robotisées sans stockage peuvent être décomposés en trois classes qui dépendent du temps pendant lequel une pièce peut rester sur une machine. Soit L_h^i le temps minimum pendant lequel la pièce i doit rester sur la machine M_h (temps d'usinage) et U_h^i le temps maximum pendant lequel la pièce i peut rester sur la machine M_h. Si $L_h^i = U_h^i$ pour tout i et pour tout h, alors il s'agit d'un problème sans attente (*no-wait*) : dès qu'elle est prête, la pièce doit être prise par le robot pour être transférée à la machine suivante. Si $L_h^i \leq U_h^i$ et si U_h^i n'est pas égal à $+\infty$ pour tout i et pour tout h, alors le problème est un HSP (*Hoist Scheduling Problem*) : la pièce doit rester sur la machine suffisamment longtemps pour pouvoir être traitée mais pas trop longtemps pour ne pas être détériorée. Si $U_h^i = +\infty$ pour tout i et pour tout h, alors il s'agit d'un problème d'ordonnancement dans une cellule robotisée : la pièce doit être usinée puis peut rester sur la machine en attendant que le robot et que la machine suivante soient libres.

Une étude des similitudes et des divergences entre ces problèmes et d'autres problèmes d'ordonnancement est en cours dans le groupe de travail Bermudes.[1]

Problème sans attente

Considérons d'abord le problème sans attente. L'instant où une pièce entre dans la cellule détermine de manière unique l'instant où elle en sort. Agnetis [1] a prouvé que, si les pièces sont différentes, alors minimiser le C_{max} dans une cellule à deux machines peut être résolu en $O(n \log n)$, où n est le nombre de pièces. Agnetis et Pacciarelli [2] ont étudié la complexité de ce problème dans une cellule à trois machines lorsque les mouvements du robot sont donnés. Le problème de maximiser le taux de production lorsque toutes les pièces sont identiques a été étudié par Hanen et Munier [42], Song et al.[69], Agnetis [1]... Ce problème est en général polynomial.

[1] Groupe de travail Bermudes, HSP, FMSSP, HFSSP : similitudes, divergences, typologies, notations (*http://bermudes.univ-bpclermont.fr/*).

HSP

Le HSP a de très nombreuses applications dans les industries qui utilisent des traitements chimiques (galvanoplastie [59], industrie pharmaceutique [44]). Il existe donc une très vaste littérature sur ce problème. Bloch et Manier [11, 12] proposent une typologie et un état de l'art sur le HSP.

Considérons le HSP mono-produit, mono-robot et la recherche du plus petit cycle de production d'une pièce. Crama et van de Klundert [27] ont prouvé que, pour des durées de transfert additives, ce problème est NP-complet. Phillips et Unger [64], Shapiro et Nuttle [67], Lei et Wang [55], Chen et al. [20], Ng [62], Hanen et Munier [42] et Song et al. [68], entre autres, ont proposé des méthodes de résolution du HSP mono-produit, mono-robot.

Cellules robotisées

Alors que le HSP s'intéresse aux traitements chimiques, le problème des cellules robotisées concerne l'usinage de pièces mécaniques. Ces deux problèmes sont très proches dans leur définition mais assez différents dans leur résolution.

Si l'objectif est C_{max}, Kise et al. ont prouvé que, pour des cellules à deux machines sans buffer, trouver l'ordre des pièces est un problème polynomial [52, 53].

Considérons les problèmes d'ordonnancement cycliques dans les cellules robotisées : une même séquence de mouvements du robot est répétée pour la production des pièces. L'objectif est de maximiser le taux de production des pièces. Le problème a été introduit dans [66]. Crama et al. [25] proposent un état de l'art sur le flow-shop robotisé cyclique.

Considérons le cas où l'on veut usiner cycliquement un ensemble de pièces différentes et où le nombre de types de pièces à produire est fini. Sethi et al. [66] décrivent un algorithme efficace pour une cellule à deux machines. Hall et al. [40] précisent ce résultat. Ils étudient également la complexité du problème en fonction du cycle de production considéré, dans une cellule à trois machines. Puis, ils prouvent que, dans une cellule à trois machines, trouver le meilleur cycle de production d'une pièce pour produire des pièces différentes est un problème NP-complet. Kamoun et al. [48] proposent des heuristiques pour résoudre ce problème rapidement dans des cellule à deux et trois machines. Hall et al. [39] décrivent un algorithme pseudo-polynomial pour un nombre quelconque de machines. Chen et al. [21] présentent un algorithme de séparation et évaluation lorsque la séquence de mouvements du robot est donnée. D'autres méthodes de résolution ont été abordées.

Considérons le cas mono-produit avec des durées de transfert additives. Sethi et al. [66] ont proposé, en 1992, une conjecture sur la dominance des cycles de production

d'une pièce. Ils ont prouvé que cette conjecture était vraie pour des cellules à deux machines. En 1993, Hall et al. [38, 40] ont prouvé que, pour des cellules à trois machines, les cycles de production de deux pièces ne dominent pas les cycles de production d'une pièce. Leur preuve est, en fait, une démonstration informatique : le problème est formulé comme un programme linéaire avec 103 contraintes, 81 variables continues et 30 variables binaires. Si la valeur minimale de la fonction objectif est strictement positive alors la Conjecture des 1-cycles est fausse pour des cellules à trois machines. Ce programme a été résolu avec un logiciel commercial et la valeur 0 a été obtenue pour la fonction objectif. En 1996, Crama et van de Klundert [29, 74] ont montré que cette conjecture était vraie pour des cellules à trois machines. Leur preuve est basée sur un graphes à 16 sommets et une étude de cas.

Chapter 2

Activités et cycles de production

Ce chapitre présente les outils et les propriétés nécessaires dans le reste du mémoire. La première section concerne la représentation des cycles de production et l'analyse des propriétés des mouvements du robot lors de l'exécution de ces cycles (Section 2.1). La section suivante décrit les paramètres de la cellule. Nous discutons la stabilité des cycles en fonction de ces paramètres et de la phase d'initialisation (Section 2.2). Ceci nous permet de définir le temps de cycle et de présenter la Conjecture des 1-cycles qui est l'un des éléments centraux de ce travail. Puis, nous présentons et analysons les graphes d'état qui permettent de décrire tous les mouvements possibles du robot dans la cellule (Section 2.3). Nous concluons ce chapitre en présentant un module de test et de calcul qui permet d'observer le déroulement des cycles, de compter le nombre de cycles et d'étudier certaines composantes du temps de cycle (Section 2.4).

2.1 Représentation générique des cycles

On considère une cellule de production composée de m machines et d'un robot chargé du transfert des pièces entre les machines. Les machines sont notées $M_1, M_2 \dots M_m$. Nous ajoutons deux machines auxiliaires, M_0 qui correspond au lieu de chargement IN et M_{m+1} qui correspond au lieu de déchargement OUT. La cellule robotisée représente un flow-shop avec un robot central chargé du transfert des pièces entre les machines. La matière brute nécessaire est disponible en quantité illimitée en M_0. Le robot central ne peut transporter qu'une seule pièce à la fois. Une pièce est prise en M_0 et transférée successivement, et dans cet ordre, sur $M_1, M_2 \dots M_m$, pour être usinée, jusqu'à ce qu'elle atteigne finalement le lieu de sortie M_{m+1}. En M_{m+1}, les pièces finies peuvent être stockées en quantité illimitée. Nous nous concentrons sur le cas classique, comme dans [66], où les machines $M_1, M_2 \dots M_m$ sont sans espace de stockage et de capacités unitaires. Dans ce cas, le robot doit être

vide pour prendre une pièce de $M_h (h = 0, 1 \ldots m)$. Pour se déplacer d'une machine à l'autre, le robot prend le plus court chemin sur l'arc de cercle formé par les machines. Par conséquent, les durées de transfert sont additives [Figure 2.1].

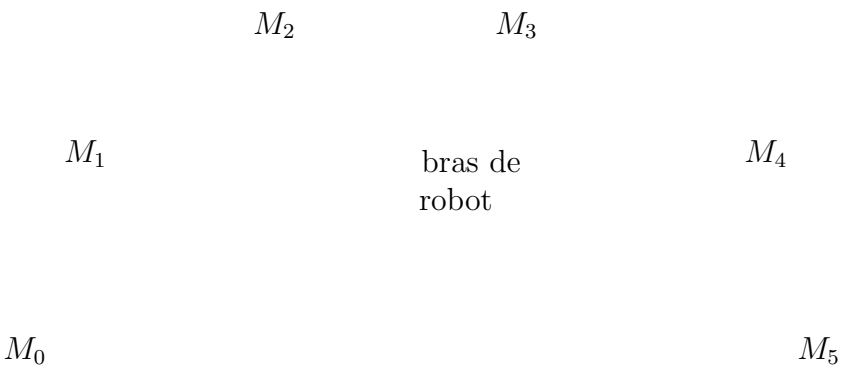

Figure 2.1: Cellule robotisée à quatre machines

Pour la production des pièces, nous considérons les mouvements cycliques du robot. Nous définissons un *k-cycle* comme un cycle de production de k pièces. Un k-cycle peut être décrit comme une séquence de mouvements du robot où exactement k pièces entrent dans le système en M_0, k pièces quittent le système en M_{m+1} et, après chaque exécution du k-cycle, l'état du système (incidence pièces/machines) et la position du robot sont restaurés. Ainsi, un k-cycle peut être répété indéfiniment. On peut remarquer que, à chaque exécution du k-cycle, seul le vecteur d'incidence pièces/machines est restauré. L'avancement de l'usinage des pièces sur les machines peut varier d'une exécution à l'autre.

Pour décrire un k-cycle, nous utilisons le concept d'activité. L'*activité* $A_h (h = 0, 1 \ldots m)$ est constituée de la séquence suivante :

- Le robot vide prend une pièce de M_h.
- Le robot transporte cette pièce de M_h à M_{h+1}.
- Le robot décharge cette pièce sur M_{h+1}.

Remarquons que de nombreuses séquences d'activités ne sont pas exécutables. Par exemple, $(\ldots A_0 A_0 \ldots)$ n'est pas admissible car le robot apporte une pièce sur M_1 qui est déjà occupée.

Les k-cycles ont été représentés de différentes manières dans la littérature avant l'utilisation des activités. Par exemple considérons le cycle $(A_3 A_2 A_0 A_1)$ dans une cellule à trois machines. Sethi et al. [66] représentent ce cycle à l'aide d'une succession d'états de la cellule :

$$(\emptyset, \Omega, \Omega, M_3^+)(\emptyset, \Omega, \emptyset, O)(\emptyset, \Omega, \emptyset, M_2^+)(\emptyset, \emptyset, \Omega, M_3^-)(\emptyset, \emptyset, \Omega, I)$$
$$(\Omega, \emptyset, \Omega, M_1^-)(\Omega, \emptyset, \Omega, M_1^+)(\emptyset, \Omega, \Omega, M_2^-)(\emptyset, \Omega, \Omega, M_3^+)$$

où $\emptyset$ signifie que la machine est vide, Ω signifie que la machine est occupée et M_h^j signifie que le robot est à la machine M_h avec $j = +$ si le robot va décharger la machine et $j = -$ si le robot vient de charger la machine. Hall et al. [40] utilisent les notations suivantes plus proches de celles que nous utilisons :

$$\{M_3^+, M_3^-, M_1^-, M_2^-, M_3^+\}$$

où M_i^- signifie que le robot charge une pièce sur M_i et M_m^+ signifie que le robot décharge une pièce de M_m. Le motif M_3^+ est répété pour indiquer que le cycle recommence.

Dans [29], Crama et van de Klundert caractérisent les k-cycles comme suit.

Définition 1 *Un k-cycle peut être décrit comme une séquence d'activités telle que chaque activité se produit exactement k fois et, entre deux occurrences consécutives (dans le sens cyclique) de A_h, il y a exactement une occurrence de A_{h-1} pour $(h = 1, 2 \dots m)$ et exactement une occurrence de A_{h+1} pour $(h = 0, 1 \dots m-1)$.*

Nous représentons un k-cycle comme sur la Figure 2.2. L'axe vertical représente les machines de la cellule. Le graphe indique la position du robot dans la cellule pendant l'exécution du cycle. Les lignes pointillées sont les mouvements à vide du robot et les lignes pleines sont les mouvements du robot lorsqu'il transporte une pièce.

Notons $m_h(C_k)$ $(h = 0, 1 \dots m)$ le nombre de fois où le robot se déplace entre M_h et M_{h+1}, dans les deux directions, pendant l'exécution du k-cycle C_k et $u_h(C_k)(h = 1, 2 \dots m)$ représente le nombre d'occurrences de la séquence d'activités $A_{h-1}A_h$ dans C_k. Pour le 1-cycle π de la Figure 2.2 nous avons, par exemple, $m_2(\pi) = 4$ et $u_7(\pi) = 1$ et $u_h(\pi) = 0$ pour $h = 1, 2 \dots 6$.

Proposition 1 *Les équations suivantes sont valides pour tout k-cycle C_k :*

$$m_0(C_k) = 2k \tag{2.1}$$
$$m_m(C_k) = 2k \tag{2.2}$$

Démonstration Pendant l'exécution du k-cycle C_k, k pièces entrent dans la cellule en M_0. Mais le robot ne fait aucun mouvement inutile. Par conséquent, il va k fois en M_0 afin de prendre une pièce et il quitte k fois M_0 chargé d'une pièce qui entre dans le système (exécution de A_0). Ainsi, le robot parcourt exactement $2k$ fois le chemin entre M_0 et M_1 : k fois vide de M_1 à M_0 et k fois chargé de M_0 à M_1. La preuve pour $m_m(C_k)$ est similaire. □

Proposition 2 *Les équations suivantes sont valides pour tout k-cycle C_k :*

$$m_1(C_k) = 4k - 2u_1(C_k) \tag{2.3}$$
$$m_{m-1}(C_k) = 4k - 2u_m(C_k) \tag{2.4}$$

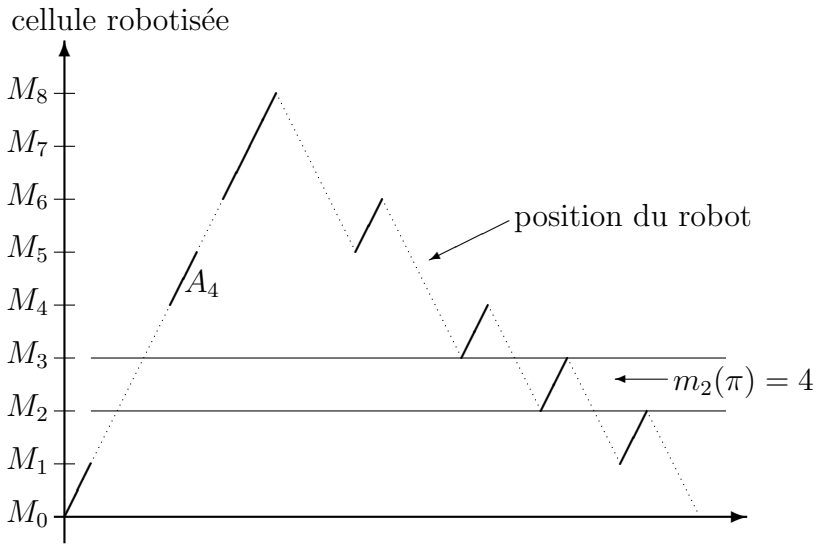

Figure 2.2: La permutation pyramidale $\pi = (A_0A_4A_6A_7A_5A_3A_2A_1)$

Démonstration Soit C_k un k-cycle. Pendant l'exécution de C_k, le robot parcourt le chemin de M_2 à M_1 dans le seul but d'exécuter A_1 ou de continuer vers M_0 pour exécuter A_0. Juste avant chacune des k exécutions de A_0 le robot vient nécessairement de M_2. De plus, pour exécuter A_1, le robot vient de M_0 s'il vient d'exécuter A_0 ($u_1(C_k)$ fois) et de M_2 sinon ($k - u_1(C_k)$ fois). Par conséquent, juste avant $(k-u_1(C_k))$ exécutions de A_1, le robot se déplace de M_2 à M_1. Ainsi, le robot se déplace de M_2 à M_1 exactement $(2k - u_1(C_k))$ fois (k fois pour exécuter A_0 et $(k - u_1(C_k))$ fois pour exécuter A_1). Mais, à chaque exécution du cycle, la position du robot est restaurée. Donc, le robot se déplace aussi exactement $(2k - u_1(C_k))$ fois de M_1 à M_2. Ceci implique que $m_1(C_k) = 4k - 2u_1(C_k)$.

La preuve pour $m_{m-1}(C_k)$ est similaire. En effet, le robot se déplace de M_m à M_{m-1}, k fois après l'exécution de A_m et $(k - u_m(C_k))$ fois immédiatement après l'exécution de A_{m-1}. □

Soit $|S|_{C_k}$ le nombre d'occurrences de la séquence d'activités S dans le k-cycle C_k. Nous supprimons C_k dans $|S|_{C_k}$ lorsqu'il n'y a pas de confusion possible.

Proposition 3 *Pour $m \geq 4$ et pour tout k-cycle C_k, on a*

$$\begin{aligned} m_2(C_k) &\geq 4k - 2|A_1A_0A_2| - 2u_2(C_k) \\ m_{m-2}(C_k) &\geq 4k - 2|A_{m-2}A_mA_{m-1}| - 2u_{m-1}(C_k). \end{aligned}$$

Démonstration Soit C_k un k-cycle. Normalisons C_k en terminant avec l'une des activités A_2. Notons S_i la sous-séquence de C_k qui commence à la i-ème occurrence de A_2 (non comptée dans la séquence) et finit à la $(i+1)$-ème occurrence de A_2 (comprise dans S_i).

Soit $|A_1A_0A_2|_i$ (respectivement $|A_1A_2|_i$) le nombre d'occurrences de $A_1A_0A_2$ (A_1A_2 respectivement) dans S_i. Notons $m_2(S_i)$ le nombre de fois où le robot effectue le trajet entre M_2 et M_3 (dans les deux directions) pendant l'exécution de S_i.

La Définition 1 indique que, entre deux occurrences consécutives de A_2, il y a exactement une occurrence de A_1. Donc, dans S_i, il y a exactement une occurrence de A_1. Au début de l'exécution de S_i, le robot est en M_3. Par conséquent, le robot parcourt au moins une fois le trajet entre M_3 et M_2, avant d'exécuter A_1.

Considérons maintenant la séquence d'activités entre A_1 et A_2 dans S_i. Si S_i contient la séquence $A_1A_0A_2$ ou la séquence A_1A_2, alors le trajet entre M_2 et M_3 est parcouru seulement une fois (exécution de A_2) après A_1. Pour toutes les autres sous-séquences A_1--A_2, le trajet entre M_2 et M_3 est parcouru au moins trois fois. Par conséquent, $m_2(S_i) \geq 1 + 3 - 2|A_1A_0A_2|_i - 2|A_1A_2|_i$.

Donc, comme le cycle C_k contient k occurrences de A_2, on a

$$\begin{aligned} m_2(C_k) &= \sum_{i=0}^{k-1} m_2(S_i) \\ &\geq \sum_{i=0}^{k-1} 4 - \sum_{i=1}^{k} 2|A_1A_0A_2|_i - \sum_{i=1}^{k} 2|A_1A_2|_i \\ &\geq 4k - 2|A_1A_0A_2| - 2|A_1A_2|. \end{aligned}$$

Ceci conclut la preuve pour la première inégalité. La preuve pour la seconde inégalité est similaire. □

Un 1-cycle est complètement défini par une permutation des activités [66]. Par conséquent, il existe exactement $m!$ 1-cycles. Nous normalisons, sans restriction, les séquences π d'activités en commençant avec l'activité A_0. Les 1-cycles sont donc de la forme $\pi = (A_0A_{i_1}A_{i_2}\dots A_{i_m})$ où $(i_1, i_2 \dots i_m)$ est une permutation de $\{1, 2 \dots m\}$. Considérons les 1-cycles π qui appartiennent à l'ensemble des permutations pyramidales. La permutation $\pi = (A_0A_{i_1}A_{i_2}\dots A_{i_m})$ est *pyramidale* s'il existe un indice p tel que $1 \leq i_1 < \dots < i_p = m$ et $m > i_{p+1} > \dots > i_m \geq 1$. La Figure 2.2 donne un exemple de permutation pyramidale.

Soit $\pi = (A_0A_{i_1}A_{i_2}\dots A_{i_m})$ une permutation pyramidale avec $i_p = m$. L'activité A_{i_j} est *montante* si $j \leq p$ et *descendante* sinon. On peut remarquer que, pendant l'exécution de π, le robot voyage deux fois entre M_h et M_{h+1} si A_h est montante ($m_h(\pi) = 2$) et quatre fois si A_h est descendante ($m_h(\pi) = 4$). Par définition, les activités A_0 et A_m sont toujours montantes et toutes les autres activités peuvent être soit montantes soit descendantes. Une partition des activités en ensembles d'activités montantes et d'activités descendantes définit, de manière unique, une permutation pyramidale. Il existe donc une bijection entre l'ensemble des permutations pyramidales et l'ensemble des vecteurs de taille $(m-1)$ dont les coefficients sont 2 ou 4. Il y a donc 2^{m-1} permutations pyramidales différentes. Posons $\mu = 2^{m-1} - 1$ et notons les permutations pyramidales, π_j pour $j = 0, 1 \dots \mu$.

Un ordonnancement est dit *actif* si le robot exécute toujours l'opération suivante dès que possible. Pour les ordonnancements actifs, tous les événements (débuts d'activités, attentes du robot...) sont définis de manière unique une fois que la séquence d'activités est donnée. Les seuls temps d'attente possibles se produisent devant des machines où le robot vide est prêt à exécuter la prochaine activité mais rencontre une pièce encore en cours d'usinage. Dans les chapitres suivants, nous ne considérons que les ordonnancements actifs.

2.2 Instances et temps de cycle

Une instance d'un problème de cellule robotisée à m machines est entièrement définie en indiquant les temps d'usinage, les temps de trajet et les temps de chargement/déchargement. Le temps d'usinage d'une pièce sur la machine $M_h (h = 1, 2 \dots m)$ est p_h. Soit δ_h le temps de trajet du robot (vide ou chargé) de M_h à M_{h+1} ou de M_{h+1} à $M_h (h = 0, 1 \dots m)$. Soit ϵ_h^l le temps de chargement (*loading*) d'une pièce sur M_h $(h = 1, 2 \dots m + 1)$ et soit ϵ_h^u le temps de déchargement (*unloading*) d'une pièce de M_h $(h = 0, 1 \dots m)$. Les durées sont additives. En effet, les machines sont disposées en ligne ou en arc de cercle, la vitesse du robot est constante et pour aller d'une machine à une autre, le robot passe par toutes les machines intermédiaires. Ainsi, le trajet du robot de M_h à $M_{h'}$ $(h \neq h')$ dure

$$\sum_{j=\min(h,h')}^{\max(h,h')-1} \delta_j \text{ unités de temps.}$$

Supposons que toutes les données d'une instance soient entières. Alors, si l'on considère que le cycle commence à l'instant 0, tous les événements suivants se produisent à des instants entiers. En effet, la seule opération algébrique est de déterminer le maximum entre le temps de trajet du robot pour arriver à une machine et le temps d'usinage d'une pièce sur la machine. Si la donnée d'une instance est en nombres rationnels, alors on multiplie tous les éléments de l'instance par le plus petit commun multiple pour obtenir à nouveau des arguments entiers. Nous restreignons donc notre étude à des instances entières.

Normalisons, sans restriction, le k-cycle C_k en commençant avec l'une des activités A_0. Le k-cycle C_k définit un unique vecteur initial d'incidence pièces/machines. Ce vecteur indique les machines occupées et les machines vides au début du cycle. Le *vecteur initial des temps d'usinage restants* de C_k, noté $P^0(C_k)$, indique les temps d'usinage restants des pièces sur les machines occupées au début de la première exécution du cycle. Le h-ème élément de $P^0(C_k)$ est égal à 0 si la machine M_h est vide au début du cycle et est dans l'intervalle $[0, p_h]$ sinon. Notons que tous les vecteurs qui vérifient ces propriétés ne sont pas réalisables pour un cycle donné. Nous n'incluons pas de phase initiale au cycle. Par exemple, on pourrait commencer

avec un système vide et charger d'abord les machines comme requis par le cycle qui est ensuite répété. En fait, la phase initiale ne change pas la durée du cycle à long terme.

Pour une instance donnée, nous représentons les k-cycles comme sur la Figure 2.3 : l'axe horizontal représente le temps. Le graphe indique la position du robot dans la cellule en fonction du temps. Les lignes pointillées sont les mouvements à vide du robot et les lignes pleines sont les mouvements du robot chargé, les processus de chargement/déchargement et les temps d'attente du robot aux machines.

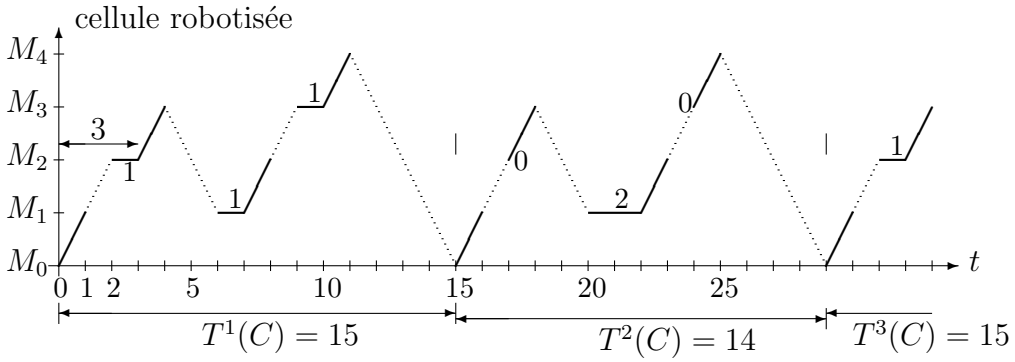

Figure 2.3: $C = (A_0A_2A_1A_3)$ pour l'instance I_0 avec $P^0(C) = (0, 3, 0)$

Par exemple, dans une cellule à trois machines, considérons le 1-cycle $C = (A_0A_2A_1A_3)$. Soit I_0 l'instance suivante :

$$\delta_h = 1 \quad (h = 0, 1, 2, 3) \; ; \quad \epsilon_h^u = \epsilon_{h+1}^l = 0 \quad (h = 0, 1, 2, 3) \; ;$$
$$p_1 = 6 \; ; \quad p_2 = 9 \; ; \quad p_3 = 6.$$

Au début du cycle $C = (A_0A_2A_1A_3)$, la machine M_2 est chargée et les machines M_1 et M_3 sont vides. Le vecteur $P^0(C) = (0, 3, 0)$ est un vecteur initial des temps d'usinage restants possible pour C. À l'instant 9, dans la Figure 2.3, le robot est à la machine M_3 et attend 1 unité de temps que la pièce soit prête afin d'exécuter l'activité A_3.

On remarque que le cycle C ne se répète pas de manière identique. Étudions ce phénomène.

Soit $T^i(C_k)$ la durée de la i-ème exécution du cycle C_k. Nous décomposons $T^i(C_k)$ en trois éléments, le temps de trajet, le temps de chargement/déchargement et le temps d'attente :

- $T_T(C_k)$ est le temps de trajet total du robot, c'est-à-dire,

$$T_T(C_k) = \sum_{h=0}^{m} m_h(C_k)\delta_h.$$

Remarquons que le temps de trajet est le même pour toutes les exécutions du cycle C_k. Par conséquent, il n'est pas indicé par i.

- $T_L(C_k)$ est le temps total de chargement/déchargement des pièces, c'est-à-dire,
$$T_L(C_k) = k \sum_{h=0}^{m} \epsilon_h^u + k \sum_{h=1}^{m+1} \epsilon_h^l.$$
$T_L(C_k)$ est contant et par conséquent il est indépendant de i.

- $T_W^i(C_k)$ est le temps total d'attente du robot aux machines durant la i-ème exécution de C_k.

Notons $w^i(C_k) = (w_1^i(C_k), w_2^i(C_k) \dots w_m^i(C_k))$, la *$i$-ème matrice des temps d'attente* de C_k. La matrice $w^i(C_k)$ est de taille $k \times m$, où les colonnes $w_h^i(C_k)$ sont les temps d'attente (vecteurs de taille k) à la machine $M_h (h = 1, 2 \dots m)$. La *matrice des temps d'attente initiaux* $w^1(C_k)$ est définie de manière unique par le vecteur initial $P^0(C_k)$. En effet, $w^1(C_k)$ ne dépend que de l'initialisation du cycle.

Le temps d'attente total $T_W^i(C_k)$ est la somme de tous les éléments de la matrice $w^i(C_k)$. Le i-ème temps de cycle, $T^i(C_k)$, vérifie

$$T^i(C_k) = T_T(C_k) + T_L(C_k) + T_W^i(C_k).$$

Le k-cycle C_k est *stationnaire* pour le vecteur initial $P^0(C_k)$ et pour l'instance I, si la séquence $w^i(C_k)$ est stationnaire, c'est-à-dire, pour i assez grand, toutes les matrices $w^i(C_k)$ sont égales. Le k-cycle C_k est *stable* s'il est stationnaire pour tout vecteur $P^0(C_k)$, et qu'il engendre le même temps d'attente pour tous les vecteurs $P^0(C_k)$. Si C_k est stable, nous notons le *temps de cycle* de C_k, $T(C_k) = T^i(C_k)$ pour i assez grand. Si C_k n'est pas stable, alors on a la propriété suivante pour des instances entières et donc pour des temps d'attente entiers. Cette proposition peut être étendue à des nombres rationnels.

Proposition 4 *Pour toute instance et pour tout vecteur initial des temps d'usinage restants, la suite $w^i(C_k)$ est cyclique, c'est-à-dire, il existe un entier l tel que, pour i assez grand, $w^{i+l}(C_k) = w^i(C_k)$.*

Démonstration Soit C_k un k-cycle. Considérons une instance entière et un vecteur initial des temps d'usinage restants $P^0(C_k)$. Ils définissent de manière unique la matrice initiale, à coefficients entiers, des temps d'attente, $w^1(C_k)$. On peut remarquer que $w^i(C_k)$ ne dépend que de l'instance et du vecteur $w^{i-1}(C_k)$. Par conséquent, s'il existe un indice i et un entier l qui vérifient $w^{i+l}(C_k) = w^i(C_k)$ alors, on a $w^{i+l+1}(C_k) = w^{i+1}(C_k)$.

Mais, chaque composant de $w_h^i(C_k)$ est un nombre entier qui se situe entre 0 et p_h. Par conséquent, il existe un nombre fini de matrices des temps d'attente possibles, $w^i(C_k)$. Ceci implique qu'il existe un l et un indice i tel que $w^{i+l}(C_k) = w^i(C_k)$. □

Notons $l(C_k)$ la plus petite périodicité du k-cycle C_k, c'est-à-dire, $l = l(C_k)$ est le plus petit entier positif qui vérifie $w^{i+l}(C_k) = w^i(C_k)$ pour i assez grand.

Proposition 5 *Tout entier l' qui vérifie $w^{i+l'}(C_k) = w^i(C_k)$, pour i assez grand, est un multiple de $l(C_k)$.*

Démonstration Soit l' un nombre entier qui vérifie $w^{i+l'}(C_k) = w^i(C_k)$, ou en abrégé $w^{i+l'} = w^i$, pour i assez grand. Comme $l = l(C_k)$ est la plus petite périodicité du k-cycle C_k, il existe deux entiers positifs α et β tels que $l' = \alpha l + \beta$ où $\beta < l$. On a

$$w^i = w^{i+l'} = w^{i+\alpha l+\beta} = w^{i+\beta}.$$

Comme $\beta < l$, on a $\beta = 0$ et, par conséquent, l' est un multiple de l. □

Nous avons vu que $w^i(C_k)$ est cyclique, ce qui implique que $T^i(C_k)$ est aussi cyclique. Pour l'exemple de la Figure 2.3, $T_T(C) = 12$ et $T_L(C) = 0$. Comme $k = 1$, $w_h^i(C)$ n'a qu'une seule composante. On obtient $w^1(C) = (1, 1, 1)$ ce qui donne $T_W^1(C) = 3$ et $T^1(C) = 15$. De même, $w^2(C) = (2, 0, 0)$, ce qui donne $T_W^2(C) = 2$ et $T^2(C) = 14$. Pour le cycle C, l'instance I_0 et le vecteur initial $P^0(C) = (0, 3, 0)$ on a, pour tout $i > 1$,

$$w^{2i+1}(C) = (1, 1, 1) \qquad \text{et} \qquad w^{2i}(C) = (2, 0, 0).$$

Donc, le cycle $C = (A_0A_2A_1A_3)$ n'est pas stationnaire pour $P^0(C_k) = (0, 3, 0)$ et pour l'instance I_0 : les durées d'exécution du cycle sont alternativement 14 et 15. On peut vérifier que C est stationnaire pour l'instance I_0 et pour le vecteur initial $P^0(C) = (0, 2.5, 0)$ avec une durée d'exécution de 14.5 (qui est aussi la moyenne de 14 et 15).

Notons que le 1-cycle $C = (A_0A_2A_1A_3)$ n'est pas une permutation pyramidale. Il semble qu'une propriété caractéristique des permutations pyramidales soit leur stabilité. Nous ne connaissons pas de preuve pour le cas général d'une cellule à m machines. Les preuves pour $m = 2$ et $m = 3$ peuvent être déduites de la littérature [66, 40]. Dans l'Annexe B (page 159), nous donnons une preuve explicite de la stabilité des permutations pyramidales pour des cellules à deux, trois et quatre machines.

Nous savons que la plus petite périodicité dépend de $P^0(C_k)$, mais nous n'avons pas d'exemple où la valeur moyenne de $T^i(C_k)$ pendant une période complète change pour des $P^0(C_k)$ différents. Nous pensons que la valeur moyenne de $T^i(C_k)$ ne dépend pas de $P^0(C_k)$ (ergodicité). Dans ce cas, les Propositions 4 et 5 permettent de définir le temps de cycle à long terme $T(C_k)$ qui est une valeur moyenne de $T^i(C_k)$ pour une période, c'est-à-dire, pour i assez grand et pour l égal à la taille d'une période,

$$T(C_k) = T_T(C_k) + T_L(C_k) + \frac{1}{l}\sum_{q=1}^{l} T_W^{i+q}(C_k).$$

Si la valeur moyenne de $T^i(C_k)$ dépend de $P^0(C_k)$, alors $T_W(C_k)$ doit être défini plus soigneusement :

$$T_W(C_k) = \inf_{P^0(C_k)} \left(\frac{1}{l} \sum_{q=1}^{l} T_W^{i+q}(C_k) \right).$$

Notons $T(C_k)$ le *temps de cycle* et $T(C_k)/k$ la *longueur du cycle* C_k [55]. Le *taux de production* de C_k est défini par $k/T(C_k)$. Ainsi, le ρ-cycle C_ρ est *optimal* s'il maximise le taux de production ou, de manière équivalente, minimise la longueur du cycle $T(C_k)/k$ parmi tous les k-cycles ($k = 1, 2, 3 \ldots$) possibles. L'approche dans [25] est, dans un sens, différente. Une périodicité stricte est obtenue en décalant les événements sous la forme d'un chemin critique. Ceci semble nécessiter un vecteur initial spécifique P^0 et la propriété des ordonnancements actifs semble perdue.

Définition 2 *Soit S_1 et S_2 deux ensembles de cycles de production. S_1 domine S_2 si, pour toute instance, on a la propriété suivante : pour tout k'-cycle $C_{k'}$ de S_2, il existe un k-cycle C_k dans S_1 qui vérifie*

$$\frac{T(C_k)}{k} \leq \frac{T(C_{k'})}{k'}.$$

Dans [27], les auteurs prouvent que les permutations pyramidales dominent les 1-cycles. Ils donnent un algorithme de complexité $O(m^3)$ pour déterminer la meilleure permutation pyramidale. L'intérêt porté aux 1-cycles est motivé par la conjecture suivante proposée par Sethi, Sriskandarajah, Sorger, Błażewicz et Kubiak [66].

Conjecture des 1-cycles *[66] l'ensemble des 1-cycles domine l'ensemble des cycles de production. Ceci signifie que le taux de production maximum sur toutes les séquences finies de mouvements cycliques du robot peut être obtenu en exécutant un 1-cycle.*

Cette conjecture est valide pour $m = 2$ [66] et $m = 3$ (première preuve dans [26]). Dans [26], les auteurs utilisent des graphes d'état et supposent que $\delta_h = \delta(h = 0, 1 \ldots m)$ et $\epsilon_h^u = \epsilon_{h+1}^l = 0(h = 0, 1 \ldots m)$. Il est mentionné dans la conclusion que la démonstration peut être généralisée à un système avec des temps de chargement/déchargement non nuls pour les machines et des valeurs arbitraires de temps de trajet δ_h. L'intérêt de cette conjecture est qu'elle réduit la complexité du problème. En effet, il suffit de trouver la meilleure permutation pyramidale pour obtenir le meilleur 1-cycle (problème polynomial).

Présentons quelques propriétés des temps de cycle. Nous utilisons l'abréviation suivante

$$\Delta_h = 2\delta_{h-1} + 2\delta_h + \epsilon_{h-1}^u + \epsilon_h^l + \epsilon_h^u + \epsilon_{h+1}^l.$$

Théorème 1 *Le temps de cycle $T(C_k)$ d'un k-cycle C_k arbitraire satisfait*

$$T(C_k) \geq k(\Delta_h + p_h) \qquad h = 1, 2 \ldots m.$$

Démonstration Cette inégalité a été introduite pour $k = 1$ dans [27].

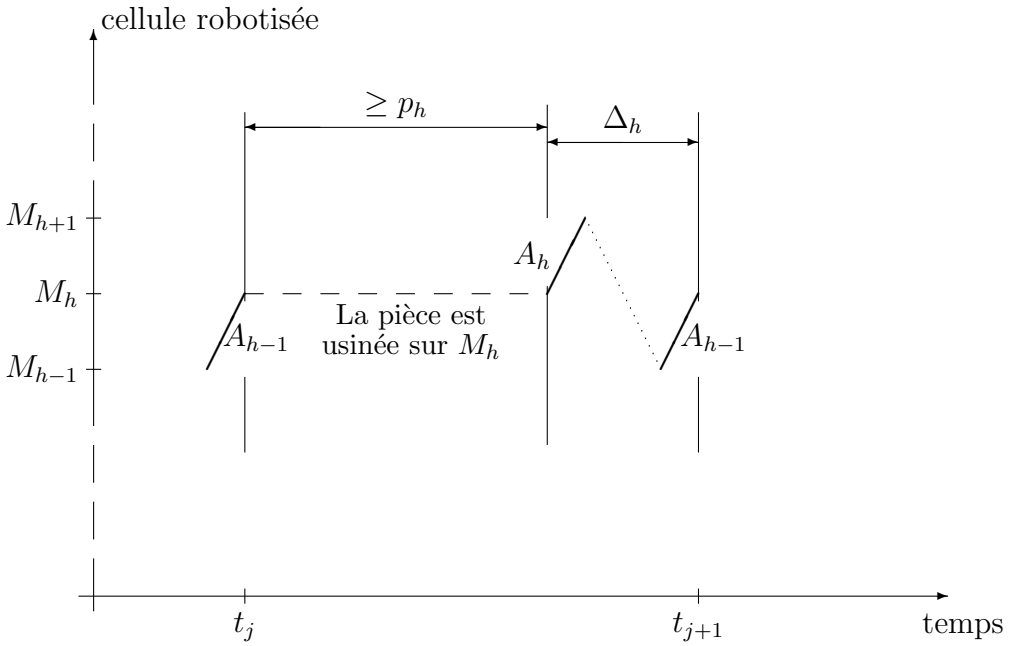

Figure 2.4: Illustration du Théorème 1

Pendant l'exécution de C_k, le robot exécute k fois les activités A_{h-1} et A_h. Ces activités ne peuvent pas être exécutées si la machine M_h est en train d'usiner une pièce. Soit $t_j (j = 1, 2 \ldots k+1)$ le j-ème instant dans le cycle où M_h commence à usiner une pièce. Nous supposons que le cycle commence à l'instant t_1 et finit à l'instant t_{k+1}. Entre t_j et $t_{j+1} (j = 1, 2 \ldots k)$, au moins les événements suivants se produisent séquentiellement [Figure 2.4] :

- la machine M_h usine une pièce [durée : p_h unités de temps],
- la pièce est transférée de M_h à M_{h+1} [$\epsilon^u_h + \delta_h + \epsilon^l_{h+1}$ unités de temps],
- le robot retourne à la machine M_{h-1} [$\delta_{h-1} + \delta_h$ unités de temps],
- une nouvelle pièce est transférée de M_{h-1} à M_h [$\epsilon^u_{h-1} + \delta_{h-1} + \epsilon^l_h$ unités de temps].

Donc $t_{j+1} - t_j \geq p_h + \Delta_h$. Comme $T^i(C_k) \geq t_{k+1} - t_1$ pour tout i, on a, pour chaque exécution, $T^i(C_k) \geq k(\Delta_h + p_h)$. On obtient donc $T(C_k) \geq k(\Delta_h + p_h)$. □

Le théorème suivant est une conséquence immédiate du Théorème 1.

Théorème 2 *Si, pour une instance donnée I, le temps de cycle d'un k-cycle C_k satisfait $T(C_k) = k(\Delta_h + p_h)$ pour un h $(1 \leq h \leq m)$, alors C_k est optimal pour I.*

Nous décrivons maintenant le temps de cycle d'une permutation pyramidale qui joue un rôle central : la permutation descendante, $\pi_d = (A_0 A_m A_{m-1} \ldots A_1)$. En effet, π_d est optimale si les temps d'usinage sont longs et, dans le cas contraire, π_d peut être exécutée sans attente. La Proposition suivante a été énoncée dans [27].

Proposition 6 *[27] Le temps de cycle de* $\pi_d = (A_0 A_m A_{m-1} ... A_1)$ *vérifie*

$$T(\pi_d) = \max_h (2\delta_0 + 4\sum_{j=1}^{m-1} \delta_j + 2\delta_m + \sum_{j=0}^{m} (\epsilon_{j+1}^l + \epsilon_j^u), \Delta_h + p_h). \qquad (2.5)$$

2.3 Graphes d'état

Le graphe d'état, G_m, défini pour les cellules à m machines, permet de décrire tous les mouvements possibles d'un robot dans la cellule. Ce type de graphes été introduit par Crama et van de Klundert dans [74, 29].

2.3.1 Présentation

Le *graphe d'état* G_m, associ aux cellules robotisées m machines, est une paire (V, E), où V est l'ensemble des sommets et E l'ensemble des arcs. Chaque sommet v_i $(0 \leq i \leq 2^m - 1)$ représente un état de la cellule (machines vides et machines occupées). La valeur de i, écrite en nombre binaire sur m bits, donne l'état du système décrit par le sommet v_i : si la machine M_q $(q = 1, 2 \ldots m)$ est occupée, alors le q-ème bit (de gauche à droite) est égal à 1, sinon ce bit est à 0. Par exemple, pour $m = 5$, le sommet v_9 [Figure 2.5] représente l'état du système pour lequel les machines M_2 et M_5 sont occupées et les machines M_1, M_3 et M_4 sont vides puisque la forme binaire de 9 est 01001. Les arcs de G_m représentent les activités du robot pour passer d'un état à un autre état. Chaque arc est pondéré par une activité. Par exemple, dans G_5, l'arc (v_{17}, v_9) est pondéré par l'activité A_1. En effet, A_1 signifie que le robot avance une pièce de la machine M_1 à la machine M_2 et l'état du système passe de 10001 01001. La Figure 2.5 présente le voisinage de v_9 dans G_5.

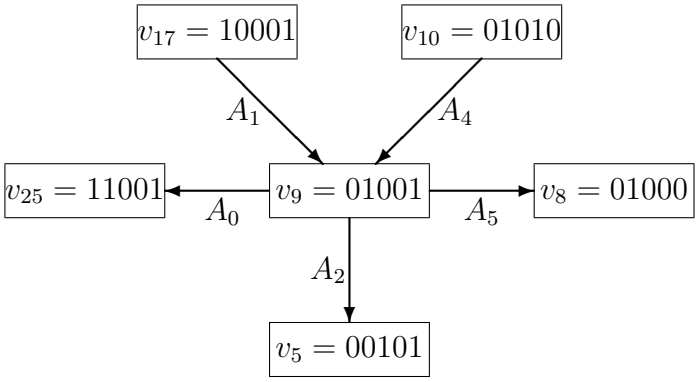

Figure 2.5: Le voisinage de v_9 dans G_5

L'ensemble E des arcs de G_m peut être décrit plus formellement. En effet, (v_i, v_j) est un arc de G_m si et seulement si l'une des propriétés suivantes est vraie (a et b sont des nombres binaires) :

- la représentation binaire de i est $a10b$ et la représentation binaire de j est $a01b$ (une pièce est avancée d'une machine à la suivante) ou
- la représentation binaire de i est $0a$ et la représentation binaire de j est $1a$ (une pièce est introduite dans le système) ou
- la représentation binaire de i est $a1$ et la représentation binaire de j est $a0$ (une pièce est sortie du système).

Le graphe d'état G_2 est donné Figure 2.6 et le graphe d'état G_3 est donné Figure 2.7.

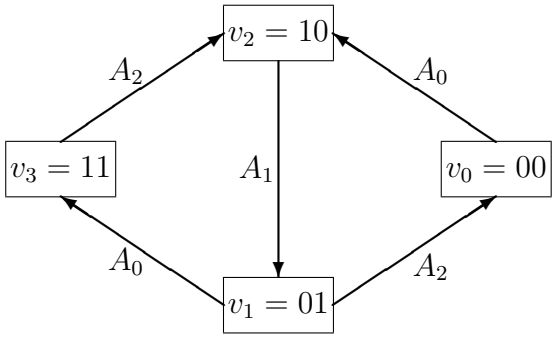

Figure 2.6: Le graphe d'état G_2

Toute séquence d'activités qui représente un k-cycle admissible correspond de manière unique (à une rotation des activités près) à un cycle dans G_m et réciproquement. Or, dans un k-cycle, chacune des $(m + 1)$ activités est répétée k fois. Ainsi, la longueur de tout circuit de G_m est un multiple de $(m + 1)$. Dans la suite, on représente indifféremment un k-cycle par une séquence de sommets de G_m ou par une séquence d'activités.

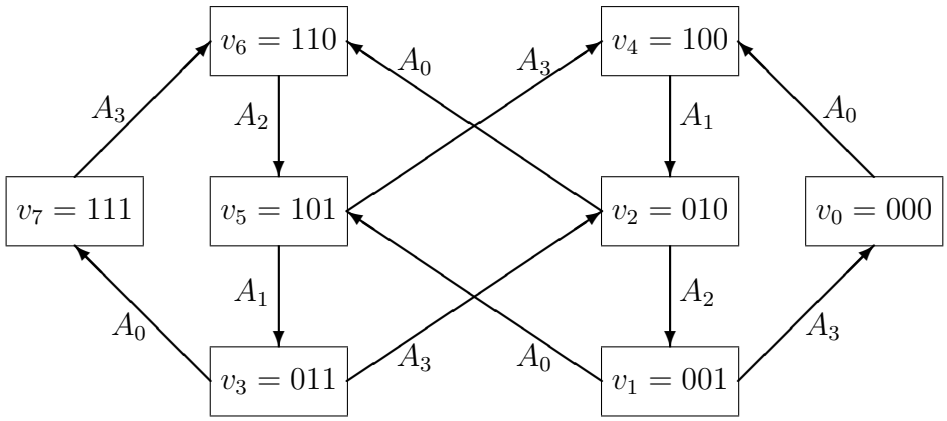

Figure 2.7: Le graphe d'état G_3

2.3.2 Propriétés générales

Les graphes d'état permettent d'établir des bornes sur les temps de cycle. De plus, ils possèdent quelques propriétés remarquables des graphes.

Le graphe d'état G_m admet deux couplages parfaits disjoints. En effet, les arcs pondérés par A_0 sont les arcs (v_i, v_j) tels que $v_i = 0a$ et $v_j = 1a$, pour tout a compris entre 0 et $2^{m-1} - 1$. Donc tout sommet est soit l'extrémité initiale, soit l'extrémité finale de exactement un arc pondéré par A_0. Les arcs pondérés par A_0 forment donc un couplage parfait. De même, les arcs pondérés par A_m forment un couplage parfait. Or, un arc est pondéré par exactement une activité. Donc, les deux couplages parfaits, celui formé par les arcs pondérés par A_0 et celui formé par les arcs pondérés par A_m, sont disjoints.

On considère les sommets qui représentent les états où la machine M_1 est vide. On les appelle *sommets initiaux*. En effet, par convention, on normalise les 1-cycles en commenant par A_0. Or, avant l'exécution de A_0, il faut que la machine M_1 soit vide pour que le robot puisse y apporter une pièce. La proposition suivante a été établie par Sethi et al. [66].

Proposition 7 [66] *Tout 1-cycle π définit un sommet initial unique v^π.*

Démonstration Soit π un 1-cycle. L'algorithme suivant construit l'unique sommet initial pour lequel π est admissible. On note $v^\pi(q) = 1$ s'il y a, au début du cycle π, une pièce sur M_q et $v^\pi(q) = 0$ sinon.

Algorithme [Détermination de v^π]

```
initialisation
   déjà(0) := faux
   pour i := 1 à m faire
      v_π(i) := 0
      déjà(i) := faux
   fin pour
début
   pour i := 1 à (m + 1) faire
      j := index de l'activité en i-ème position dans π
      si j ≠ 0 et non déjà(j − 1) alors
         v_π(j) := 1
      fin si
      déjà(j) := vrai
   fin pour
fin.
```

□

Dans [27], Crama et van de Klundert ont montr que les permutations pyramidales dominent les 1-cycles. Ils décrivent un algorithme en $O(m^3)$ qui donne la meilleure

permutation pyramidale. Nous savons qu'il existe 2^{m-1} sommets initiaux possibles (pas de pièce sur M_1). Le théorème suivant nous permet d'affirmer qu'il existe aussi 2^{m-1} permutations pyramidales.

Proposition 8 *Il existe une bijection entre l'ensemble des permutations pyramidales et l'ensemble des sommets initiaux.*

Démonstration La Proposition 7 indique qu'à toute permutation pyramidale correspond un sommet initial unique. Il nous reste donc à prouver que tout sommet initial définit une permutation pyramidale unique.

Soit π une permutation pyramidale et $h < h'$. Si l'activité A_h est après l'activité $A_{h'}$ dans π, alors A_h est une activité descendante. Si A_h précède $A_{h'}$ dans π, alors A_h est une activité montante.

À partir de l'état initial v, on obtient un partitionnement en activités montantes et descendantes de la manière suivante. Si la machine M_{h+1} est occupée dans l'état initial, alors l'activité A_h est exécutée après l'activité A_{h+1} ce qui implique que A_h est une activité descendante. Si M_{h+1} est initialement inoccupée, A_h est exécutée avant A_{h+1} et, par conséquent, A_h est une activité montante. Or, l'ensemble des activités montantes et descendantes définit de manière unique une permutation pyramidale. □

Par exemple, le sommet initial $v_\pi = 0111010$ correspond à la permutation pyramidale $\pi = (A_0A_4A_6A_7A_5A_3A_2A_1)$.

Soit E_m l'ensemble des arcs du graphe G_m pour lesquels toutes les machines sont vides et le robot est en train de transporter l'unique pièce présente dans le système. L'ensemble E_m est défini par

$$E_m = \{(v_o, v_{2^{m-1}}), (v_1, v_0)\} \cup \{(v_{2^{q+1}}, v_{2^q}) \text{ pour } q = 0, 1 \ldots m-2\}.$$

Par exemple, dans G_3, l'arc (v_2, v_1) ($v_2 = 010$ et $v_1 = 001$) est dans E_3 car sur cet arc toutes les machines sont vides et le robot transporte une pièce de la machine M_2 à la machine M_3. On peut remarquer que E_m est l'ensemble des arcs qui composent le 1-cycle identité, $Id = (A_0A_1 \ldots A_m)$. Le théorème suivant est appelé *Théorème d'additivité*.

Théorème 3 *Si un k-cycle C_k (k fini) couvre au moins deux fois le même arc de E_m alors C_k peut être décomposé en deux cycles C_{k_1} et C_{k_2} ($k_1 > 0$ et $k_2 > 0$) tels que*

- *C_{k_1} est un k_1-cycle ; C_{k_2} est un k_2-cycle ;*
- *$k_1 + k_2 = k$, ;*
- *$C_k = (C_{k_1}, C_{k_2})$;*
- *$T(C_k) = T(C_{k_1}) + T(C_{k_2})$.*

De plus, si C_k est optimal, alors C_{k_1} et C_{k_2} sont aussi optimaux.

Démonstration Soit C_k, un k-cycle qui couvre deux fois l'arc $(v_i, v_j) \in E_m$

pondéré par A_h. On peut écrire C_k de la manière suivante,

$$C_k = (\underbrace{v_i, v_j, \ldots}_{C_{k_1}}, \underbrace{v_i, v_j, \ldots}_{C_{k_2}}).$$

Soit t_ℓ le ℓ-ème instant dans le cycle C_k où le robot quitte le sommet v_i afin d'exécuter l'activité A_h. On sait qu'à l'instant t_ℓ toutes les machines sont vides et le robot transporte l'unique pièce du système. Par conséquent, à l'instant t_ℓ, l'état de la cellule est exactement le même qu'à l'instant $t_{\ell+1}$. Supposons que C_k commence à t_1. Entre deux occurrences de (v_i, v_j), le robot exécute un cycle complet dans le graphe. Notons C_{k_1} le k_1-cycle qui commence à l'instant t_1 et finit à l'occurrence suivante de (v_i, v_j) dans C_k. Ainsi, C_{k_1} commence à l'instant t_1, finit à l'instant t_2 et l'état initial est exactement identique à l'état final. Donc chaque exécution de C_{k_1} dure $T(C_{k_1}) = t_2 - t_1$. De plus C_k finit à $t_\ell(\ell > 2)$. Notons C_{k_2} le k_2-cycle qui commence à t_2 (le robot quitte le sommet v_i) et se termine à la fin de C_k. L'état final et l'état initial de C_{k_2} sont identiques. Par conséquent, $T(C_{k_2}) = t_\ell - t_2$.

On sait que $T(C_{k_1})+T(C_{k_2}) = t_\ell - t_2 + t_2 - t_1$. Donc, $T(C_k) = T(C_{k_1})+T(C_{k_2})$ où C_{k_1} est un k_1-cycle, C_{k_2} est un k_2-cycle et C_k peut être écrit comme une concaténation de C_{k_1} et de C_{k_2}.

Si C_k est optimal alors, par définition,

$$\frac{T(C_{k_1})}{k_1}, \frac{T(C_{k_2})}{k_2} \geq \frac{T(C_k)}{k}$$

ce qui implique

$$kT(C_{k_1}) \geq k_1 T(C_k) \text{ et } kT(C_{k_2}) \geq k_2 T(C_k).$$

D'où

$$kT(C_k) = k(T(C_{k_1}) + T(C_{k_2})) \geq (k_1 + k_2)T(C_k) = kT(C_k).$$

Donc toutes les inégalités deviennent des égalités ce qui signifie que C_{k_1} et C_{k_2} sont aussi optimaux. □

Corollaire 1 *Pour une instance I donnée, soit ρ le plus petit entier tel qu'il existe un ρ-cycle optimal C_ρ. Alors C_ρ contient au plus une fois un même arc de E_m.*

Démonstration Supposons que C_ρ contienne au moins deux fois l'un des arcs de E_m. D'après le Théorème 3, C_ρ peut être décomposé en deux cycles optimaux C_{ρ_1} et C_{ρ_2} avec $\rho_1 < \rho$ et $\rho_2 < \rho$. Ceci est en contradiction avec la minimalité de ρ. □

2.3.3 Line-graphs

À chaque graphe G_m (orienté), est associé un *line-graph* LG_m (orienté) comme suit :

– les sommets de LG_m correspondent aux arcs de G_m ;
– (a, a') est un arc de LG_m si et seulement si il existe dans G_m un sommet v qui est l'extrémité finale de a et l'extrémité initiale de a' [Figure 2.8].

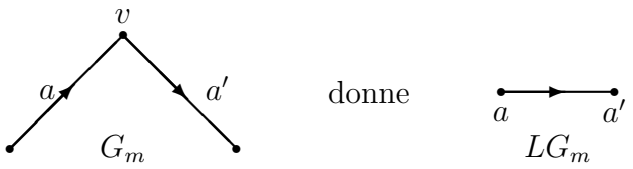

Figure 2.8: Définition d'un arc dans LG_m

Les circuits de G_m et de LG_m sont équivalents. On peut donc travailler avec l'un ou l'autre graphe. Le graphe LG_2 est présenté Figure 2.9 et le graphe LG_3 est présenté Figure 2.10.

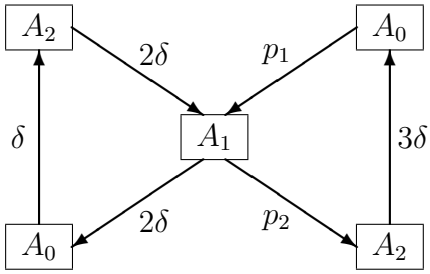

Figure 2.9: Le line-graph de G_2

Dans la suite, nous n'utilisons les line-graphs que pour le cas régulier, c'est-à-dire, $\delta_h = \delta$ et $\epsilon_h^u = \epsilon_{h+1}^l = \epsilon$ pour $h = 0, 1 \ldots m$. Par conséquent, nous décrivons les propriétés des line-graphs pour ce cas uniquement. Les arcs de LG_m sont pondérés par les durées des trajets du robot entre deux activités et par certains temps d'attente. En effet, un sommet de LG_m est étiqueté par l'activité qui pondère l'arc correspondant dans G_m. Ainsi, les arcs (A_h, A_{h+1}) de LG_m sont pondérés par le temps d'usinage en M_{h+1}, c'est-à-dire p_{h+1} : le robot attend p_{h+1} unités de temps entre A_h et A_{h+1} dans la séquence $A_h A_{h+1}$. L'arc $A_h A_{h'}$ avec $h' \neq h+1$ est pondéré par $\delta|h' - (h+1)|$. Cette valeur correspond au temps de trajet entre la fin de A_h (robot en M_{h+1}) et le début de $A_{h'}$ (robot en $M_{h'}$).

De même que le line-graph de G_2 est un sommet entouré de deux ¡¡ oreilles ¿¿ formées par l'identité et l'anti-identité [Figure 2.9], le line-graph de G_3 peut être représenté comme un cube possédant deux ¡¡ oreilles ¿¿, aussi formées par l'identité et l'anti-identité [Figure 2.10].

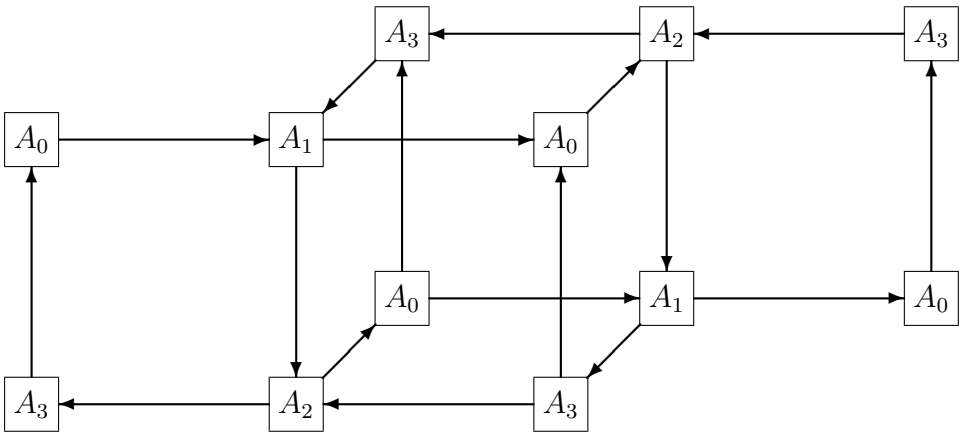

Figure 2.10: Le line-graph de G_3

2.4 Module de test et de calcul

Nous avons développé un module informatique en langage C afin de tester la validité des conjectures, d'exécuter des algorithmes et de calculer des résultats. Les développements qui n'aident pas à la compréhension de ce travail ne sont pas mentionnés.[1]

Ce module est composé de trois parties. La première partie permet d'observer l'évolution des cycles. La deuxième partie concerne les graphes d'état et la troisième partie traite des line-graphs.

2.4.1 Évolution des cycles

La première partie du module prend en entrée une instance à valeurs entières, un cycle et le nombre de répétitions du cycle. Le programme calcule le vecteur initial d'incidence pièces/machines et suppose qu'au début du cycle toutes les pièces sont prêtes. Cette hypothèse pourrait être supprimée, mais il faudrait alors entrer en plus l'état initial. La sortie est un tableau au format latex des événements dans la cellule et de l'état de la cellule à chaque instant.

Cette partie a permis de tester de nombreux cycles pour étudier, par exemple, la dominance des 1-cycles, l'ergodicité des cycles (en changeant l'état initial) ou leur stabilité. Le Tableau 3.4, page 82, a été généré directement par ce module.

[1] Ce module a été implémenté en collaboration avec Benjamin Vettier.

2.4.2 Graphes d'état et nombre de k-cycles

La seconde partie du module permet de générer des graphes d'état. La donnée est le nombre de machines et la sortie est le graphe d'état sous la forme

v_i : `liste des successeurs de` v_i `et pondération des arcs`

Par exemple le graphe d'état G_2 (voir Figure 2.6, page 41) est décrit comme suit

```
0 :  2 (A0)
1 :  0 (A2) 3 (A0)
2 :  1 (A1)
3 :  2 (A2)
```

Ainsi, la deuxième ligne signifie que les successeurs de v_1 sont v_0 et v_3 et que l'arc (v_1, v_0) est pondéré par l'activité A_2 et que l'arc (v_1, v_3) est pondéré par l'activité A_0.

Ce module nous a permis de calculer le nombre de k-cycles dans une cellule à m machines (m et k donnés). Dans un premier temps, nous décrivons un algorithme simple qui permet de calculer le nombre de k-cycles dans une cellule à deux machines. Puis, nous présentons une approche combinatoire qui fournit une formule récursive pour obtenir le nombre de k-cycles pour $m = 2$. Enfin, nous présentons un algorithme qui utilise les idées de l'approche combinatoire. Cet algorithme permet de calculer le nombre de k-cycles dans une cellule à m machines.

Approche algorithmique pour $m = 2$

Considérons une cellule à deux machines. Pour k donné, il s'agit de trouver le nombre de k-cycles. Si $k = 1$, alors il existe deux 1-cycles différents : $(A_0A_1A_2)$ et $(A_0A_2A_1)$. Remarquons que $(A_1A_2A_0)$ et $(A_2A_0A_1)$ sont identiques, à une rotation des activités près, au cycle $(A_0A_1A_2)$. De même, toujours pour deux machines, il existe trois 2-cycles différents : $(A_0A_1A_2A_0A_1A_2)$, $(A_0A_1A_0A_2A_1A_2)$ et $(A_0A_2A_1A_0A_2A_1)$.

En fait, pour deux machines, un k-cycle est une succession des séquences $a = A_1A_0A_2$ et $b = A_1A_2A_0$ (voir Figure 2.6, page 41). Il s'agit donc de compter les mots de k lettres, sur l'alphabet $\{a, b\}$, qui sont différents. Deux mots sont différents s'ils ne sont pas identiques à une rotation des lettres près. Ainsi, *aab* est différent de *bba* et est égal à *aba* et à *baa*.

L'idée de l'algorithme est de créer un tableau de taille 2^k, pour les nombres binaires de 0 à $2^k - 1$, puis de parcourir ce tableau en éliminant les nombres identiques. Le nombre s retourné à la fin de l'algorithme est le nombre de k-cycles dans une

Tableau 2.1: Nombre de k-cycles pour $m = 2$ (approche algorithmique)

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
nb de k-cycles	2	3	4	6	8	14	20	36	60	108	188	352	632	1182

k	15	16	17	18	19	20	21	22
nb de k-cycles	2192	4116	7712	14602	27596	52488	99880	190746

cellule à m machines.

Algorithme [Calcul du nombre de k-cycles pour $m = 2$]

```
donnée k
initialisation
   K = 2^k ;
   s = 0 ;
   pour  i = 0   K − 1 faire vu[i] = vrai ;
dbut
   pour  x = 0   K − 1 faire
     si  vu[x] = vrai alors
       s = s + 1 ;
       vu[x] = faux ;
       y = x ;
       pour  i = 0   k − 1 faire
         y = y ∗ 2 ;
         si  y ≥ K alors y = y − K + 1 ;
         vu[y] = faux ;
       fin pour
     fin si
   fin pour
   retourner s ;
fin.
```

Cet algorithme à été programmé en C et donne rapidement le nombre de k-cycles [Tableau 2.1].

Approche combinatoire pour $m = 2$

L'approche combinatoire fournit une formule qui permet d'obtenir le nombre de k-cycles pour deux machines. [2] Plus généralement, nous proposons des formules

[2]Ces formules ont été élaborées avec l'aide de Sylvain Gravier et Charles Payan du laboratoire Leibniz-IMAG à Grenoble.

qui permettent d'obtenir le nombre de mots de k lettres différents, sur un alphabet $\{a_1, a_2 \ldots a_n\}$ (où ¡¡ différent ¿¿ signifie toujours ¡¡ non identiques à une permutation des lettres près ¿¿).

Notons $B_n(k)$ le nombre de mots, de longueur k, différents et $M_n(d)$ le nombre de mots différents de longueur d non périodiques, c'est-à-dire, les mots qui ne sont pas la répétition d'un même motif. Par exemple, *abab* est périodique, mais *aaab* ne l'est pas. On a

$$M_n(k) = \frac{n^k - \sum_{d \ div \ k} dM_n(d)}{k}.$$

où $d \ div \ k$ signifie que ¡¡ d divise k ¿¿ et d peut être égal à 1 mais d est strictement plus petit que k ($d \neq k$). Le premier terme, n^k compte le nombre total de mots de longueur k. Il faut ensuite soustraire tous les mots périodiques. Puis, il faut diviser par k pour éliminer $(k-1)$ permutations de chaque mot restant.

Par exemple, pour $n = 2$ et $k = 4$, il existe 16 mots dont 2 sont périodiques de période 1 (*aaaa* et *bbbb*) et 1 est périodique de période 2 (*abab*). Il faut enlever ce mot deux fois (pour éliminer également *baba*). Il reste alors 12 mots non périodiques ($aabb = abba = bbaa = baab$; $aaab = aaba = abaa = baaa$; $bbba = bbab = babb = abbb$). Sur ces 12 mots, 3 sont différents. On a donc $M_2(4) = (16-2*1-1*2)/4 = 3$.

Pour obtenir tous les mots différents on calcule $B_n(k)$ comme suit :

$$B_n(k) = \sum_{d \ div \ k} M_n(d).$$

Pour $n = 2$, $B_2(k)$ donne tous les k-cycles dans une cellule à deux machines (Tableau 2.2). À partir de trois machines, le problème se complique car on ne peut plus décomposer les cycles en motifs intéressants.

Tableau 2.2: Nombre de k-cycles pour $m = 2$ (approche combinatoire)

k	1	2	3	4	5	6	7	8	9	10	11	12	13
$M_2(k)$	2	1	2	3	6	9	18	30	56	99	186	335	630
$B_2(k)$	2	3	4	6	8	14	20	36	60	108	188	352	632

Nombre de k-cycles pour m arbitraire

Nous présentons un algorithme qui calcule le nombre de k-cycles pour m quelconque. Cet algorithme utilise le principe présenté dans l'approche combinatoire.

Tableau 2.3: Nombre de k-cycles dans une cellule à m machines

k \ m	2	3	4	5	6	7
1	2	6	24	120	720	5040
2	3	20	260	5588	175112	7439072
3	4	70	3656	375984	65117280	
4	6	300	60648	29222424		
5	8	1350	1073696			
6	14	6580	19847316			
7	20	32646				
8	36	166620				
9	60	862470				

L'idée est d'énumérer tous les k-cycles en parcourant le graphe d'état et de donner à chaque cycle un poids. Ce poids correspond au nombre de fois où le cycle apparaît lors de l'énumération.

Un cycle qui apparaît n fois lors de l'énumération (cycle de période n) a un poids de $1/n$. Ainsi, les n occurrences du même cycle auront ensemble un poids de 1.

Algorithme [Calcul du poids du cycle C]

```
début
  poids = −1 ;
  pour tous les diviseurs d de k à partir de 1 faire
    si poids < 0 alors
      p = 1/d ; n=(m+1) d ;
      fini = faux ;
      pour i = 0 à n faire
        si (non fini) alors
          pour j = 1 à p − 1 faire
            si (non fini) alors fini = (C(i) ≠ C(i + jn)) ;
          fin pour ;
        fin si ;
      fin pour ;
      si (non fini) alors poids = p ;
    fin si ;
  fin pour ;
  renvoyer(poids) ;
fin.
```

Pour énumérer tous les k-cycles, l'algorithme de parcours démarre de chaque sommet initial avec l'arc pondéré par A_0. Puis il effectue un parcours en profondeur du

graphe. Il continue le parcours sur une branche tant que, pour tout h, le nombre de A_h rencontré est inférieur à k et que la branche ne décrit pas un k-cycle.

Le Tableau 2.3 indique que le nombre de k-cycles explose très rapidement. Ainsi, une analyse des conjectures en énumérant tous les k-cycles s'avère impossible.

2.4.3 Line-graphs et plus petits circuits moyens

Cette partie concerne les line-graphs des graphes d'état. Le programme prend en entrée le nombre de machines, m, et génère le line-graph LG_m.

Dans cette partie, nous considérons le cas régulier, c'est-à-dire $\delta_h = \delta$ et $\epsilon_h^u = \epsilon_{h+1}^l = \epsilon$, pour tout h.

Les arcs de LG_m sont pondérés par les temps de trajet ou par les temps d'attente engendrés par les séquences d'activités $A_h A_{h+1}$. Pour obtenir la durée complète d'un cycle il ne manque que les temps d'attente lorsque les temps d'usinage sont grands. Ainsi, si les temps d'usinage sont petits ($p_h \leq 4\delta$, pour tout $h = 1, 2 \ldots m$), alors la durée d'un cycle est obtenue en additionnant les poids des arcs de LG_m qui composent le cycle. À cette valeur, il faut ajouter $\delta + 2\epsilon$ (durée d'une activité) pour chaque sommet visité pendant le parcours du cycle.

Comme tous les circuits de LG_m correspondent à des cycles de production, ils ont une longueur multiple de $(m + 1)$. Ainsi, s'il n'y a pas d'attente, un cycle de production optimal correspond à un plus court circuit moyen dans LG_m. Nous définissons le plus court circuit moyen et décrivons un algorithme pour le trouver.

Soit $G = (V, A)$ un graphe orienté à n sommets et c_{ij}, une distance sur les arcs de G. Le plus court circuit moyen (*minimum mean cycle*) C de G est le circuit qui minimise

$$\frac{\text{somme des coûts des arcs de } C}{\text{nombre de sommets dans } C}$$

Trouver le plus court circuit moyen est un problème qui peut être résolu en temps polynomial. Dans la théorie des flots, il existe une très vaste littérature sur ce sujet. Des algorithmes de plus en plus rapides ont été conçus pour résoudre ce problème. Nous utilisons l'algorithme de Karp [49] décrit dans [4].

Fixons un sommet s de G. Soit $d^k(j)$ la longueur, par rapport à c_{ij}, du plus court chemin de s à j qui contient exactement k arcs. Pour tout sommet j et pour tout $k = 1, 2 \ldots n$, on peut calculer $d^k(j)$ en utilisant la formule récursive suivante :

$$d^k(j) = \min_{\{i:(i,j)\in A\}} \{d^{k-1}(i) + c_{ij}\}$$

avec $d^0(j) = 0$.

La longueur μ^* du plus court circuit moyen est

$$\mu^* = \min_{j \in V} \max_{0 \leq k \leq n-1} \left[\frac{d^n(j) - d^k(j)}{n-k} \right]$$

Nous avons implémenté cet algorithme pour trouver le plus court circuit moyen de LG_m. Nous avons commencé par simplifier le graphe en utilisant des propriétés intéressantes des k-cycles.

Pour une instance donnée, notons C^* le plus court circuit moyen minimal en nombre de sommets et k^* le degré de C^*, c'est-à-dire, C^* passe par $k^*(m+1)$ sommets. Notons que C^* ne passe pas deux fois par le même sommet.

Le circuit C^* passe k^* fois par chaque sommet A_h, $h = 1, 2 \ldots m$. Pour aller d'un sommet A_h au sommet A_h suivant, C^* emprunte le plus court chemin entre ces deux sommets. En effet, les chemins entre deux sommets pondérés par A_h et qui ne passent pas par d'autres sommets pondérés par A_h ont tous la même longueur.

Chaque activité $A_h, (h = 1, 2 \ldots m-1)$ pondère 2^{m-2} sommets de LG_m et les activités A_0 et A_m pondèrent 2^{m-1} sommets de LG_m. Fixons h entre 1 et $m-1$. Il suffit de considérer les plus courts chemins entre tous les sommets pondérés par A_h. On obtient un graphe complet de 2^{m-2} sommets. Il s'agit alors de trouver le plus court circuit moyen dans ce graphe.

Remarquons que cet algorithme n'est pas polynomial car LG_m possède $(m+3)2^{m-2}$ sommets et que le graphe réduit en possède 2^{m-2}.

D'autres parties du module ont été développées mais ne sont pas présentées dans ce mémoire. Par exemple, nous avons implémenté l'algorithme de Crama et van de Klundert pour trouver la meilleure permutation pyramidale [27]. Cet algorithme a été utilisé pour calculer le gain exact engendré par l'ajout d'espaces de stockage unitaires dans la cellule et pour générer la Figure 7.2, page 125.

Part II

Analyse de la production cyclique

Introduction de la deuxième partie

Cette partie concerne l'analyse de la production cyclique dans des flow-shops robotisés simples, sans espace de stockage et dont les distances inter-machines sont additives.

Nous étudions d'abord la Conjecture des 1-cycles lorsque les machines ne sont pas équidistantes (Chapitre 3). Dans ce cas, Sethi et al. [66] ont prouvé que la conjecture est vraie pour des cellules à deux machines et Crama et van de Klundert [29] ont prouvé qu'elle est vraie pour des cellules à trois machines. Nous présentons de nouvelles preuves de ces résultats avec deux approches différentes : une approche par les graphes et une approche algébrique. Nous utilisons cette dernière approche pour démontrer d'autres résultats connus de la littérature comme la dominance de certains cycles de production appelés permutations pyramidales [27]. Cette approche nous a également permis de trouver un contre-exemple à la Conjecture des 1-cycles pour des cellules à quatre machines et plus. Nous étudions ensuite la performance des cycles de production d'une pièce.

Dans le chapitre suivant (Chapitre 4), nous imposons des restrictions sur les paramètres de la cellule. Nous considérons d'abord le cas régulier où toutes les machines sont équidistantes. La Conjecture des 1-cycles avait initialement été proposée pour ce type de configuration. Pour des cellules régulières à quatre machines, les cycles de production de deux pièces sont dominés par les cycles de production d'une seule pièce. Par contre, il existe des cycles de production de trois pièces meilleurs que tous les cycles de production d'une pièce. Nous étudions ensuite le cas régulier équilibré où les temps d'usinage des pièces sont identiques sur toutes les machines. Dans ce cas, nous prouvons que le meilleur cycle de production d'une pièce peut être trouvé en temps constant et que la Conjecture des 1-cycles est vraie pour des systèmes à quatre machines.

Chapter 3

Conjecture des 1-cycles

Dans ce chapitre, nous établissons complètement la validité de la Conjecture des 1-cycles dans des cellules robotisées dont les durées de transfert sont additives. Nous présentons deux approches différentes de l'analyse de la production cyclique : une approche par les graphes (Section 3.1) et une approche algébrique (Section 3.2). Nous utilisons ces deux approches pour redémontrer la Conjecture des 1-cycles pour des cellules à deux et trois machines. L'approche algébrique nous a permis de trouver un contre-exemple à la conjecture pour des cellules à quatre machines et plus (Section 3.3). Puis, la conjecture étant fausse, nous analysons la performance des 1-cycles par rapport aux k-cycles.

3.1 Approche par les graphes

Dans cette section, nous démontrons la validité de la Conjecture des 1-cycles pour des cellules à deux et trois machines à l'aide de graphes d'état.[1]

Dans [26] les auteurs donnent la première preuve de la validité de la conjecture pour une cellule à trois machines. Nous présentons une nouvelle preuve, basée sur des graphes plus simples, comportant deux fois moins de sommets que les graphes utilisés dans [26].

Rappelons que $u_h(C_k)$ est le nombre total d'occurrences de $A_{h-1}A_h$ $(h = 1, 2 \ldots m)$ dans le k-cycle C_k et notons $W_h(C_k)$ l'attente totale en M_h pendant l'exécution de C_k.

[1] Les résultats présentés dans cette section ont été publiés dans la revue INFOR [15].

3.1.1 Étude de G_2

La première preuve de la validité de la Conjecture des 1-cycles dans une cellule à deux machines a été donnée par Sethi et al. [66]. Nous présentons une nouvelle preuve qui utilise les graphes d'état. Le graphe G_2 est donné Figure 3.1.

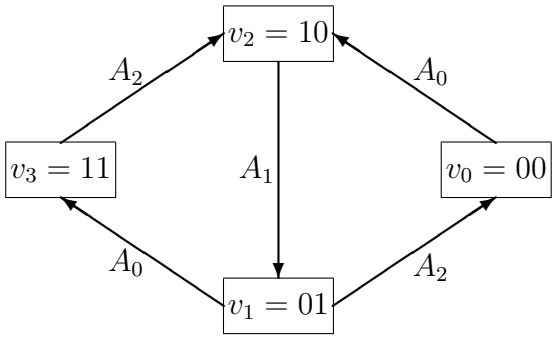

Figure 3.1: Le graphe d'état G_2 et son line-graph

Dans une cellule composée de deux machines, il existe exactement deux permutations pyramidales : l'identité $\pi_0 = (A_0, A_1, A_2)$ et l'anti-identité $\pi_1 = (A_0, A_2, A_1)$.

Théorème 4 *La Conjecture des 1-cycles est vraie pour $m = 2$.*

Démonstration Pour I une instance, soit ρ la plus petite valeur telle qu'il existe un ρ-cycle C_ρ optimal. Il suffit de prouver que $\rho = 1$.

Dans C_ρ, l'activité A_1 est répétée ρ fois. Dans G_2, seul l'arc (v_2, v_1) est pondéré par A_1 [Figure 3.1]. Donc, C_ρ contient ρ fois l'arc (v_2, v_1) qui appartient à l'ensemble E_2 (sur cet arc, toutes les machines sont vides). Le Corollaire 1 (page 44) indique que C_ρ contient au plus une seule fois tout arc de E_2. Donc, $\rho = 1$. □

3.1.2 Étude de G_3

Pour le cas de trois machines, le graphe G_3 est représenté Figure 3.2.

Comme indiqué précédemment, la première preuve de la Conjecture des 1-cycles pour $m = 3$ a été donnée par Crama et van de Klundert [26, 29]. Nous présentons une nouvelle preuve basée sur les graphes d'état.

Théorème 5 *La Conjecture des 1-cycles est vraie pour $m = 3$.*

Dans une cellule trois machines, il existe exactement six 1-cycles dont quatre sont des permutations pyramidales. Or, le temps de cycle minimum pour tous les 1-cycles est atteint pour une permutation pyramidale [27]. Les quatre permutations

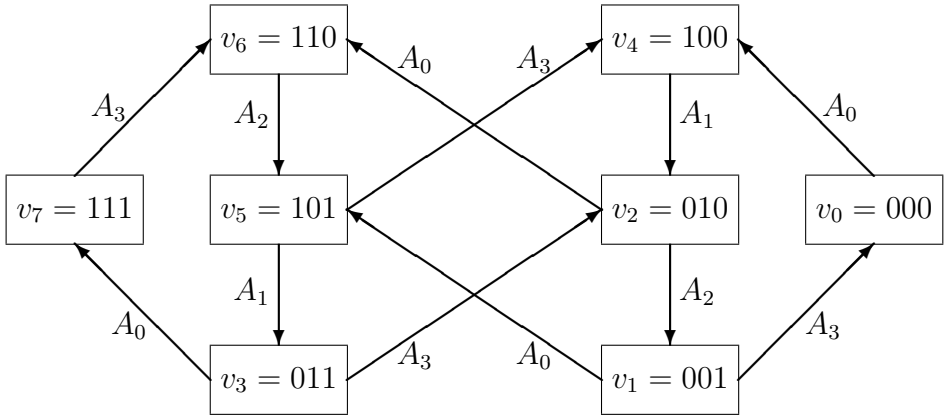

Figure 3.2: Le graphe d'état G_3

pyramidales sont :

$$\begin{aligned}
\pi_0 &= (A_0A_1A_2A_3)\ ;\\
\pi_1 &= (A_0A_1A_3A_2)\ ;\\
\pi_2 &= (A_0A_2A_3A_1)\ ;\\
\pi_3 &= (A_0A_3A_2A_1).
\end{aligned}$$

Les temps de cycle de ces permutations pyramidales sont donnés par [66] :

$$\begin{aligned}
T(\pi_0) &= 8\delta + 8\epsilon + p_1 + p_2 + p_3\ ;\\
T(\pi_1) &= 10\delta + 8\epsilon + p_1 + \max(0, p_3 - p_1 - 6\delta - 4\epsilon, p_2 - 4\delta - 2\epsilon)\ ;\\
T(\pi_2) &= 10\delta + 8\epsilon + \max(0, p_1 - p_3 - 6\delta - 4\epsilon, p_2 - 4\delta - 2\epsilon) + p_3\ ;\\
T(\pi_3) &= 12\delta + 8\epsilon + \max_h(0, p_h - 8\delta - 4\epsilon).
\end{aligned}$$

Considérons une instance I. Soit ρ la plus petite valeur telle qu'il existe un ρ-cycle C_ρ optimal. On démontre que $\rho = 1$. En utilisant les Propositions 1 et 2 (page 31) et en posant $u_h(C_\rho) = u_h$, on obtient l'inégalité suivante pour un plus petit cycle optimal C_ρ :

$$T(C_\rho) \geq (12\rho - 2u_1 - 2u_3)\delta + 8\rho\epsilon + W_1(C_\rho) + W_2(C_\rho) + W_3(C_\rho) \tag{3.1}$$

Nous pouvons imposer les conditions suivantes pour le reste de la démonstration :

$$u_1p_1 + u_2p_2 + u_3p_3 < (2u_1 + 2u_3)\delta \tag{3.2}$$

$$\max(p_1, p_2, p_3) < 8\delta + 4\epsilon \tag{3.3}$$

En effet, si $T(\pi_3) = 4\delta+4\epsilon+\max_h(p_h)$ alors π_3 est optimale sinon, $T(\pi_3) = 12\delta+8\epsilon$. Or si l'inégalité (3.2) ou l'inégalité (3.3) n'est pas vérifiée alors $T(C_\rho) \geq 12\rho\delta+8\rho\epsilon$ ou bien π_3 est optimale. Donc dans les deux cas, π_3 domine C_ρ ce qui implique que $\rho = 1$ et donc le Théorème 5 est vrai.

Avec l'inégalité (3.3), le temps de cycle de la permutation pyramidale π_3 satisfait

$$T(\pi_3) = 12\delta + 8\epsilon.$$

Nous établissons maintenant quelques lemmes à propos du ρ-cycle C_ρ (ρ minimal comme expliqu ci-dessus).

Lemme 1 *Pour le ρ-cycle C_ρ, les propriétés suivantes sont vérifiées :*
(1.1) *C_ρ contient au moins u_1 fois l'arc (v_2, v_1) et u_3 fois l'arc (v_4, v_2) ;*
(1.2) *$u_h \leq 1$ pour $h = 1, 2$ et 3 ;*
(1.3) *si $u_2 = 1$ alors soit $\rho = 1$ soit $u_1 = u_3 = 1$.*

Démonstration de la propriété (1.1) Montrons d'abord que chaque fois que C_ρ contient la séquence A_0A_1, il contient l'arc (v_2, v_1). Supposons que la séquence A_0A_1 soit représentée par la séquence de sommets (v_0, v_4, v_2). L'unique arc qui arrive en v_0 est l'arc (v_1, v_0) et l'unique arc qui arrive en v_1 est l'arc (v_2, v_1). Donc, toute séquence (v_0, v_4, v_2) est toujours incluse dans la séquence $(v_2, v_1, v_0, v_4, v_2)$ qui contient l'arc (v_2, v_1).

Supposons que la séquence A_0A_1 soit représentée par la séquence de sommets (v_1, v_5, v_3). L'unique arc qui arrive en v_1 est l'arc (v_2, v_1). Donc, chaque séquence (v_1, v_5, v_3) est toujours précédée immédiatement par l'arc (v_2, v_1). Par conséquent, C_ρ contient au moins u_1 fois l'arc (v_2, v_1).

De la même manière, on peut prouver que toute séquence de sommets qui représente la séquence A_2A_3 est toujours suivie par l'arc (v_4, v_2) avant la prochaine occurrence de A_2. □

Démonstration de la propriété (1.2) On sait que C_ρ ne contient pas plusieurs fois le même arc de E_3 (Corollaire 1, page 44). La propriété (1.1) indique que pour $h = 1$ et $h = 3$, C_ρ contient au moins u_h fois un arc de E_3 : au moins u_1 fois (v_2, v_1) et u_3 fois (v_4, v_2). De plus, nous avons u_2 fois la séquence de sommets (v_4, v_2, v_1). Donc C_ρ contient u_2 fois l'arc (v_4, v_2) et u_2 fois l'arc (v_2, v_1) qui sont tous deux dans E_3. Donc $u_h \leq 1$ pour $h = 1, 2$ et 3. □

Démonstration de la propriété (1.3) Posons $u_2 = 1$. On peut remarquer que le seul arc qui arrive en v_0 est l'arc (v_1, v_0) et que le seul arc qui arrive en v_1 est l'arc (v_2, v_1). De plus, le seul arc qui quitte v_0 est l'arc (v_0, v_4) et le seul arc qui quitte v_4 est l'arc (v_4, v_2). Par conséquent, si C_ρ contient le sommet v_0 alors C_ρ contient la séquence $(v_2, v_1, v_0, v_4, v_2)$ qui représente la séquence d'activités $A_2A_3A_0A_1$. Ceci implique que $u_1 \geq 1$ et $u_3 \geq 1$. Supposons que C_ρ ne contienne pas le sommet v_0. Comme $u_2 = 1$, C_ρ contient la séquence $\sigma = (v_5, v_4, v_2, v_1, v_5)$. On sait que C_ρ ne contient pas plus d'une fois le même arc de E_3. Donc, soit $C_\rho = \sigma$, c'est--dire $\rho = 1$, soit la séquence σ est précédée par le sommet v_6 et suivie par le sommet v_3. Dans ce cas, C_ρ contient la séquence $(v_6, v_5, v_4, v_2, v_1, v_5, v_3)$ qui représente la séquence d'activités $A_2A_3A_1A_2A_0A_1$. Donc $u_1 = 1$ et $u_3 = 1$. □

Lemme 2 *Les trois propriétés suivantes sont équivalentes pour le ρ-cycle optimal*

C_ρ :

(2.1) $\rho = 1$ *ou*

$$\begin{aligned} W_1(C_\rho) + W_2(C_\rho) + W_3(C_\rho) \geq \\ u_1p_1 + u_2p_2 + u_3p_3 \\ +(1-u_2) \quad [u_1 \max(0, p_3 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon) \\ +u_3 \max(0, p_1 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)] \ ; \end{aligned}$$

(2.2) $\rho = 1$ *ou*

$$\begin{aligned} T(C_\rho) \quad \geq \quad & (\rho-2)T(\pi_3) \\ & +(1-u_2)[u_1T(\pi_1) + u_3T(\pi_2) + (2-u_1-u_3)T(\pi_3)] \\ & +u_2[T(\pi_3) + T(\pi_0)]; \end{aligned}$$

(2.3) $\rho = 1$.

La propriété (2.3) implique (2.1) de manière évidente. Il suffit donc de prouver que la propriété (2.1) implique la propriété (2.2) et que la propriété (2.2) implique la propriété (2.3). On peut remarquer que cette dernière propriété est équivalente au Théorème 5.

Démontrons d'abord que la propriété (2.1) implique la propriété (2.2).

Démonstration En utilisant l'inégalité (3.1) et la propriété (2.1), on obtient

$$\begin{aligned} T(C_\rho) \geq \quad & (12\rho - 2u_1 - 2u_3)\delta + 8\rho\epsilon + u_1p_1 + u_2p_2 + u_3p_3 \\ & +(1-u_2) \quad [u_1 \max(0, p_3 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon) \\ & \qquad +u_3 \max(0, p_1 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)]. \end{aligned}$$

Or d'après la propriété (2.1), u_2 ne peut prendre que les valeurs 0 ou 1. Cette égalité peut donc être réécrite de la manière suivante :

$$\begin{aligned} T(C_\rho) \geq \quad & 12(\rho-2)\delta + 8(\rho-2)\epsilon \\ & +(1-u_2)(12 - 2u_1 + 12 - 2u_3)\delta \quad + \quad u_2(12 + 12 - 2u_1 - 2u_3)\delta \\ & +(1-u_2)(8+8)\epsilon \quad + \quad u_2(8+8)\epsilon \\ & +(1-u_2)(u_1p_1 + u_2p_2 + u_3p_3) \quad + \quad u_2(u_1p_1 + u_2p_2 + u_3p_3) \\ & +(1-u_2)[u_1 \max(0, p_3 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon) \\ & \qquad +u_3 \max(0, p_1 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)]. \end{aligned}$$

La propriété (1.3) indique que, si $u_2 = 1$, alors $\rho = 1$ (et (2.2) est valide) ou $u_1 = u_3 = 1$. Ainsi, les égalités suivantes sont valides pour $u_2 = 0$ et pour $u_2 = 1$:

$$\begin{aligned} & u_2(12 + 12 - 2u_1 - 2u_3)\delta = u_2(12+8)\delta \ ; \\ & u_2(u_1p_1 + u_2p_2 + u_3p_3) = u_2(p_1 + p_2 + p_3) \ ; \\ & u_2(1-u_2) = 0. \end{aligned}$$

Donc, en groupant les termes par colonne, on obtient :

$$\begin{array}{rcll} T(C_\rho) & \geq & 12(\rho-2)\delta + 8(\rho-2)\epsilon & \\ & & +(1-u_2) & [(12-2u_1)\delta + 8\epsilon \\ & & & +u_1 p_1 + u_1 \max(0, p_3 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon) \\ & & & +(12-2u_3)\delta + 8\epsilon \\ & & & +u_3 p_3 + u_3 \max(0, p_1 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)] \\ & & +u_2 & [12\delta + 8\epsilon + 8\delta + 8\epsilon + p_1 + p_2 + p_3]. \end{array}$$

Comme $p_3 - 4\delta - 2\epsilon \geq p_3 - 6\delta - 4\epsilon - p_1$ et $p_1 - 4\delta - 2\epsilon \geq p_1 - 6\delta - 4\epsilon - p_3$, on obtient finalement (2.2) :

$$\begin{array}{rlll} T(C_\rho) \geq & (\rho-2) & T(\pi_3) & \\ & +(1-u_2) & [u_1 T(\pi_1) + u_3 T(\pi_2) + (2-u_1-u_3)T(\pi_3)] & \\ & +u_2 & [T(\pi_3) + T(\pi_0)]. & \end{array}$$

□

On démontre ensuite que la propriété (2.2) implique (2.3).

Démonstration Posons $T = \min[T(\pi_0), T(\pi_1), T(\pi_2), T(\pi_3)]$. La propriété (2.2) implique que $\rho = 1$, c'est--dire $T(C_\rho) = T$ ou que

$$T(C_\rho) \geq (\rho-2)T + (1-u_2)[u_1 T + u_3 T + (2-u_1-u_3)T] + u_2[T+T]$$

ce qui conduit

$$T(C_\rho) \geq \rho \times T.$$

Donc, C_ρ est domin par l'une des quatre permutations pyramidales. La minimalité de ρ implique que C_ρ est un 1-cycle, c'est--dire $\rho = 1$. □

Afin de terminer la preuve du Théorème 5, il reste établir la propriété (2.1).

Lemme 3 *Si les inégalités (3.2) et (3.3) sont vérifiées, alors la propriété (2.1) est valide.*

Démonstration Si $u_2 = 1$, alors l'inégalité dans (2.1) se réduit $W_1(C_\rho) + W_2(C_\rho) + W_3(C_\rho) \geq u_1 p_1 + u_2 p_2 + u_3 p_3$. Or cette inégalité est toujours valide.

On peut donc poser que $u_2 = 0$. Ceci signifie que C_ρ ne contient pas la séquence (v_4, v_2, v_1). Nous devons prouver que $\rho = 1$ ou que

$$\begin{array}{rcl} W_1(C_\rho) + W_2(C_\rho) + W_3(C_\rho) & \geq & u_1[p_1 + \max(0, p_3 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)] \\ & & +u_3[p_3 + \max(0, p_1 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)]. \end{array}$$

Nous prouvons d'abord que $u_3 = 1$ implique que C_ρ contient la séquence $A_1 A_0 A_2 A_1$ ou que $\rho = 1$.

Si $u_3 = 1$, alors C_ρ contient au moins une fois l'arc (v_4, v_2) ce qui implique que C_ρ contient la séquence d'activités $A_1 A_0 A_2$. De plus, on sait que C_ρ ne contient

pas deux fois l'un des arcs de E_3. Donc, soit $C_\rho = (A_0, A_2, A_3, A_1)$ et $\rho = 1$, soit l'activité A_1 suit la séquence $A_1A_0A_2$. Donc C_ρ contient la séquence $A_1A_0A_2A_1$. La Figure 3.3 montre que, pendant l'exécution de la séquence $A_1A_0A_2A_1$, le robot doit attendre au moins $\max(0, p_1 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)$ unités de temps.

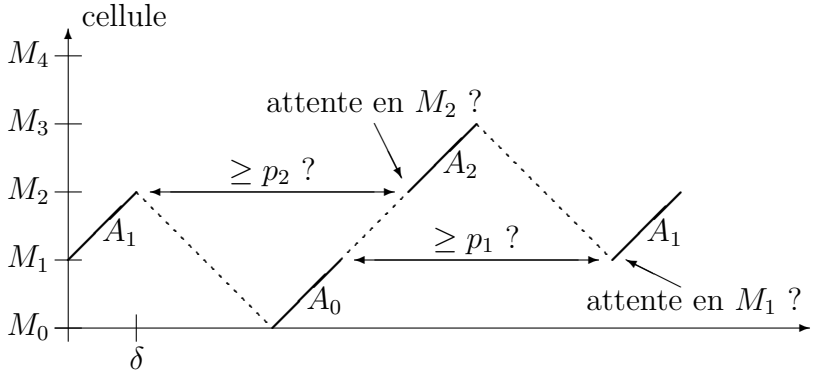

Figure 3.3: La séquence $A_1A_0A_2A_1$

De la même manière, on peut prouver que $u_1 = 1$ implique que $\rho = 1$ ou que C_ρ contient la séquence $A_2A_1A_3A_2$. D'après l'inégalité (3.2) avec $u_2 = 0$ et $u_1 = 1$, on a $p_1 < 2\delta + u_3(2\delta - p_3) \leq 4\delta$. Il n'y a donc pas d'attente en M_1 lors de l'exécution de la séquence $A_2A_1A_3A_2$. La Figure 3.4 indique que le robot doit attendre au moins $\max(0, p_3 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)$ unités de temps lors de l'exécution de cette séquence.

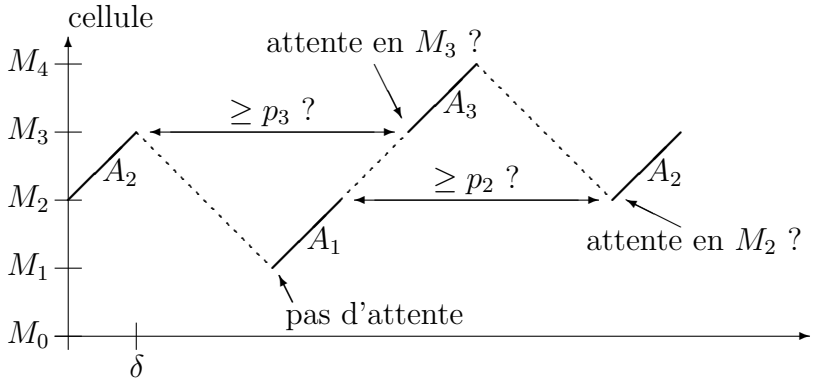

Figure 3.4: La séquence $A_2A_1A_3A_2$

Nous allons maintenant étudier les quatre cas suivants qui correspondent aux différentes valeurs possibles de u_1 et u_3 : $u_1 = u_3 = 0$; $u_1 = 0$ et $u_3 = 1$; $u_1 = 1$ et $u_3 = 0$; $u_1 = u_3 = 1$.

Cas 1 $u_1 = u_3 = 0$
Dans ce cas, la propriété (2.1) se réduit à $\rho = 1$ ou $W_1(C_\rho) + W_2(C_\rho) + W_3(C_\rho) \geq 0$ ce qui est toujours vrai.

Cas 2 $u_1 = 0$ et $u_3 = 1$
Dans ce cas, $C_\rho = (A_0, A_2, A_3, A_1)$ et $\rho = 1$, ou C_ρ contient les séquences A_2A_3 et $A_1A_0A_2A_1$. D'après la discussion précédente, on obtient $W_1(C_\rho) + W_2(C_\rho) + W_3(C_\rho) \geq p_3 + \max(0, p_1 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)$ ce qui conduit la propriété (2.1).

Cas 3 $u_1 = 1$ et $u_3 = 0$
Ce cas est similaire au cas 2. En utilisant les séquences A_0A_1 et $A_2A_1A_3A_2$, on démontre que $\rho = 1$ ou que $W_1(C_\rho) + W_2(C_\rho) + W_3(C_\rho) \geq p_1 + \max(0, p_3 - 4\delta - 2\epsilon, p_2 - 4\delta - 2\epsilon)$.

Cas 4 $u_1 = u_3 = 1$
Dans ce cas, l'inégalité (3.2) devient $p_1 + p_3 < 4\delta$. Donc la propriété (2.1) donne $\rho = 1$ ou $W_1(C_\rho) + W_2(C_\rho) + W_3(C_\rho) \geq p_1 + 2\max(0, p_2 - 4\delta - 2\epsilon) + p_3$. On a $W_1(C_\rho) + W_3(C_\rho) \geq p_1 + p_3$. De plus, on sait que C_ρ contient les séquences $A_1A_3A_2$ et $A_1A_0A_2$. Sous la condition que $p_1 + p_3 < 4\delta$, chacune de ces séquences induit un temps d'attente de $\max(0, p_2 - 4\delta - 2\epsilon)$ en M_2. Ces attentes sont indépendantes, c'est--dire additives. On obtient

$$W_2(C_\rho) \geq 2\max(0, p_2 - 4\delta - 2\epsilon).$$

Ceci conclut la preuve. □

3.1.3 Remarques

Notons que pour des cellules à trois machines, nous aurions pu prouver la conjecture des 1-cycles en utilisant les line-graphs et la théorie des plus court circuits moyens (voir Section 2.4.3, page 51).

Nous avons tenté de poursuivre l'étude des cellules robotisées à l'aide des graphes d'état. Mais G_4 [Figure 3.5] s'avère beaucoup plus complexe à analyser que G_3.

Dans une cellule comportant jusqu'à trois machines, l'analyse des graphes d'état à l'aide du théorème d'additivité donne des résultats exploitables. Toutefois, ce théorème peut difficilement être utilisé pour des cellules de quatre machines et plus puisque l'importance relative, R_m, de E_m diminue rapidement. En effet, dans le graphe $G_m = (V, E)$, on a,

$$R_m = \frac{card(E_m)}{card(E)} = \frac{m+1}{2^{m-2}(m+3)}.$$

Le Tableau 3.1 indique que R_m tends rapidement vers zéro. Ainsi, il semble que pour $m \geq 4$, le théorème d'additivité et les graphes d'état ne soient plus assez puissants pour l'analyse de la Conjecture des 1-cycles.

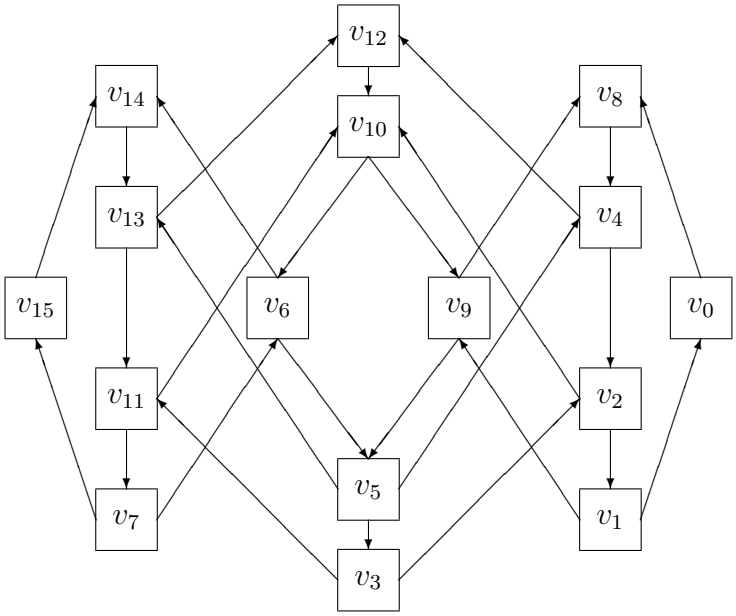

Figure 3.5: Le graphe d'état G_4

Tableau 3.1: Quelques valeurs arrondies de R_m

m	2	3	4	5	6	7
R_m	0,6	0,333	0,178	0,094	0,049	0,025

3.2 Approche algébrique

Nous présentons une nouvelle approche de la dominance des permutations pyramidales et de la Conjecture des 1-cycles. Cette approche permet de prouver simplement, et en utilisant un même cadre, certains résultats connus de la littérature. Elle permet également de trouver de nouveaux résultats. [2]

Dans la section suivante, une formulation algébrique du problème est décrite (Section 3.2.1). Cette formulation fournit un nouvel accès unifié aux principaux résultats connus sur l'ordonnancement de cellules robotisées. Ces résultats concernent la dominance des permutations pyramidales sur les 1-cycles (Section 3.2.2) et la validité de la Conjecture des 1-cycles pour des cellules à deux machines (Section 3.2.3) puis à trois machines (Section 3.2.4).

[2]Les résultats présentés dans les Sections 3.2.1 et 3.3 ont été soumis à la revue *Mathematical and Computer Modelling* [16].

3.2.1 Couvertures pyramidales

Nous présentons d'abord le cadre de l'approche algébrique. L'idée générale est de décomposer le temps de cycle d'un k-cycle C_k en trois composantes : le temps de chargement/déchargement $T_L(C_k)$, le temps de transfert $T_T(C_k)$ et le temps d'attente $T_W(C_k)$. Rappelons que les 2^{m-1} permutations pyramidales sont numérotées de 0 à $\mu = 2^{m-1} - 1$.

L'idée de cette approche est de trouver une combinaison linéaire convexe des permutations pyramidales qui couvre les temps de trajet et les temps d'attente. Ainsi, nous cherchons un ensemble de k permutations pyramidales, meilleures que C_k pour le temps de trajet, puis nous déterminons si cet ensemble domine également C_k pour le temps d'attente. Cet ensemble est décrit à l'aide d'un vecteur x de taille $\mu + 1$. Chaque composante x_j donne le nombre de fois où la permutation π_j apparaît dans l'ensemble.

Définition 3 *Soit C_k, un k-cycle. Soit x un vecteur à coefficients entiers non négatifs, x_j avec $\sum_{j=0}^{\mu} x_j = k$.*

- *Le vecteur x est une couverture pyramidale de C_k si*

$$T(C_k) \geq \sum_{j=0}^{\mu} x_j T(\pi_j) \tag{3.4}$$

- *Le vecteur x est une T-couverture pyramidale de C_k si*

$$T_T(C_k) \geq \sum_{j=0}^{\mu} x_j T_T(\pi_j) \tag{3.5}$$

- *Le vecteur x est une W-couverture pyramidale de C_k si*

$$T_W(C_k) \geq \sum_{j=0}^{\mu} x_j T_W(\pi_j) \tag{3.6}$$

Pour tout vecteur x tel que $\sum x_j = k$, les temps de chargement/déchargement $T_L(C_k)$ et $T_L(\pi_j) = \sum_{h=0}^{m} \epsilon_h^u + \sum_{h=1}^{m+1} \epsilon_h^l$ vérifient l'égalité suivante :

$$T_L(C_k) = k \sum_{h=0}^{m} \epsilon_h^u + k \sum_{h=1}^{m+1} \epsilon_h^l = \sum_{j=0}^{\mu} x_j \left[\sum_{h=0}^{m} \epsilon_h^u + \sum_{h=1}^{m+1} \epsilon_h^l \right] = \sum_{j=0}^{\mu} x_j T_L(\pi_j) \tag{3.7}$$

Lemme 4 *Soit C_k, un k-cycle. Si x est une T-couverture pyramidale et une W-couverture pyramidale de C_k, alors x est une couverture pyramidale de C_k.*

Démonstration Soit x une T-couverture pyramidale et une W-couverture pyramidale de C_k. On obtient l'inégalité (3.4) à partir des inégalités (3.5), (3.6) et (3.7)

comme suit :

$$\begin{aligned} T(C_k) &= T_T(C_k) + T_W(C_k) + T_L(C_k) \\ &\geq \sum_{j=0}^{\mu} x_j T_T(\pi_j) + \sum_{j=0}^{\mu} x_j T_W(\pi_j) + \sum_{j=0}^{\mu} x_j T_L(\pi_j) \\ &\geq \sum_{j=0}^{\mu} x_j T(\pi_j). \end{aligned}$$

□

Lemme 5 *Soit C_k un k-cycle. S'il existe une couverture pyramidale de C_k, alors C_k est dominé par l'ensemble des permutations pyramidales.*

Démonstration Soient I une instance et x une couverture pyramidale de C_k. Notons π_h la permutation pyramidale dont le temps de cycle est minimal pour l'instance I :

$$T(\pi_h) \leq T(\pi_j) \quad \text{pour tout } j \in \{0, 1 \dots \mu\}.$$

L'inégalité (3.4) implique que

$$T(C_k) \geq \sum_{j=0}^{\mu} x_j T(\pi_h) \geq kT(\pi_h).$$

Par conséquent, pour toute instance, il existe une permutation pyramidale qui domine C_k. □

Rappelons que $m_h(C_k)$ est le nombre de fois où le robot voyage entre M_h et M_{h+1}. Nous désirons définir, pour C_k, un ensemble (décrit par un vecteur x) de k permutations pyramidales pour lesquelles $m_h(C_k)$ est égal à la somme des $x_j m_h(\pi_j)$. Comme cette somme est inférieure à $4k$, nous posons $m'_h(C_k) = \min(m_h(C_k), 4k)$. Soit $S_m(C_k)$ le système suivant

$$S_m(C_k) \begin{cases} \displaystyle\sum_{j=0}^{\mu} x_j m_h(\pi_j) = m'_h(C_k) & h = 0, 1 \dots m \\ \displaystyle\sum_{j=0}^{\mu} x_j = k & \\ x_j \text{ entier ;} \quad x_j \geq 0 \text{ ;} & j = 0, 1 \dots \mu \end{cases} \tag{3.8}$$

où le vecteur x est la variable. Le système $S_m(C_k)$ contient des équations redondantes. En effet, l'équation (2.1), page 31, indique que $m'_0(C_k) = m_0(C_k) = 2k$ et que $m_0(\pi_j) = 2$ pour toute permutation pyramidale π_j. Donc, l'équation (3.8) est toujours vraie pour $h = 0$ et pour $h = m$ (équation (2.2), page 31). La Proposition 2 (page 31) indique que $m_1(C_k) = m'_1(C_k) = 4k - 2u_1(C_k)$. Par conséquent, l'équation (3.8) pour $h = 1$ est équivalente à

$$u_1(C_k) = \sum_{j=0}^{\mu} x_j u_1(\pi_j).$$

De même $m_{m-1}(C_k) = m'_{m-1}(C_k) = 4k-2u_m(C_k)$ (Proposition 2, page 31) implique que l'équation (3.8) pour $h = m - 1$ devient

$$u_m(C_k) = \sum_{j=0}^{\mu} x_j u_m(\pi_j).$$

Lemme 6 *Toute solution de $S_m(C_k)$ est une T-couverture pyramidale de C_k.*

Démonstration Soit x une solution de $S_m(C_k)$. L'inégalité (3.5) est déduite de l'équation (3.8) de la manière suivante : multiplier chaque équation de (3.8) par la durée de transfert correspondante δ_h et remplacer $m'_h(C_k)$ par $m_h(C_k)$ pour obtenir

$$m_h(C_k)\delta_h \geq \sum_{j=0}^{\mu} x_j m_h(\pi_j)\delta_h \qquad (h = 0, 1 \ldots m).$$

En additionnant ces inégalités, on obtient

$$T_T(C_k) = \sum_{h=0}^{m} m_h(C_k)\delta_h \geq \sum_{h=0}^{m}\sum_{j=0}^{\mu} x_j m_h(\pi_j)\delta_h$$

ce qui implique

$$T_T(C_k) \geq \sum_{j=0}^{\mu} x_j \sum_{h=0}^{m} m_h(\pi_j)\delta_h = \sum_{j=0}^{\mu} x_j T_T(\pi_j).$$

□

La proposition suivante présente des conditions sur la dominance des permutations pyramidales.

Proposition 9 *Soit C_k un k-cycle. Chacune des conditions suivantes est suffisante pour garantir que C_k est dominé par une permutation pyramidale.*

(C1) *Il existe une permutation pyramidale π_j qui vérifie $T(C_k) \geq kT(\pi_j)$.*

(C2) *Il existe une permutation pyramidale π_j qui vérifie $T(\pi_j) = \Delta_h + p_h$ pour un $h \in \{1, 2 \ldots m\}$.*

(C3) *Il existe une solution x de $S_m(C_k)$ qui est une couverture pyramidale de C_k.*

(C4) *Il existe une solution x de $S_m(C_k)$ qui est une W-couverture pyramidale de C_k.*

Démonstration Si $T(C_k) \geq kT(\pi_j)$ (Condition (C1)) alors la Proposition 9 est vraie par définition. Le Théorème 2 (page 39) indique que si (C2) est valide, alors la permutation pyramidale π_j est optimale. Pour (C3), la Proposition 9 est

une conséquence immédiate du Lemme 5 et (C4) peut être déduite aisément du Lemme 4 et du Lemme 6. □

En utilisant la Proposition 9, nous prouvons que les permutations pyramidales dominent les 1-cycles et que la Conjecture des 1-cycles est valide pour des cellules robotisées à deux et trois machines.

3.2.2 Dominance des permutations pyramidales

Dans [27], Crama et van de Klundert prouvent que l'ensemble des permutations pyramidales domine l'ensemble des 1-cycles. Ils transforment pas à pas une permutation non pyramidale en une permutation pyramidale et prouvent que chaque transformation n'augmente pas le temps de cycle. Avec notre définition du temps de cycle, la stabilité des permutations pyramidales est requise pour la validité de la preuve. Nous établissons la dominance des permutations pyramidales sur les 1-cycles en utilisant l'approche algébrique présentée dans la section précédente.

Soit C un 1-cycle. Le système $S_m(C)$ est donné par

$$\begin{cases} \sum_{j=0}^{\mu} x_j m_h(\pi_j) = m'_h(C) & h = 0, 1 \dots m \\ \sum_{j=0}^{\mu} x_j = 1 & \\ x_j \text{ entier ;} \quad x_j \geq 0 ; & j = 0, 1 \dots \mu \end{cases}$$

Il existe une bijection entre l'ensemble des permutations pyramidales et l'ensemble des vecteurs $V = (v_h)_{h=1,2\dots m-1}$ tels que $v_h \in \{2, 4\}$. En effet, soit π une permutation pyramidale. Dans π, l'activité A_h est montante si et seulement si $m_h(\pi) = 2$ et A_h est descendante si et seulement si $m_h(\pi) = 4$ (A_0 et A_m sont toujours montantes). Pour tout 1-cycle C, on définit donc, de manière unique, la permutation pyramidale π^C comme suit,

$$m_h(\pi^C) = m'_h(C) = \min(m_h(C), 4) \text{ pour tout } h = 0, 1 \dots m.$$

Le 1-cycle π^C est *la permutation pyramidale associée* à C.

Théorème 6 *[27] Supposons que les permutations pyramidales soient stables pour les cellules robotisées à m machines. Alors, les permutations pyramidales dominent les 1-cycles.*

Démonstration Supposons que toutes les permutations pyramidales soient stables. Nous prouvons, en utilisant la Proposition 9, que tout 1-cycle C est dominé par sa permutation pyramidale associée π^C, c'est-à-dire

$$T(C) \geq T(\pi^C). \tag{3.9}$$

La permutations pyramidales π^C associée à C définit une solution x de $S_m(C)$ donnée par

$$x_j = 1 \quad \text{si } \pi_j = \pi^C \quad \text{et } x_j = 0 \quad \text{sinon.}$$

Nous présentons une preuve par induction sur le nombre m de machines dans la cellule robotisée. Pour $m = 2$, tous les 1-cycles sont des permutations pyramidales et l'inégalité (3.9) est satisfaite. Supposons que l'inégalité (3.9) soit vérifiée pour tout 1-cycle dans une cellule à q machines où $q < m$. Nous démontrons que l'inégalité (3.9) reste valide pour une cellule à m machines.

Considérons une cellule robotisée à m machines et un 1-cycle C. Il existe plusieurs cas pour lesquels on peut prouver directement la validité de l'inégalité (3.9) sans utiliser l'hypothèse d'induction. En fait, cette hypothèse n'est requise que dans le dernier cas considéré (Cas 3).

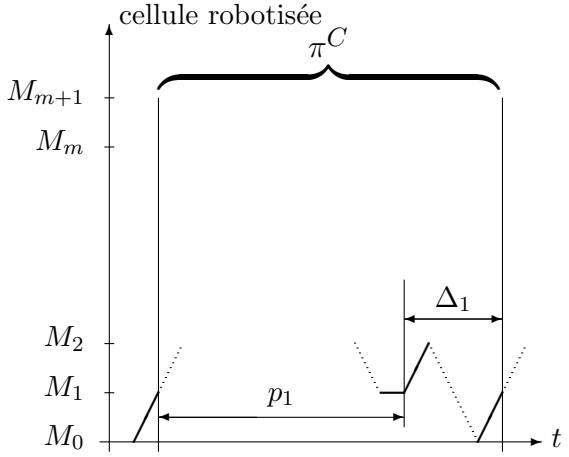

Figure 3.6: A_1 est descendante dans π^C et le robot attend en M_1

Considérons d'abord les hypothèses suivantes pour π^C [Figure 2.4, page 39] :

- Si A_1 est descendante et si le robot attend en M_1 [Figure 3.6] alors, comme π^C est supposée stable, $T(\pi^C)$ est égal au temps qui s'écoule entre deux instants consécutifs où M_1 commence à usiner une pièce. Ainsi $T(\pi^C) = p_1 + \Delta_1$ et π^C est optimal (la Condition (C2) de la Proposition 9 est satisfaite) ;
- Si A_{m-1} est descendante et si le robot attend à la machine M_m mais n'attend pas à la machine M_{m-1} alors, de la même façon, on peut démontrer que $T(\pi^C) = p_m + \Delta_m$;
- Si A_{h-1} et A_h sont descendantes pour $h \in \{2, 3 \dots m-1\}$ et si le robot attend à la machine M_h mais n'attend pas à la machine M_{h-1} alors, de la même manière on obtient $T(\pi^C) = p_h + \Delta_h$.

Considérons les cas qui restent. S'il n'y a pas d'attente aux machines pendant l'exécution de π^C, alors $T_W(\pi^C) = 0 \leq T_W(C)$. Donc l'équation (3.6) est vérifiée et (C4) est vraie. Considérons maintenant le cas où le robot attend à une machine.

Soit τ le plus petit indice tel que le robot attend à la machine M_τ lors de l'exécution de π^C.

Cas 1 $\tau = m$.

Dans ce cas, A_{m-1} est montante dans π^C ce qui implique que π^C contient la sous-séquence $A_{m-1}A_m$. Comme il n'y a pas de temps d'attente aux machines $M_1, M_2 \dots M_{m-1}$, on a $T_W(\pi^C) = p_m$. Comme A_{m-1} est montante dans π^C, on a $m_{m-1}(\pi^C) = 2 = m_{m-1}(C)$. La Proposition 2 (page 31) indique que $m_{m-1}(C) = 4 - 2u_m(C)$ d'où $u_m(C) = 1$. Donc, C contient la séquence $A_{m-1}A_m$. Ceci implique que $T_W(C) \geq p_m = T_W(\pi^C)$ et (C4) est vraie.

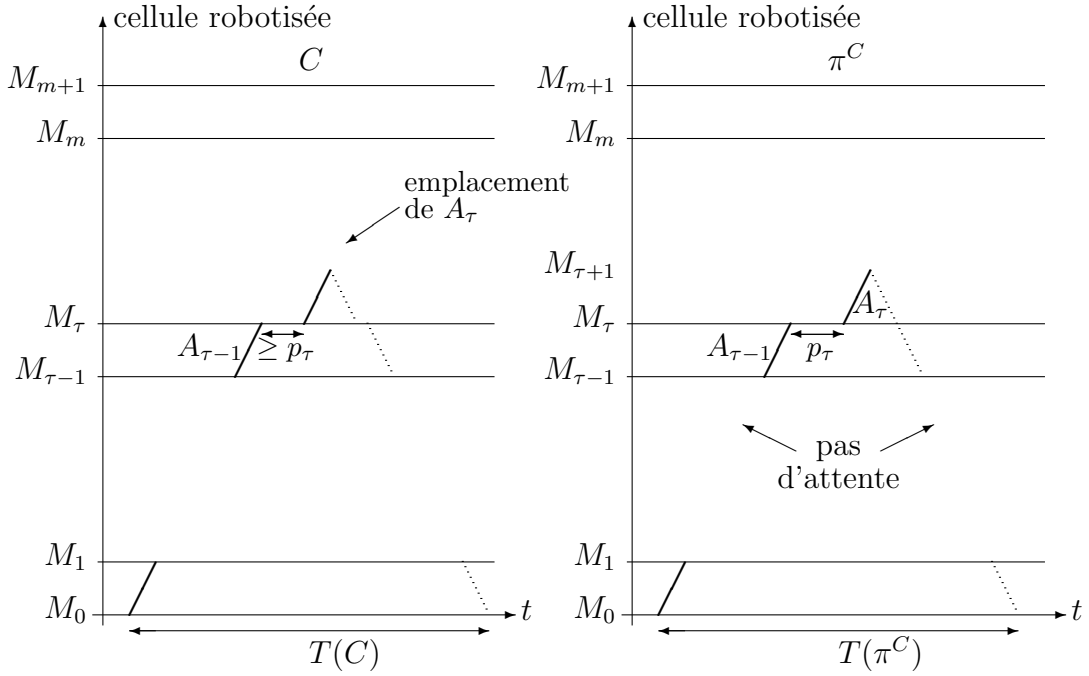

Figure 3.7: Illustration du Cas 2

Cas 2 $1 < \tau < m$ et A_τ est descendante. Par conséquent $A_{\tau-1}$ est montante dans π^C.

Comme $A_{\tau-1}$ est montante dans π^C (c'est-à-dire $m_{\tau-1}(C) = m_{\tau-1}(\pi^C) = 2$) on remarque que [Figure 3.7] :

- dans C, toutes les activités entre $A_{\tau-1}$ et A_τ ont un indice supérieur à τ ;
- dans π^C, toutes les activités dont l'indice est supérieur à τ sont entre $A_{\tau-1}$ et A_τ.

On a donc [Figure 3.7] :

$$T(\pi^C) = p_\tau + 2\delta_\tau + 2\delta_{\tau-1} + \epsilon^u_{\tau-1} + \epsilon^l_\tau + \epsilon^u_\tau + \epsilon^l_{\tau+1} + \sum_{h=0}^{\tau-2} m_h(\pi^C)\delta_h + \sum_{h=0}^{\tau-2}(\epsilon^u_h + \epsilon^l_{h+1}).$$

Par construction, $m_h(C) \geq m_h(\pi^C)$ pour $h = 0, 1 \ldots m$. Donc, chaque temps d'exécution de C est plus grand que $T(\pi^C)$:

$$T(C) \geq p_\tau + 2\delta_\tau + 2\delta_{\tau-1} + \epsilon^u_{\tau-1} + \epsilon^l_\tau + \epsilon^u_\tau + \epsilon^l_{\tau+1} + \sum_{h=0}^{\tau-2} m_h(C)\delta_h + \sum_{h=0}^{\tau-2}(\epsilon^u_h + \epsilon^l_{h+1}) \geq T(\pi^C).$$

Par conséquent, la condition (C3) est vraie.

Cas 3 $1 \leq \tau < m$ et A_τ est montante dans π^C.

L'hypothèse d'induction n'est utilisée explicitement que pour ce cas. Une illustration de l'induction et des notations suivantes est donnée dans la Figure 3.8.

Comme A_τ est montante dans π^C, on a $m_\tau(C) = m_\tau(\pi^C) = 2$. Donc, pendant l'exécution du cycle C, le robot ne va qu'une fois de M_τ à $M_{\tau+1}$, à savoir quand il exécute A_τ. Soit t_τ l'instant de début de l'activité A_τ dans C, lorsque le robot prend une pièce de la machine M_τ. Le robot se déplace aussi une seule fois de $M_{\tau+1}$ à M_τ. Notons t'_τ l'instant où le robot passe en M_τ en venant de $M_{\tau+1}$. De la même manière, nous définissons les temps s_τ et s'_τ pour le cycle π^C.

Soient

$$\begin{aligned} T &= T(C) - (t'_\tau - t_\tau) \qquad \text{et} \\ T^* &= T(\pi^C) - (s'_\tau - s_\tau). \end{aligned}$$

Nous calculons maintenant la valeur de T^* et une borne inférieure de T afin de prouver que $T^* \leq T$. Le robot attend à la machine M_τ pendant l'exécution de π^C. Par conséquent, la durée entre la fin de $A_{\tau-1}$ et le début de A_τ est p_τ. Donc, si $A_{\tau-1}$ est descendante dans π^C (dans ce cas, on a $\tau \neq 1$), alors [Figure 3.8]

$$T^* = p_\tau + 2\delta_{\tau-1} + \epsilon^u_{\tau-1} + \epsilon^l_\tau$$

et

$$T \geq p_\tau + 2\delta_{\tau-1} + \epsilon^u_{\tau-1} + \epsilon^l_\tau.$$

Si $A_{\tau-1}$ est montante dans π^C [Figure 3.9] alors, comme le robot n'attend pas aux machines M_h pour $h < \tau$, T^* vérifie

$$T^* = p_\tau + \sum_{h=0}^{\tau-1} m_h(\pi^C)\delta_h + \sum_{h=0}^{\tau-1}(\epsilon^u_h + \epsilon^l_{h+1}).$$

Les activités $A_{\tau-1}$ et A_τ sont montantes dans π^C. Par conséquent $m_{\tau-1}(C) = m_{\tau-1}(\pi^C) = 2$ et $m_\tau(C) = m_\tau(\pi^C) = 2$. Dans ce cas, il n'y a pas d'activité entre $A_{\tau-1}$ et A_τ dans C. Donc C contient la séquence $A_{\tau-1}A_\tau$ et $m_h(C) \geq m_h(\pi^C)$ et

$$T \geq p_\tau + \sum_{h=0}^{\tau-1} m_h(C)\delta_h + \sum_{h=0}^{\tau-1}(\epsilon^u_h + \epsilon^l_{h+1}) \geq T^*.$$

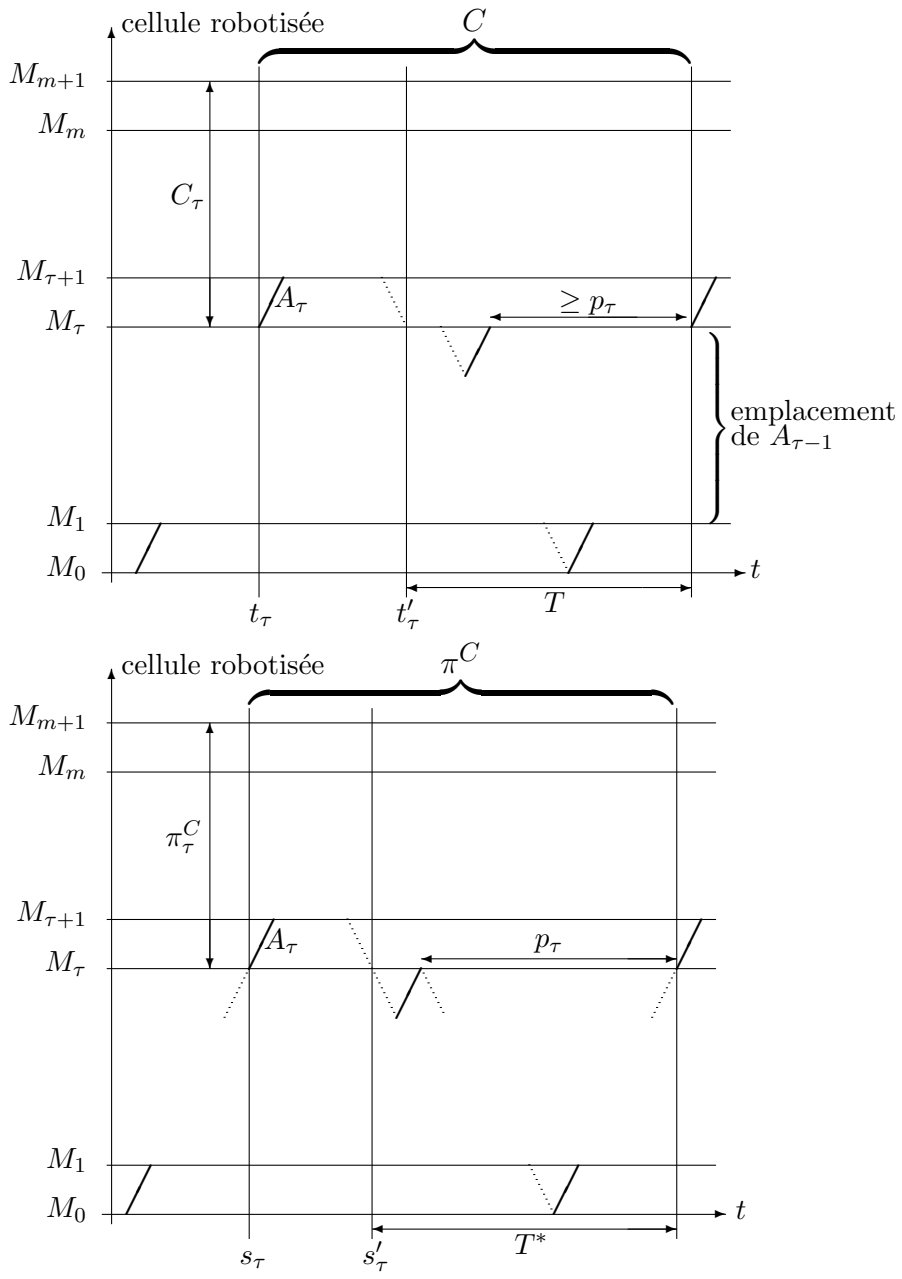

Figure 3.8: Illustration de l'étape d'induction si $A_{\tau-1}$ est descendante dans π^C

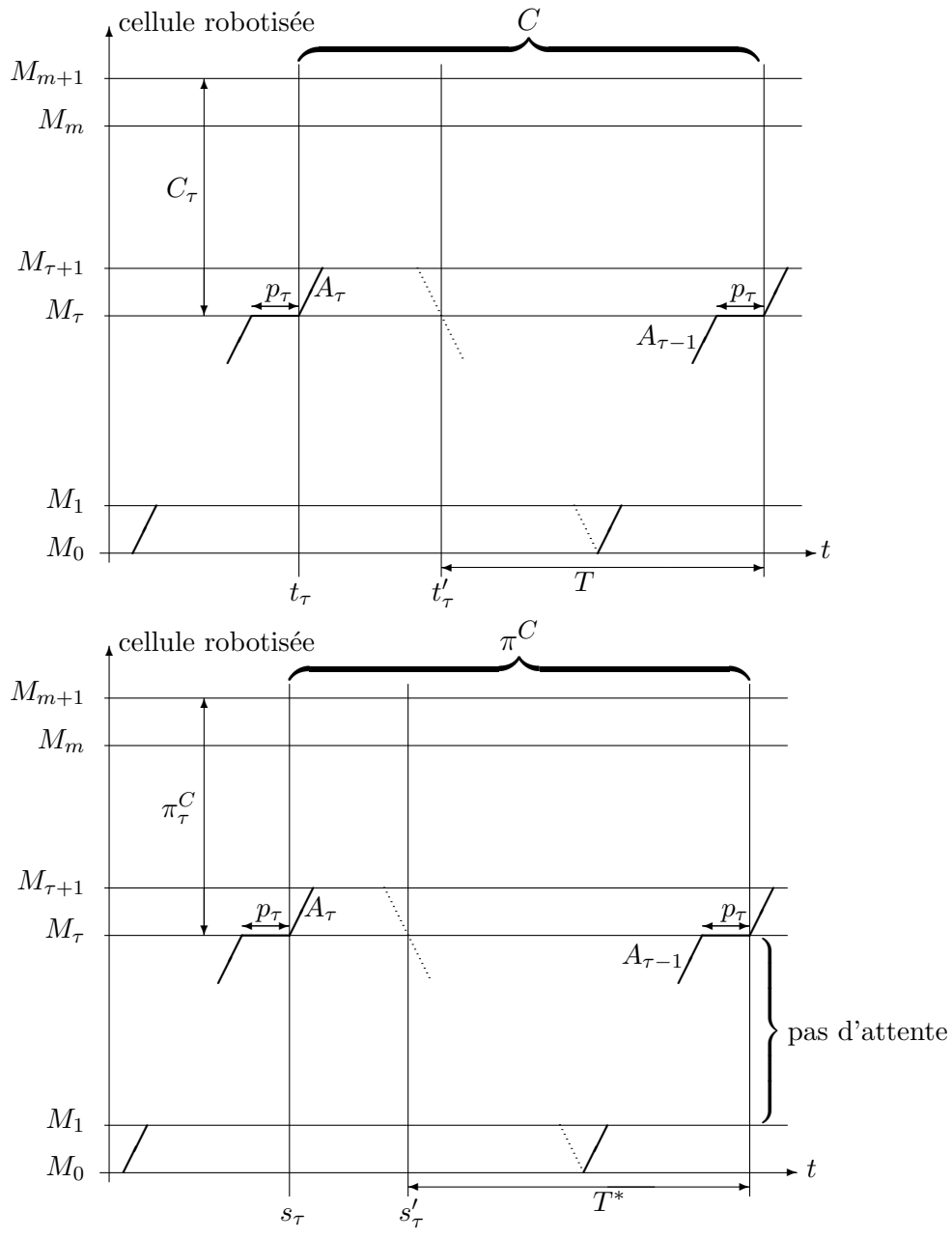

Figure 3.9: Illustration de l'étape d'induction si $A_{\tau-1}$ est montante dans π^C

Nous avons prouvé que $T^* \leq T$. Soit S_τ la cellule robotisée réduite, composée des $m - \tau$ machines $M_\tau, M_{\tau+1} \ldots M_{m+1}$ où M_τ correspond à la station de chargement et M_{m+1} correspond à la station de déchargement. Considérons les cycles réduits, π_τ^C de π^C et C_τ de C dans S_τ. Par construction, π_τ^C est la permutation pyramidale associée à C_τ dans S_τ. Pour la cellule robotisée S_τ, nous conservons les données précédentes $\epsilon_h^u, \epsilon_h^l, \delta_h, p_h$ excepté ϵ_τ^u qui est redéfini par $\epsilon_\tau^u(S_\tau) = T$. Nous pouvons maintenant appliquer l'hypothèse d'induction pour obtenir

$$T(C_\tau) \geq T(\pi_\tau^C).$$

Finalement, nous devons établir un lien entre les temps de cycle $T(C_\tau)$ et $T(\pi_\tau^C)$ et les temps de cycle $T(C)$ et $T(\pi^C)$. Supposons que C et C_τ soient exécutés en parallèle, en commençant avec le robot chargé en M_τ. Le robot exécute la même activité puis voyage non chargé de $M_{\tau+1}$ à M_τ au même moment. Le cycle C se termine après T unités de temps et le robot est retourné à sa position initiale. Dans la cellule S_τ, le robot peut charger sans délai une pièce en T unités de temps. Par conséquent, les deux cycles se terminent au même moment, c'est-à-dire $T(C_\tau) = T(C)$. En utilisant, en plus, le fait que $T \geq T^*$, on montre de la même manière que $T(\pi_\tau^C) \geq T(\pi^C)$. Donc, $T(\pi^C) \leq T(C)$ et la condition (C3) de la Proposition 9 est vraie. □

3.2.3 Cellule à deux machines

Dans une cellule à deux machines, les deux permutations pyramidales et les composants de leurs temps de cycles sont donnés dans le Tableau 3.2. Les calculs des temps de cycle des permutations pyramidales pour $m = 2$ sont décrits dans l'Annexe B.

Tableau 3.2: Temps de cycle des permutations pyramidales pour $m = 2$

j	π_j	$T_T(\pi_j)$	$T_W(\pi_j)$
0	$(A_0A_1A_2)$	$2\sum_{h=0}^{2} \delta_h$	$p_1 + p_2$
1	$(A_0A_2A_1)$	$2\sum_{h=0}^{2} \delta_h + 2\delta_1$	$max(0, p_1 - 2\delta_1 - 2\delta_2 - \epsilon_2^u - \epsilon_3^l,$ $p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l)$

Proposition 10 *La Conjecture des 1-cycles est valide pour une cellule à deux machines.*

Démonstration Nous montrons que la condition (C2) ou que la condition (C4) de la Proposition 9 est vraie pour tout k-cycle C_k. Considérons deux cas :

Tableau 3.3: Temps de cycle des permutations pyramidales pour $m = 3$

j	π_j	$T_T(\pi_j) + T_L(\pi_j)$	$T_W(\pi_j)$
0	$(A_0A_1A_2A_3)$	$\Delta_1 + \Delta_3$	$p_1 + p_2 + p_3$
1	$(A_0A_1A_3A_2)$	$\Delta_1 + \Delta_3$ $+2\delta_2$	$p_1 + max(0, p_3 - \Delta_1 - 2\delta_2 - p_1,$ $p_2 - 2\delta_2 - 2\delta_3 - \epsilon_3^u - \epsilon_4^l)$
2	$(A_0A_2A_3A_1)$	$\Delta_1 + \Delta_3$ $+2\delta_1$	$p_3 + max(0, p_1 - 2\delta_1 - \Delta_3 - p_3,$ $p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l)$
3	$(A_0A_3A_2A_1)$	$\Delta_1 + \Delta_3$ $+2\delta_1 + 2\delta_2$	$max(0, p_1 - 2\delta_1 - 2\delta_2 - \Delta_3,$ $p_2 - 2\delta_1 - 2\delta_2 - \Delta_1 - \Delta_3 + \Delta_2,$ $p_3 - 2\delta_1 - 2\delta_2 - \Delta_1)$

Cas 1 $T_W(\pi_1) > 0$

Dans ce cas, le temps de cycle de π_1 satisfait $T(\pi_1) = \max(p_1 + \Delta_1, p_2 + \Delta_2)$ et (C2) est vraie.

Cas 2 $T_W(\pi_1) = 0$

Dans ce cas, les temps d'attente des 1-cycles vérifient $T_W(\pi_0) = p_1 + p_2$ et $T_W(\pi_1) = 0$. Pour tout k-cycle C_k, le système $S_2(C_k)$ est donné par

$$\begin{cases} u_1(C_k) & = & x_0 \\ u_2(C_k) & = & x_0 \\ x_0 + x_1 & = & k \\ x_j \geq 0 & & \text{pour } j = 0, 1 \\ x_j \text{ entier} & & \text{pour } j = 0, 1. \end{cases}$$

La Proposition 2 (page 31) indique que $m_1(C_k) = 4k - 2u_1(C_k) = m_{m-1}(C_k) = 4k - 2u_2(C_k)$, ce qui implique que $u_1(C_k) = u_2(C_k)$. Le vecteur $x = (u_1(C_k), k - u_1(C_k))$ est une solution de $S_2(C_k)$ et satisfait

$$\sum_{j=0}^{1} x_j T_W(\pi_j) = u_1(C_k)p_1 + u_2(C_k)p_2 \leq T_W(C_k).$$

Donc, (C4) est vraie. □

3.2.4 Cellule à trois machines

Considérons une cellule à trois machines. Dans cette section, u_h représente $u_h(C_k)$ pour tout $h \in \{1, 2, 3\}$. Les quatre permutations pyramidales sont décrites dans le Tableau 3.3. Les calculs des temps de cycle des permutations pyramidales pour $m = 3$ sont décrits dans l'Annexe B.

Le système $S_3(C_k)$ est donné par

$$\begin{cases} u_1 = x_0 + x_1 & (3.10) \\ u_3 = x_0 + x_2 & (3.11) \\ k = x_0 + x_1 + x_2 + x_3 & (3.12) \\ x_j \text{ entier} \qquad x_j \geq 0 \qquad \text{pour } j = 0, 1, 2, 3. \end{cases}$$

Proposition 11 *La Conjecture des 1-cycles est valide pour une cellule à trois machines.*

Démonstration Considérons une instance et un k-cycle C_k. Nous étudions trois cas.

Cas 1 L'une des trois conditions suivantes est vérifiée :

$$\begin{aligned} T_W(\pi_1) &= p_3 - \Delta_1 - 2\delta_2 \ ; \\ T_W(\pi_2) &= p_1 - 2\delta_1 - \Delta_3 \ ; \\ T_W(\pi_3) &\neq 0. \end{aligned}$$

Dans ce cas, la condition (C2) est toujours vraie. Supposons, par exemple que $T_W(\pi_1) = p_3 - \Delta_1 - 2\delta_2$. Alors, on a $T(\pi_1) = p_3 + \Delta_3$.

Cas 2 Pour tous les autres cas, les conditions suivantes sont vérifiées :

$$\begin{aligned} T_W(\pi_1) &= p_1 + w_2^1 \ ; \\ T_W(\pi_2) &= p_3 + w_2^2 \ ; \\ T_W(\pi_3) &= 0 \end{aligned}$$

où $w_2^1 = \max(0, p_2 - 2\delta_2 - 2\delta_3 - \epsilon_3^u - \epsilon_4^l)$ et $w_2^2 = max(0, p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l)$. Nous prouvons que la condition (C4) est vraie. Soit x le vecteur suivant

$$\begin{aligned} x_0 &= \min_h(u_h) \ ; \\ x_1 &= u_1 - \min_h(u_h) \ ; \\ x_2 &= u_3 - \min_h(u_h) \ ; \\ x_3 &= k - u_1 - u_3 + \min_h(u_h). \end{aligned}$$

Le vecteur x est un vecteur entier qui vérifie les équations (3.10), (3.11) et (3.12) de $S_3(C_k)$. Il est clair que x_0, x_1 et x_2 ont des valeurs non négatives. Nous devons seulement prouver que x_3 est non négatif et que l'équation (3.6) est vérifiée, c'est-à-dire

$$T_W(C_k) \geq x_0(p_1 + p_2 + p_3) + x_1(p_1 + w_2^1) + x_2(p_3 + w_2^2) \qquad (3.13)$$

Dans C_k, considérons la séquence A_0A_1, si elle existe, et le premier A_1 qui précède cette séquence (dans le sens cyclique). D'après la Définition 1 (page 31), il y a exactement une occurrence de A_2 entre deux occurrences de A_1 :

$$A_1 — A_2 \ldots A_0 A_1.$$

Analysons les possibilités pour la séquence A_1—A_2 à gauche : on ne peut avoir ni A_0, ni A_2, car chacune des ces activités ne peut se produire qu'une fois entre deux occurrences de A_1 ; l'activité A_1 est impossible par construction. Ainsi, chacune des u_1 occurrences de A_0A_1 est précédée par une occurrence de A_1A_2 ou par une occurrence de $A_1A_3A_2$ (où les deux A_1 sont consécutifs) ce qui implique que

$$u_1 \leq u_2 + |A_1A_3A_2| \tag{3.14}$$

De la même manière, on peut prouver que chaque séquence A_2A_3 est suivie par A_1A_2 ou par $A_1A_0A_2$ (où les deux A_2 sont consécutifs) ce qui implique que

$$u_3 \leq u_2 + |A_1A_0A_2| \tag{3.15}$$

En ajoutant les équations (3.14) et (3.15), on obtient

$$u_1 + u_3 \leq 2u_2 + |A_1A_3A_2| + |A_1A_0A_2|.$$

L'activité A_2 apparaît k fois dans C_k. Donc $|A_1A_2| + |A_1A_3A_2| + |A_1A_0A_2| \leq k$ et $u_1 + u_3 \leq u_2 + k$. Si $\min_h(u_h) = u_2$, alors l'équation précédente indique que $x_3 \geq 0$. De plus, si $\min_h(u_h)$ est égal à u_1 ou à u_3 alors, comme $u_h \leq k$, on a aussi $x_3 \geq 0$.

Nous prouvons maintenant que l'inégalité (3.13) est vérifiée. Pour cela, nous avons besoin de l'inégalité suivante :

$$w_2^1 + w_2^2 \leq p_2 \tag{3.16}$$

Les inégalités $w_2^1 \leq p_2$ et $w_2^2 \leq p_2$ sont toujours vérifiées. Donc, si $w_2^1 = 0$ ou $w_2^2 = 0$, alors l'inégalité (3.16) est vraie. Sinon, $w_2^1 > 0$ et $w_2^2 > 0$. Si $p_2 > \Delta_1 + \Delta_3 - \Delta_2 + 2\delta_1 + 2\delta_2$ alors $T_W(\pi_3) > 0$ ce qui est en contradiction avec l'hypothèse que $T_W(\pi_3) = 0$. On a donc

$$p_2 \leq 2\delta_0 + 2\delta_1 + 2\delta_2 + 2\delta_3 + \epsilon_0^u + \epsilon_1^l + \epsilon_3^u + \epsilon_4^l.$$

En ajoutant p_2 des deux cotés de l'inégalité, on obtient l'inégalité (3.16) :

$$w_2^1 + w_2^2 = p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l + p_2 - 2\delta_2 - 2\delta_3 - \epsilon_3^u - \epsilon_4^l \leq p_2.$$

La sous-séquence $A_1A_3A_2$ (respectivement $A_1A_0A_2$) implique un temps d'attente d'au moins w_2^1 (respectivement w_2^2) [Figure 3.10]. Donc, $T_W(C_k) \geq u_1p_1 + u_2p_2 + u_3p_3 + |A_1A_3A_2|w_2^1 + |A_1A_0A_2|w_2^2$. L'équation (3.13) est déduite des équations (3.14), (3.15) et (3.16) de la manière suivante :

$$\begin{aligned}
&x_0(p_1 + p_2 + p_3) + x_1(p_1 + w_2^1) + x_2(p_3 + w_2^2) \\
&\quad = \min(u_h)[p_1 + p_2 + p_3] + [u_1 - \min(u_h)]p_1 + [u_3 - \min(u_h)]p_3 \\
&\qquad +[u_1 - \min(u_h)]w_2^1 + [u_3 - \min(u_h)]w_2^2 \\
&\quad \leq u_1p_1 + \min(u_h)p_2 + u_3p_3 \\
&\qquad +[u_2 - \min(u_h) + |A_1A_3A_2|]w_2^1 + [u_2 - \min(u_h) + |A_1A_0A_2|]w_2^2 \\
&\quad \leq u_1p_1 + \min(u_h)p_2 + [u_2 - \min(u_h)]p_2 + u_3p_3 + |A_1A_3A_2|w_2^1 + |A_1A_0A_2|w_2^2 \\
&\quad \leq u_1p_1 + u_2p_2 + u_3p_3 + |A_1A_3A_2|w_2^1 + |A_1A_0A_2|w_2^2 \\
&\quad \leq T_W(C_k).
\end{aligned}$$

Ceci conclut la preuve. □

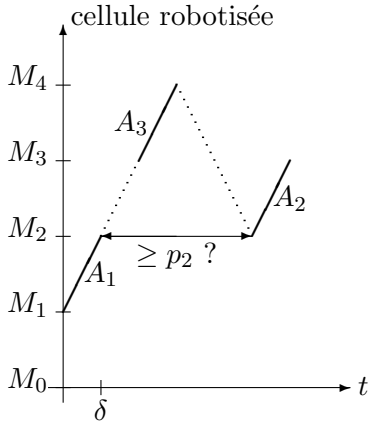

Figure 3.10: La séquence $A_1A_3A_2$

3.3 Limites de la conjecture

Nous considérons maintenant des cellules robotisées dont le nombre de machines, m, est supérieur ou égal à 4.

Proposition 12 *La Conjecture des 1-cycles n'est pas valide pour des cellules robotisées à m machines lorsque $m \geq 4$.*

Nous prouvons cette proposition dans les deux sections suivantes. Nous présentons d'abord un contre-exemple de la plus petite taille possible à la Conjecture des 1-cycles. Puis, nous étendons ce contre-exemple à des cellules robotisées à m machines pour $m \geq 4$.

3.3.1 Contre-exemple minimal

On sait que la Conjecture des 1-cycles est vraie pour des cellules robotisées à deux et trois machines. Par conséquent, un contre-exemple à la conjecture pour $m = 4$ et $k = 2$ est minimal.

Proposition 13 *Dans des cellules robotisées générales, les 1-cycles ne dominent pas les 2-cycles pour $m = 4$.*

Démonstration Considérons l'instance suivante I_4 [Figure 3.11] :

$$
\begin{array}{l}
m = 4 \ ; \\
\epsilon_h^u = \epsilon_h^l = 0 \qquad \text{pour tout } h \ ; \\
p_1 = 6 \ ; \quad p_2 = 10 \ ; \quad p_3 = 0 \ ; \quad p_4 = 0 \ ; \\
\delta_0 = 6 \ ; \quad \delta_1 = 6 \ ; \quad \delta_2 = 1 \ ; \quad \delta_3 = 1 \ ; \quad \delta_4 = 1.
\end{array}
$$

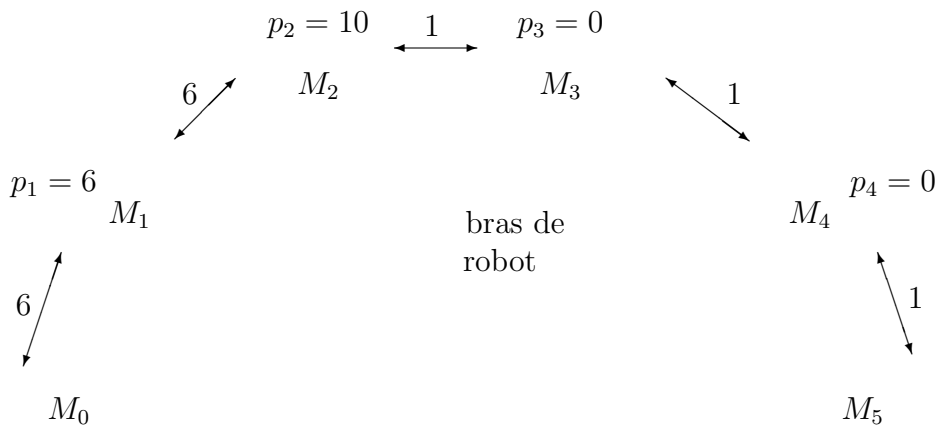

Figure 3.11: L'instance I_4

Soit $C_2 = (A_0A_1A_0A_2A_1A_4A_3A_4A_2A_3)$. Pour l'instance I_4, le 2-cycle C_2 domine les $2^{m-1} = 8$ permutations pyramidales, c'est-à-dire

$$\frac{T(C_2)}{2} < T(\pi_j), \text{ pour toute permutation pyramidale } \pi_j.$$

Considérons d'abord le 2-cycle $C_2 = (A_0A_1A_0A_2A_1A_4A_3A_4A_2A_3)$. Pour cette représentation de C_2, le seul état initial possible est décrit de la manière suivante : les machine M_1, M_2 et M_3 ne contiennent pas de pièce et la machine M_4 contient une pièce prête (le temps d'usinage d'une pièce sur M_4 est 0). Le Tableau 3.4 décrit, pour l'instance I_4, l'exécution de C_2 pas à pas entre deux états identiques de la cellule qui représentent le début et la fin du cycle. La première colonne décrit tous les instants t, un par un. La seconde colonne indique les événements en train de se produire dans la cellule à l'instant t (déplacements du robot, exécution d'activités...). La troisième colonne donne la position du robot (pr) dans la cellule lorsque le robot est à une machine. Les quatre dernières colonnes décrivent l'état de la cellule : elles indiquent les temps d'usinage restants s'il y a une pièce sur la machine M_h à l'instant t. La valeur "-1" signifie qu'il n'y a pas de pièce sur M_h à l'instant t. Après une exécution de C_2, l'état du système est restauré et la pièce sur M_4 est prête. Le 2-cycle C_2 est donc stationnaire et stable pour l'instance I_4 et son temps de cycle est 82 unités de temps. Il reste à montrer que $T(\pi_j) > T(C_2)/2 = 41$ unités de temps.

Les temps de cycle des permutations pyramidales pour l'instance I_4 sont donnés par

$$\begin{array}{ll} T(A_0A_1A_2A_3A_4) = 46\ ; & T(A_0A_1A_2A_4A_3) = 48\ ; \\ T(A_0A_1A_3A_4A_2) = 42\ ; & T(A_0A_1A_4A_3A_2) = 42\ ; \\ T(A_0A_2A_3A_4A_1) = 42\ ; & T(A_0A_2A_4A_3A_1) = 44\ ; \\ T(A_0A_3A_4A_2A_1) = 44\ ; & T(A_0A_4A_3A_2A_1) = 46. \end{array}$$

Pour $m = 4$, il existe donc une instance pour laquelle un 2-cycle domine strictement tous les 1-cycles. □

Si l'on n'autorise que des temps d'usinage non nuls, il suffit de multiplier tous les éléments de I_4 par 10 et de poser que p_3 et p_4 sont égaux à 1 pour obtenir

$$\begin{array}{l} m = 4 \ ; \\ \epsilon_h^u = \epsilon_h^l = 0 \text{ pour tout } h \ ; \\ p_1 = 60 \ ; \quad p_2 = 100 \ ; \quad p_3 = 1 \ ; \quad p_4 = 1 \ ; \\ \delta_0 = 60 \ ; \quad \delta_1 = 60 \ ; \quad \delta_2 = 10 \ ; \quad \delta_3 = 10 \ ; \quad \delta_4 = 10. \end{array}$$

Pour cette instance, $T(C_2)/2 = 411$ et le plus petit temps de cycle des permutations pyramidales est 420.

3.3.2 Extension à m machines

Le contre-exemple précédent peut être étendu à une cellule robotisée à m machines ($m \geq 4$).

Proposition 14 *Dans une cellule robotisée, les 1-cycles ne dominent pas les 2-cycles pour $m \geq 4$.*

Démonstration Nous prouvons que, quelle que soit la valeur de $m \geq 4$, il existe un 2-cycle qui est strictement meilleur que toutes les permutations pyramidales. Considérons l'instance I_m suivante [Figure 3.12] :

$$\begin{array}{lll} m \ ; & \epsilon_h^u = \epsilon_h^l = 0 \quad \forall h \ ; & \\ \delta_0 = 6 \ ; & \delta_1 = 6 \ ; \quad \delta_2 = 1 \ ; \quad \delta_3 = 1 \ ; & \delta_m = 1 \ ; \\ \text{si } m > 4, & \delta_h = 0 & \text{pour } 4 \leq h \leq m-1 \ ; \\ p_1 = 6 \ ; & p_2 = 10 \ ; & p_h = 0 \quad \text{pour } 3 \leq h \leq m. \end{array}$$

Notons U_m la séquence composée de toutes les activités en ordre croissant dont l'indice est plus grand que 4. Par exemple, $U_6 = A_4A_5A_6$. Soit C_2 le 2-cycle $(A_0A_1A_0A_2A_1U_mA_3U_mA_2A_3)$. Pour l'instance I_m, nous prouvons que le 2-cycle C_2 domine strictement les 2^{m-1} permutations pyramidales, c'est-à-dire

$$\frac{T(C_2)}{2} < T(\pi_j), \text{ pour toute permutation pyramidale } \pi_j \quad (j = 0, 1 \ldots \mu).$$

Prouvons d'abord que $T(C_2) = 82$. Pour la même raison que pour le cas de quatre machines, le cycle C_2 est stationnaire et stable pour l'instance I_m. On remarque que $m_0(C_2) = 4$ et $m_h(C_2) = 6$ pour $1 \leq h \leq 3$ et $m_h(C_2) = 4$ pour $4 \leq h \leq m$. Par conséquent, on a

$$T_T(C_2) \ = \ \sum_{h=0}^{m} m_h(C_2)\delta_h$$

Tableau 3.4: Exécution pas à pas de $C_2 = (A_0A_1A_0A_2A_1A_4A_3A_4A_2A_3)$ pour I_4

t		pr	état de la cellule à t			
			temps d'usinage restant à			
			M_1	M_2	M_3	M_4
0	début de A_0, trajet de M_0 à M_1	M_0	-1	-1	-1	0
1	trajet de M_0 à M_1		-1	-1	-1	0
2	trajet de M_0 à M_1		-1	-1	-1	0
3	trajet de M_0 à M_1		-1	-1	-1	0
4	trajet de M_0 à M_1		-1	-1	-1	0
5	trajet de M_0 à M_1		-1	-1	-1	0
6	fin de A_0, attente en M_1	M_1	6	-1	-1	0
7	attente en M_1	M_1	5	-1	-1	0
8	attente en M_1	M_1	4	-1	-1	0
9	attente en M_1	M_1	3	-1	-1	0
10	attente en M_1	M_1	2	-1	-1	0
11	attente en M_1	M_1	1	-1	-1	0
12	début de A_1, trajet de M_1 à M_2	M_1	0	-1	-1	0
13	trajet de M_1 à M_2		-1	-1	-1	0
14	trajet de M_1 à M_2		-1	-1	-1	0
15	trajet de M_1 à M_2		-1	-1	-1	0
16	trajet de M_1 à M_2		-1	-1	-1	0
17	trajet de M_1 à M_2		-1	-1	-1	0
18	fin de A_1, trajet de M_2 à M_0	M_2	-1	10	-1	0
19	trajet de M_2 à M_0		-1	9	-1	0
20	trajet de M_2 à M_0		-1	8	-1	0
21	trajet de M_2 à M_0		-1	7	-1	0
22	trajet de M_2 à M_0		-1	6	-1	0
23	trajet de M_2 à M_0		-1	5	-1	0
24	trajet de M_2 à M_0	M_1	-1	4	-1	0
25	trajet de M_2 à M_0		-1	3	-1	0
26	trajet de M_2 à M_0		-1	2	-1	0
27	trajet de M_2 à M_0		-1	1	-1	0
28	trajet de M_2 à M_0		-1	0	-1	0
29	trajet de M_2 à M_0		-1	0	-1	0
30	début de A_0, trajet de M_0 à M_1	M_0	-1	0	-1	0
31	trajet de M_0 à M_1		-1	0	-1	0
32	trajet de M_0 à M_1		-1	0	-1	0
33	trajet de M_0 à M_1		-1	0	-1	0
34	trajet de M_0 à M_1		-1	0	-1	0
35	trajet de M_0 à M_1		-1	0	-1	0
36	fin de A_0, trajet de M_1 à M_2	M_1	6	0	-1	0
37	trajet de M_1 à M_2		5	0	-1	0
38	trajet de M_1 à M_2		4	0	-1	0
39	trajet de M_1 à M_2		3	0	-1	0
40	trajet de M_1 à M_2		2	0	-1	0

t		pr	état de la cellule à t			
			temps d'usinage restant à			
			M_1	M_2	M_3	M_4
41	trajet de M_1 à M_2		1	0	-1	0
42	début de A_2, trajet de M_2 à M_3	M_2	0	0	-1	0
43	fin de A_2, trajet de M_3 à M_1	M_3	0	-1	0	0
44	trajet de M_3 à M_1	M_2	0	-1	0	0
45	trajet de M_3 à M_1		0	-1	0	0
46	trajet de M_3 à M_1		0	-1	0	0
47	trajet de M_3 à M_1		0	-1	0	0
48	trajet de M_3 à M_1		0	-1	0	0
49	trajet de M_3 à M_1		0	-1	0	0
50	début de A_1, trajet de M_1 à M_2	M_1	0	-1	0	0
51	trajet de M_1 à M_2		-1	-1	0	0
52	trajet de M_1 à M_2		-1	-1	0	0
53	trajet de M_1 à M_2		-1	-1	0	0
54	trajet de M_1 à M_2		-1	-1	0	0
55	trajet de M_1 à M_2		-1	-1	0	0
56	fin de A_1, trajet de M_2 à M_4	M_2	-1	10	0	0
57	trajet de M_2 à M_4	M_3	-1	9	0	0
58	début de A_4, trajet de M_4 à M_5	M_4	-1	8	0	0
59	fin de A_4, trajet de M_5 à M_3	M_5	-1	7	0	-1
60	trajet de M_5 à M_3	M_4	-1	6	0	-1
61	début de A_3, trajet de M_3 à M_4	M_3	-1	5	0	-1
62	fin de A_3, début de A_4, trajet de M_4 à M_5	M_4	-1	4	-1	0
63	fin de A_4, trajet de M_5 à M_2	M_5	-1	3	-1	-1
64	trajet de M_5 à M_2	M_4	-1	2	-1	-1
65	trajet de M_5 à M_2	M_3	-1	1	-1	-1
66	début de A_2, trajet de M_2 à M_3	M_2	-1	0	-1	-1
67	fin de A_2, début de A_3, trajet de M_3 à M_4	M_3	-1	-1	0	-1
68	fin de A_3, trajet de M_4 à M_0	M_4	-1	-1	-1	0
69	trajet de M_4 à M_0	M_3	-1	-1	-1	0
70	trajet de M_4 à M_0	M_2	-1	-1	-1	0
71	trajet de M_4 à M_0		-1	-1	-1	0
72	trajet de M_4 à M_0		-1	-1	-1	0
73	trajet de M_4 à M_0		-1	-1	-1	0
74	trajet de M_4 à M_0		-1	-1	-1	0
75	trajet de M_4 à M_0		-1	-1	-1	0
76	trajet de M_4 à M_0	M_1	-1	-1	-1	0
77	trajet de M_4 à M_0		-1	-1	-1	0
78	trajet de M_4 à M_0		-1	-1	-1	0
79	trajet de M_4 à M_0		-1	-1	-1	0
80	trajet de M_4 à M_0		-1	-1	-1	0
81	trajet de M_4 à M_0		-1	-1	-1	0
82	fin du cycle	M_0	-1	-1	-1	0

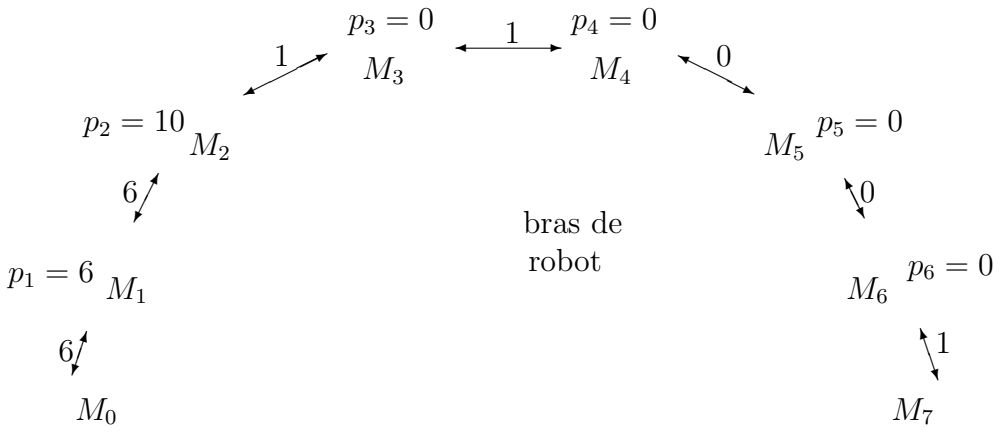

Figure 3.12: L'instance I_6

$$\begin{aligned} &= 4 * 6 + 6 * (6 + 1 + 1) + 4 * (1) \\ &= 76. \end{aligned}$$

De plus, comme $p_h = 0$, il n'y a pas d'attente aux machines M_h pour $3 \leq h \leq m$. Par conséquent, nous ne devons considérer que les temps d'attente à la machine M_1 avant l'exécution de A_1 et à la machine M_2 avant l'exécution de A_2. La séquence A_0A_1 génère un temps d'attente de $p_1 = 6$ unités de temps. La séquence $A_0A_2A_1$ ne génère pas de temps d'attente car le temps de transport entre la fin de A_0 et le début de A_1 est égal à 14 ce qui est supérieur à p_1. De la même manière, on peut vérifier que la séquence $A_1A_0A_2$ ne génère pas de temps d'attente à la machine M_2. Dans la séquence $A_1U_mA_3U_mA_2$, le temps de transport entre la fin de l'exécution de A_1 et l'arrivée du robot à la machine M_2, de manière à exécuter l'activité A_2, est 10 ce qui est égal à p_2. Par conséquent, cette séquence n'induit pas de temps d'attente en M_2. De plus, comme pour tout h, $\epsilon_h^u = \epsilon_h^l = 0$, on a $T_L(C_2) = 0$. Par conséquent, le temps de cycle de C_2 vérifie

$$T(C_2) = T_T(C_2) + T_W(C_2) + T_L(C_2) = 76 + 6 + 0 = 82.$$

Concentrons nous maintenant sur les permutations pyramidales et prouvons que $T(\pi_j) \geq 42$ pour tout $j \in \{0, 1 \ldots \mu\}$. Nous considérons deux cas en fonction de la position de A_1 dans π_j.

D'abord, si A_1 est descendante, alors $m_1(\pi_j) = 4$ et $m_h(\pi_j) \geq 2$ pour tout h. On a donc

$$T_T(\pi_j) \geq 4 * 6 + 2 * (6 + 1 + 1 + 1) \geq 42.$$

Deuxièmement, si A_1 est montante, alors π_j est de la forme $(A_0A_1[\ldots]A_2)$ ou de la forme $(A_0A_1A_2[\ldots])$. Dans les deux cas, le robot doit

- exécuter A_0 [$\delta_0 = 6$ unités de temps] ;

- attendre en M_1 avant d'exécuter A_1 [$p_1 = 6$ unités de temps] ;
- exécuter A_1 [$\delta_1 = 6$ unités de temps] ;
- entre la fin de A_1 et le début de A_2, il y a au moins $p_2 = 10$ unités de temps ;
- exécuter A_2 [$\delta_2 = 1$ unités de temps] ;
- revenir de M_3 à M_0 [$\delta_2 + \delta_1 + \delta_0 = 1 + 6 + 6 = 13$ unités de temps].

Par conséquent, le temps de cycle de π_j vérifie

$$\begin{aligned} T(\pi_j) &\geq 6 + 6 + 6 + 10 + 1 + 13 \\ &\geq 42. \end{aligned}$$

Nous avons prouvé que, pour l'instance I_m, $T(C_2)/2 = 41$ et que $T(\pi_j) \geq 42$ pour toute permutation pyramidale π_j. Par conséquent, il existe un 2-cycle qui domine strictement toutes les permutations pyramidales. □

3.4 Performance des 1-cycles

Nous avons vu que, à partir de quatre machines, les 1-cycles ne sont plus dominants. Toutefois, dans la pratique, les 1-cycles ne semblent pas très éloignés des cycles optimaux. Ainsi, nous cherchons un facteur de performance des 1-cycles par rapport aux k-cycles.

Pour une instance donnée, soit C_1 le meilleur 1-cycle et soit C_{opt} le cycle optimal parmi tous les k-cycles. Soit k_{opt} le degré du cycle C_{opt}. Le facteur de performance des 1-cycles est la plus petite valeur λ qui vérifie, pour toutes les instances,

$$T(C_1) \leq \lambda \frac{T(C_{opt})}{k_{opt}}$$

Théorème 7 *Le facteur de performance des 1-cycles, λ, vérifie*

$$\lambda \leq \left(2 - \frac{\delta_0 + \delta_m}{\delta_0 + \delta_m + \sum_{i=1}^{m-1} \delta_i} \right) \leq 2 \tag{3.17}$$

Démonstration Notons $\mathcal{E} = \sum_{i=1}^{m+1} \epsilon_i^l + \sum_{i=0}^{m} \epsilon_i^u$. La Proposition 6, page 40 indique que ou bien la permutation descendante $\pi_d = (A_0 A_m A_{m-1} \dots A_1)$ est optimale, ce qui implique que l'inégalité (3.17) est vraie, ou bien $T(\pi_d) = 2\delta_0 + 4\sum_{i=1}^{m-1} \delta_i + 2\delta_m + \mathcal{E}$. Considérons le deuxième cas. Comme $T(C_1) \leq T(\pi_d)$, on a $T(C_1) \leq 2\delta_0 + 4\sum_{i=1}^{m-1} \delta_i + 2\delta_m + \mathcal{E}$. De plus, pendant l'exécution d'un k-cycle,

- le robot parcourt au moins $2k$ deux fois le trajet entre M_i et M_{i+1} pour tout i entre 0 et m ;
- chaque machine $M_i (i = 0, 1 \dots m)$ est déchargée k fois ;
- chaque machine $M_i (i = 1, 2 \dots m+1)$ est chargée k fois.

Par conséquent, le temps de cycle de tout k-cycle C_k satisfait $T(C_k) \geq 2k \sum_{i=0}^{m} \delta_i + k\mathcal{E}$. On a donc

$$\begin{aligned} T(C_1) &\leq \left(\frac{2\delta_0 + 4\sum_{i=1}^{m-1} \delta_i + 2\delta_m + \mathcal{E}}{2\delta_0 + 2\sum_{i=1}^{m-1} \delta_i + 2\delta_m + \mathcal{E}} \right) \frac{T(C_k)}{k} \\ &\leq \left(2 - \frac{\delta_0 + \delta_m}{\delta_0 + \sum_{i=1}^{m-1} \delta_i + \delta_m} \right) \frac{T(C_k)}{k} \leq 2\frac{T(C_k)}{k} \end{aligned}$$

Comme cette inégalité est vraie pour tout k-cycle, elle est également vraie pour le cycle optimal. □

Dans [27], Crama et van de Klundert ont décrit le facteur de performance de π_d par rapport aux 1-cycles. En effet, ils observent que ¡¡ *the algorithm that outputs* π_d*, [...], is a 2-approximation algorithm for the identical part scheduling problem* ¿¿.

Nous pensons que le facteur de performance proposé dans le Théorème 7 n'est pas serré, c'est-à-dire qu'il ne peut pas être atteint.

3.5 Conclusion

Dans ce chapitre, nous avons présenté deux approches différentes de la Conjecture des 1-cycles et de la dominance des permutations pyramidales. La deuxième approche nous a permis de décrire précisément les intervalles de validité de la conjecture. Le résultat original de ce chapitre est le contre-exemple qui clôt définitivement la conjecture pour des cellules robotisées dont les durées de transfert sont additives. Nous étudions, dans les chapitres suivants, quelques configurations spéciales de cellules pour lesquelles la Conjecture des 1-cycles a un intervalle de validité plus large.

Chapter 4

Configurations spéciales : extension de la conjecture

Nous avons vu dans le chapitre précédent que, pour le cas général, la Conjecture des 1-cycles est fausse pour des cellules à quatre machines et plus. Dans ce chapitre, nous imposons des contraintes supplémentaires sur les paramètres de la cellule afin d'identifier les limites de la validité de la conjecture. Nous considérons d'abord le cas régulier où toutes les machines sont équidistantes (Section 4.1). Dans ce cas, la Conjecture des 1-cycles est vraie pour $m = 4$ et $k = 2$ et fausse à partir de $m = 4$ et $k = 3$. Nous analysons ensuite le cas régulier équilibré (Section 4.2) où les temps d'usinage des pièces sont identiques sur toutes les machines. Dans ce cas, la Conjecture des 1-cycles est vraie pour des cellules à quatre machines. De plus, nous prouvons que, pour le cas régulier équilibré, le meilleur 1-cycle peut être trouvé en temps constant.[1]

4.1 Cas régulier

Dans cette section, nous étudions le cas régulier pour lequel les durées de transfert sont additives, toutes les machines sont équidistantes et les temps de chargement et de déchargement sont égaux. Cette configuration semble raisonnable dans l'industrie lorsque les machines sont de dimensions similaires. Dans [66], la Conjecture des 1-cycles avait été proposée pour le cas régulier.

Formellement, une instance est définie par
- le nombre de machines, m,
- la durée du trajet entre deux machines consécutives, δ,
- le temps de chargement ou de déchargement, ϵ,

[1] Les résultats présentés dans la Section 4.1 et dans la Section 4.2.2 ont été publiés dans les actes de la conférence IEPM'99 [17].

– les temps d'usinage $p_i (i = 1, 2 \dots m)$.

Les durées de transfert étant additives, le trajet pour aller de la machine M_i à la machine M_j dure $|j - i|\delta$ unités de temps.

Nous savons que la conjecture des 1-cycles est vraie pour $m = 2$ et pour $m = 3$ dans le cas additif [Chapitre 3]. Elle est donc également vraie dans le cas régulier pour des cellules à deux et trois machines. À partir de quatre machines, les résultats sont légèrement différents. Nous étudions, dans le cas régulier, la Conjecture des 1-cycles pour $m = 4$ et $k = 2$. Puis, nous décrivons une instance pour laquelle un 3-cycle est strictement meilleur que tous les 1-cycles ce qui contredit la Conjecture pour $m = 4$ et $k = 3$.

Théorème 8 *Dans une cellule régulière à 4 machines, les 1-cycles dominent les 2-cycles.*

Idée de la démonstration Le fait que les 1-cycles dominent les 2-cycles nous paraît intéressant et ce résultat est dans la philosophie de ce travail. Toutefois, la preuve de ce théorème est très technique. Pour cette raison, nous n'en donnons, dans ce chapitre, que les principales idées. La preuve complète est présentée dans l'Annexe A, page 149.

La preuve du Théorème 8 est basée sur l'approche algébrique et plus particulièrement sur la Proposition 9 (page 68). L'idée est d'éliminer d'abord les cas très simples. Puis, étant donné un 2-cycle C_2, nous proposons une solution x du système $S_4(C_2)$ en fonction de la valeur de $m_2(C_2)$ et nous prouvons que x est une W-couverture pyramidale de C_2.

Nous décrivons maintenant les principaux éléments de la preuve. Les temps de cycle des permutations pyramidales, dans une cellule à quatre machines, sont donnés dans le Tableau 4.1 [Annexe B].

Le temps de chargement/déchargement de toute permutation pyramidale $\pi_j (j = 0, 1 \dots 7)$ vérifie $T_L(\pi_j) = 10\epsilon$. Pour le k-cycle C_k, le système $S_4(C_k)$ est donné par

$$S_4(C_k) = \begin{cases} x_0 + x_1 + x_2 + x_3 = u_1(C_k) \\ x_0 + x_2 + x_4 + x_6 = u_4(C_k) \\ x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = k \\ 2x_0 + 2x_1 + 4x_2 + 4x_3 + 2x_4 + 2x_5 + 4x_6 + 4x_7 = min(m_2(C_k), 4k) \\ x_i \text{ entier} \quad \text{pour } i = 0, 1 \dots 7 \end{cases}$$

La preuve est décomposée en cinq étapes qui couvrent tous les cas possibles :

- **Étape 0** Les cas où une permutation pyramidale est optimale de façon triviale sont éliminés (lorsque, par exemple, le temps de cycle d'une permutation pyramidale est égal à une borne inférieur).

Tableau 4.1: Temps de cycle des permutations pyramidales pour $m = 4$

j	π_j	$T_T(\pi_j)$	$T_W(\pi_j)$
0	$(A_0A_1A_2A_3A_4)$	10δ	$p_1 + p_2 + p_3 + p_4$
1	$(A_0A_1A_2A_4A_3)$	12δ	$p_1 + p_2 +$ $\max(0, p_3 - 4\delta - 2\epsilon, p_4 - 8\delta - 6\epsilon - p_1 - p_2)$
2	$(A_0A_1A_3A_4A_2)$	12δ	$p_1 + p_4 +$ $\max(0, p_2 - 6\delta - 4\epsilon - p_4, p_3 - 6\delta - 4\epsilon - p_1)$
3	$(A_0A_1A_4A_3A_2)$	14δ	$p_1 + \max(0, p_2 - 8\delta - 4\epsilon,$ $p_3 - 10\delta - 6\epsilon - p_1, p_4 - 10\delta - 6\epsilon - p_1)$
4	$(A_0A_2A_3A_4A_1)$	12δ	$p_3 + p_4 +$ $\max(0, p_1 - 8\delta - 6\epsilon - p_3 - p_4, p_2 - 4\delta - 2\epsilon)$
5	$(A_0A_2A_4A_3A_1)$	14δ	$\max(0, p_1 - 10\delta - 6\epsilon, p_2 - 4\delta - 2\epsilon,$ $p_3 - 4\delta - 2\epsilon, p_4 - 10\delta - 6\epsilon, p_2 + p_3 - 8\delta - 4\epsilon)$
6	$(A_0A_3A_4A_2A_1)$	14δ	$p_4 + \max(0, p_1 - 10\delta - 6\epsilon - p_4,$ $p_2 - 10\delta - 6\epsilon - p_4, p_3 - 8\delta - 4\epsilon)$
7	$(A_0A_4A_3A_2A_1)$	16δ	$\max_i(0, p_i - 12\delta - 6\epsilon)$

- **Étape 1** Si, pour un k-cycle C_k, $u_4(C_k) = k$ ou $u_1(C_k) = k$ alors, la condition (C1) est vraie (dans les étapes suivantes, nous considérons donc que $u_1(C_k) < k$ et $u_4(C_k) < k$).
- **Étape 2** Si le k-cycle C_k vérifie $m_2(C_k) = 2k$ alors (C4) est vraie.
- **Étape 3** Si le 2-cycle C_2 vérifie $m_2(C_2) = 6$, alors (C1) ou (C2) ou (C4) est vraie.
- **Étape 4** Si le 2-cycle C_2 vérifie $m_2(C_2) \geq 8$ alors, (C2) ou (C4) est vraie.

La suite de la preuve est présentée dans l'Annexe A, page 149. □

Ainsi, la Conjecture des 1-cycles est vraie pour $k = 2$. Elle devient fausse à partir de $m = 4$ et $k = 3$. En effet, considérons l'instance I suivante [Figure 4.1] :

$$m = 4 \ ; \qquad \delta = 1 \ ; \qquad \epsilon = 0 \ ;$$

$$p_1 = 0 \ ; \qquad p_2 = 10 \ ; \qquad p_3 = 10 \ ; \qquad p_4 = 0.$$

Soit $C_3 = (A_0A_1A_4A_3A_4A_2A_0A_1A_0A_3A_4A_2A_1A_3A_2)$. Pour l'instance I, le 3-cycle C_3 domine strictement tous les 1-cycles, c'est-à-dire,

$$\frac{T(C_3)}{3} < T(\pi_j), \text{ pour tous les 1-cycles } \pi_j.$$

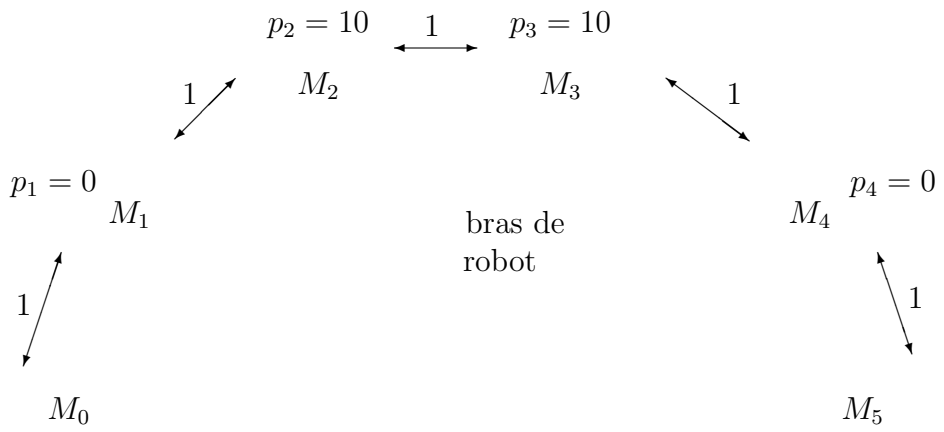

Figure 4.1: L'instance I

Nous savons [27] qu'il est suffisant de vérifier cette inégalité pour les $2^{m-1} = 8$ permutations pyramidales π_j $(j = 0, 1 \dots 7)$ dont les temps de cycle sont donnés dans le Tableau 4.1. Pour l'instance I, la meilleure permutation pyramidale dure 16 unités de temps. Or, le temps de cycle de C_3 vérifie $T(C_3) = 46$. Donc, comme $T(C_3)/3 = 15.3333$, le 3-cycle C_3 domine strictement tous les 1-cycles.

Avec la même instance, nous pouvons construire un 4-cycle pour lequel $T(C_k)/k$ est réduit à 15 unités de temps. Ce cycle est

$$(A_0A_1A_0A_3A_4A_2A_1A_0A_3A_2A_1A_4A_3A_2A_0A_1A_4A_3A_4A_2).$$

Remarquons que pour le cas régulier, le Théorème 7 (page 85) peut être réécrit comme suit :

Corollaire 2 *Le facteur de performance des 1-cycles, λ, vérifie*

$$\lambda \leq \left(2 - \frac{2}{m+1}\right) \tag{4.1}$$

Par exemple, dans une cellule à quatre machines, le meilleur 1-cycle est moins de 1.6 fois l'optimum. Pour quatre machines, ce facteur de performance n'est pas serré et nous travaillons à l'améliorer. Nous savons que λ est supérieur à 16/15 (voir le contre-exemple précédent), qu'il est inférieur à 9/8 et que ce nombre n'est pas serré non plus. Ce facteur de 9/8 a été trouvé grâce à une étude de cas assez technique. Pour cette raison, nous ne présentons pas la preuve. Finalement, l'intervalle à réduire est $[1.06666; 1.125]$.

4.2 Cas régulier équilibré

Dans le cas régulier équilibré, nous montrons que l'indice de la permutation pyramidale optimale peut être trouvé en temps constant.

Dans cette section, nous étudions le cas régulier équilibré, qui est un cas particulier du cas régulier. Nous ajoutons l'hypothèse que tous les temps d'usinage sont égaux ($p_i = p$ pour tout i). Une instance est donc définie par
- le nombre de machines, m,
- le temps de transfert, δ,
- le temps de chargement ou de déchargement, ϵ,
- le temps d'usinage, p.

Pour le cas régulier (étudié dans la section précédente), le meilleur 1-cycle peut être trouvé en temps polynomial et la Conjecture des 1-cycles est vraie pour $m \leq 3$. Pour le cas régulier équilibré, nous prouvons que l'indice du meilleur 1-cycle peut être trouvé en temps constant ce qui rend la Conjecture des 1-cycles encore plus intéressante. Puis, nous présentons de nouveaux résultats sur la validité de cette conjecture.

Nous notons $[a..b]$ l'intervalle de tous les entiers entre a et b. Rappelons que $u(\pi)$ désigne le nombre d'activités montantes dans la permutation pyramidale π et $u_i(\pi)$ est le nombre d'occurrences de la séquence $A_{i-1}A_i$ dans π.

4.2.1 Meilleur 1-cycle

Dans cette section, nous décrivons un ensemble dominant de 1-cycles.[2]

Considérons les $\frac{m}{2} + 1$ permutations pyramidales suivantes pour $m \geq 4$ et m pair :

$$\begin{aligned} \pi_0 &= (A_0 A_1 A_2 \dots A_m) \\ \pi_\alpha &= (A_0 A_{\alpha+1} A_{\alpha+3} \dots A_{m-\alpha-1} A_m A_{m-1} A_{m-2} \dots A_{m-\alpha} A_{m-\alpha-2} \\ &\quad A_{m-\alpha-4} \dots A_{\alpha+2} A_\alpha A_{\alpha-1} A_{\alpha-2} \dots A_2 A_1) \text{ pour } \alpha \in [1..\frac{m}{2} - 1] \\ \pi_{m/2} &= (A_0 A_m A_{m-1} \dots A_2 A_1) \end{aligned} \tag{4.2}$$

Le lemme suivant donne les temps de cycle de ces permutations.

Lemme 7 *On a*

$$\begin{aligned} T(\pi_0) &= 2(m+1)\delta + 2(m+1)\epsilon + mp \\ T(\pi_\alpha) &= (3m + 2\alpha)\delta + 2(m+1)\epsilon + 2\max(0, p - 4\alpha\delta - 2\alpha\epsilon) \end{aligned} \tag{4.3}$$

[2]Wieslaw Kubiak, de la *Memorial University of Newfoundland* au Canada, nous a aidé à rédiger cette section.

$$\text{pour } \alpha \in [1..\frac{m}{2}-1]$$
$$T(\pi_{m/2}) \;=\; 4m\delta + 2(m+1)\epsilon + \max(0, p-4(m-1)\delta - 2(m-1)\epsilon)$$

Démonstration Pour des raisons de clarté, nous posons $\epsilon = 0$ dans les preuves. Toutefois, toutes les preuves peuvent être généralisées à ϵ non nul.

Le cycle π_α contient

$$\frac{(m-\alpha-1)-(\alpha+1)}{2}+1+2 = (\frac{m}{2}-\alpha+2)$$

activités montantes. Donc, le temps de trajet de π_α est

$$T_T(\pi_\alpha) = 2(\frac{m}{2}-\alpha+2)\delta + 4(m+1-\frac{m}{2}+\alpha-2)\delta = (3m+2\alpha)\delta.$$

Calculons maintenant le temps d'attente de π_α. En règle générale si, dans un cycle π, les activités A_j et A_{j+1}, dans cet ordre, sont comprises entre les activités A_i et A_{i+1}, dans cet ordre, alors l'attente en M_{i+1} est nulle. En effet, entre l'exécution de A_j et l'exécution de A_{j+1}, il s'écoule un temps supérieur ou égal à p (temps minimum pendant lequel une pièce doit rester sur M_{j+1}). Par conséquent entre la fin de l'exécution de l'activité A_i et l'arrivée du robot en M_{i+1} pour exécuter A_{i+1}, il s'écoule un temps supérieur à p. Donc aucun temps d'attente n'est nécessaire à la machine M_{i+1} dans π.

L'équation (4.2) indique que, dans π_α, les activités A_α et $A_{\alpha+1}$ sont entre A_i et A_{i+1} pour $i = \alpha+2, \alpha+4 \ldots m-\alpha-4, m-\alpha-2$ et pour $i = m-\alpha, m-\alpha+1 \ldots m-2, m-1$ et $A_{m-\alpha-1}$ et $A_{m-\alpha}$ sont entre A_i et A_{i+1} pour $i = 0, 1 \ldots \alpha-2, \alpha-1$ et pour $i = \alpha+1, \alpha+3 \ldots m-\alpha-5, m-\alpha-3$. Donc, il n'y a pas d'attente en M_i pour $i \neq m-\alpha$ et $i \neq \alpha+1$. De plus, entre la fin de $A_{m-\alpha-1}$ et le début de $A_{m-\alpha}$ (et entre la fin de A_α et le début de $A_{\alpha+1}$) le temps de trajet est $4\alpha\delta$ et il n'y a pas d'attente aux machines intermédiaires. Comme les temps d'attente en $M_{m-\alpha}$ et en $M_{\alpha+1}$ n'interfèrent pas, on a

$$T_W(\pi_\alpha) = 2\max(0, p-4\alpha\delta).$$

Finalement

$$T(\pi_\alpha) = (3m+2\alpha)\delta + 2\max(0, p-4\alpha\delta).$$

Pour $T(\pi_{\frac{m}{2}})$, le Lemme 7 est une conséquence immédiate de la Proposition 2.5, page 40. Pour π_0, on a de manière immédiate :

$$T_T(\pi_0) = 2(m+1)\delta \quad \text{et} \quad T_W(\pi_0) = \sum_{i=1}^{m} p_i.$$

□

Lemme 8 *Les permutations $\pi_0, \pi_1 \ldots \pi_{\frac{m}{2}}$ dominent toutes les permutations pyramidales.*

Nous avons besoin des deux propriétés suivantes pour démontrer ce lemme.

Propriété 1 Pour toute permutation pyramidale π telle que $m+1 \geq u(\pi) \geq \frac{m}{2}+2$, si $p \leq \delta$ alors $T(\pi) \geq T(\pi_0)$ sinon $T(\pi) \geq T(\pi_1)$.

Démonstration Soit π une permutations pyramidale avec $m/2+2 \leq u(\pi) \leq m+1$. Nous montrons que si $p \leq \delta$ alors π est dominée par π_0 sinon π est dominée par π_1. On a

$$\sum_{i=1}^{m} u_i(\pi) \geq 2u(\pi) - m - 2.$$

Donc le temps de cycle de π vérifie

$$\begin{aligned} T(\pi) &\geq \sum_{i=1}^{m} u_i(\pi)p + 2u(\pi)\delta + 4(m+1-u(\pi))\delta \\ &\geq (2u(\pi) - m - 2)p + (4m + 4 - 2u(\pi))\delta. \end{aligned}$$

Si $0 \leq p \leq 4\delta$, alors

$$\begin{aligned} T(\pi) &\geq (2u(\pi) - m - 2)p + (2m + 2 - 2u(\pi))\delta + 2(m+1)\delta \\ &\geq (2u(\pi) - m - 2)\min(p,\delta) + (2m + 2 - 2u(\pi))\min(p,\delta) + 2(m+1)\delta \\ &= m\min(p,\delta) + 2(m+1)\delta \\ &\geq \min(T(\pi_0), T(\pi_1)). \end{aligned}$$

Si $4\delta \leq p$, alors

$$\begin{aligned} T(\pi) &\geq 2p + (2u(\pi) - m - 4)p + (4m + 4 - 2u(\pi))\delta \\ &\geq 2p + (2u(\pi) - m - 4)4\delta + (4m + 4 - 2u(\pi))\delta \\ &\geq 2p + (6u(\pi) - 12)\delta \\ &\geq 2p + 3m\delta \\ &\geq 2p + (3m - 6)\delta \\ &\geq T(\pi_1). \end{aligned}$$

De plus, π_0 domine π_1 pour $p \leq \delta$ et π_1 domine π_0 pour $\delta \leq p$. □

Propriété 2 Si $u(\pi) = u(\pi_\alpha) = \frac{m}{2} - \alpha + 2$ pour une permutation π_α et $\alpha \in [1..\frac{m}{2} - 1]$ alors $T(\pi) \geq T(\pi_\alpha)$.

Démonstration Soit π une permutation pyramidale qui contient $u(\pi_\alpha) = \frac{m}{2} - \alpha + 2$ activités montantes. Cette propriété implique que $T_T(\pi) = T_T(\pi_\alpha)$. Par conséquent, si $T_W(\pi_\alpha) = 0$, alors $T_W(\pi_\alpha) \leq T_W(\pi)$ et π_α domine π.

Supposons que $T_W(\pi_\alpha) > 0$, c'est-à-dire $p \geq 4\alpha\delta$. Alors $T_W(\pi_\alpha) = 2p - 8\alpha\delta$. Notons f l'indice de la première activité montante de π après A_0 et l l'indice de la dernière activité montante de π avant A_m. Comme $u(\pi) \geq 3$, f et l existent toujours et ils peuvent être égaux. Nous considérons les trois cas suivants.

Si $(l - f) \leq m - 2\alpha - 4$, alors

$$\begin{aligned}
\sum_{i=1}^{m} u_i(\pi) &\geq 2(u(\pi) - 2) - (l - f) - 2 \\
&\geq 2(\frac{m}{2} - \alpha) - (l - f) - 2 \\
&\geq m - 2\alpha - m + 2\alpha + 4 - 2 \\
&\geq 2.
\end{aligned}$$

Par conséquent $T_W(\pi) \geq 2p \geq T_W(\pi_\alpha)$.

Si $(l - f) = m - 2\alpha - 3$, alors

$$\begin{aligned}
\sum_{i=1}^{m} u_i(\pi) &\geq 2(u(\pi) - 2) - (l - f) - 2 \\
&\geq 2(\frac{m}{2} - \alpha) - (l - f) - 2 \\
&\geq m - 2\alpha - m + 2\alpha + 3 - 2 \\
&\geq 1.
\end{aligned}$$

De plus, $f < \alpha + 1$ ou $l > m - \alpha - 1$. Donc

$$T_W(\pi) \geq p + \max(0, p - 4\alpha\delta) \geq T_W(\pi_\alpha).$$

Si $(l - f) \geq m - 2\alpha - 2$ alors

$$\begin{aligned}
T_W(\pi) &\geq \max(0, p - 4(f - 1)\delta) + \max(0, p - 4(m - l - 1)\delta) \\
&\geq p - 4(f - 1)\delta + p - 4(m - l - 1)\delta \\
&\geq 2p - 4(-m + 2\alpha + 2 - 1 + m - 1)\delta \\
&\geq 2p - 8\alpha\delta \\
&\geq T_W(\pi_\alpha).
\end{aligned}$$

Pour les trois cas considérés $T_W(\pi) \geq T_W(\pi_\alpha)$. Or $T_T(\pi) = T_T(\pi_\alpha)$ implique que

$$T(\pi) \geq T(\pi_\alpha)$$

et π_α domine π. □

La preuve du Lemme 8 se poursuit comme suit.

Démonstration du Lemme 8 La seule permutation pyramidale telle que $u(\pi) = 2$ est $\pi_{\frac{m}{2}}$. Si $\frac{m}{2} + 2 \leq u(\pi) \leq m + 1$, alors la Propriété 1 indique que $T(\pi) \geq \min(T(\pi_0), T(\pi_1))$. Sinon, $u(\pi) = \frac{m}{2} + 2 - \alpha$ pour un $\alpha \in [1..\frac{m}{2} - 1]$. Par conséquent, d'après la Propriété 2, une permutation optimale est l'une des permutations pyramidales $\pi_0, \pi_1, \ldots, \pi_{\frac{m}{2}}$. □

Le Lemme 8 réduit la recherche d'un 1-cycle optimal à l'ensemble $\pi_0, \pi_1 \ldots \pi_{\frac{m}{2}}$. L'indice *opt* et le temps de cycle de la permutation optimale peuvent facilement être calculés à partir des valeurs de m, p, δ, et ϵ. Les détails sont donnés dans le Théorème 9.

Théorème 9 *Nous avons*

$$opt = \begin{cases} 0 & si\ 0 \leq p \leq \delta\ ; \\ \alpha & si\ (4\alpha - 3)\delta + 2(\alpha - 1)\epsilon \leq p \leq (4\alpha + 1)\delta + 2\alpha\epsilon\ ; \quad \alpha \in [1..\frac{m}{2} - 1]\ ; \\ \frac{m}{2} & si\ (2m - 3)\delta + (m - 2)\epsilon \leq p. \end{cases}$$

Nous avons besoin des deux propriétés suivantes pour prouver ce théorème.

Propriété 3 Si $p \geq (4\alpha - 3)\delta$ pour un $\alpha \in [1..\frac{m}{2}]$, alors $T(\pi_\alpha) \leq T(\pi_{\alpha'})$ pour $\alpha' \in [0..\alpha - 1]$.

Démonstration Si $p \geq (4\alpha - 3)\delta$ et $1 \leq \alpha' \leq \alpha - 1$ alors $p \geq 4\alpha'\delta$. Par conséquent, d'après l'équation (4.3), on a

$$T(\pi_{\alpha'}) = (3m - 6\alpha')\delta + 2p \geq (3m - 6\alpha + 6)\delta + 2p \geq (3m + 2\alpha)\delta.$$

Donc pour $\alpha < \frac{m}{2}$, on obtient

$$T(\pi_\alpha) = \max((3m + 2\alpha)\delta, (3m - 6\alpha)\delta + 2p) \leq T(\pi_{\alpha'})$$

et pour $\alpha = \frac{m}{2}$ on a

$$T(\pi_\alpha) = \max(4m\delta, 4\delta + p) \leq T(\pi_{\alpha'}).$$

De plus, on a $p \geq \delta$ pour $\alpha' = 0$. Donc, d'après la Propriété 1, π_1 domine π_0. Par conséquent π_α domine également π_0. □

Propriété 4 Si $p \leq (4\alpha + 1)\delta$ pour un $\alpha \in [0..\frac{m}{2} - 1]$ alors $T(\pi_\alpha) \leq T(\pi_{\alpha'})$ pour tout $\alpha' \in [\alpha + 1..\frac{m}{2}]$.

Démonstration Si $p \leq (4\alpha + 1)\delta$ et $\frac{m}{2} \geq \alpha' \geq \alpha + 1$ alors $p \leq 4\alpha'\delta$. Par conséquent, d'après l'équation (4.3), on a

$$\begin{aligned} T(\pi_{\alpha'}) &= (3m + 2\alpha')\delta \\ &\geq (3m + 2\alpha + 2)\delta. \end{aligned}$$

Cette inégalité est également valide pour $\alpha' = \frac{m}{2}$ car $p \leq 4(m - 1)\delta$.

Or, si $\alpha > 0$ alors

$$\begin{aligned} T(\pi_\alpha) &= \max((3m+2\alpha)\delta, (3m-6\alpha)\delta + 2p) \\ &\leq \max((3m+2\alpha)\delta, (3m+2\alpha+2)\delta) \\ &\leq T(\pi_{\alpha'}) \end{aligned}$$

et si $\alpha = 0$ alors

$$T(\pi_0) = 2(m+1)\delta + mp \leq (3m+2)\delta \leq T(\pi_{\alpha'}).$$

□

Nous avons maintenant tous les éléments pour prouver le Théorème 9.

Démonstration du Théorème 9 :

Si $(4\alpha-3)\delta \leq p \leq (4\alpha+1)\delta$ pour un $\alpha \in [1..\frac{m}{2}-1]$, alors, d'après la Propriété 3, $T(\pi_\alpha) \leq T(\pi_{\alpha'})$ pour $\alpha' \in [0, \alpha-1]$ et d'après la Propriété 4, $T(\pi_\alpha) \leq T(\pi_{\alpha'})$ pour $\alpha' \in [\alpha+1..\frac{m}{2}]$. Par conséquent, π_α est optimale.

Si $p \leq \delta$, alors d'après la Propriété 4, $T(\pi_0) \leq T(\pi_{\alpha'})$ pour $\alpha' \in [1, \frac{m}{2}]$. Ainsi, π_0 est optimale.

Finalement, si $p \geq (2m-3)\delta$, alors, d'après la Propriété 3, $T(\pi_{\frac{m}{2}}) \leq T(\pi_{\alpha'})$ pour $\alpha' \in [0..\frac{m}{2}-1]$. Ainsi, $\pi_{\frac{m}{2}}$ est optimal. □

Le Théorème 9 indique que l'indice, α, du meilleur 1-cycle peut être trouvé en temps constant. En effet,

- si $p - \delta \leq 0$ alors α vaut 0,
- si $p - (2m-3)\delta - (m-2)\epsilon \geq 0$ alors α vaut $\dfrac{m}{2}$,
- sinon α est l'entier compris entre $\dfrac{p-\delta}{4\delta+2\epsilon}$ et $\dfrac{p+3\delta+2\epsilon}{4\delta+2\epsilon}$.

Le cas m impair est similaire. Considérons les permutations pyramidales suivantes pour $m \geq 3$ et m impair :

$$\begin{aligned} \pi_0 &= (A_0A_1A_2A_3\ldots A_m) \\ \pi_0^1 &= (A_0A_1A_3\ldots A_{m-2}A_mA_{m-1}A_{m-3}\ldots A_4A_2) \\ \pi_0^2 &= (A_0A_2A_4\ldots A_{m-1}A_mA_{m-2}A_{m-4}\ldots A_3A_1) \\ \pi_\alpha^1 &= (A_0A_{\alpha+1}A_{\alpha+3}\ldots A_{m-\alpha-2}A_mA_{m-1}A_{m-2}\ldots A_{m-\alpha-1} \\ &\quad A_{m-\alpha-3}A_{m-\alpha-5}\ldots A_{\alpha+2}A_\alpha A_{\alpha-1}A_{\alpha-2}\ldots A_2A_1) \\ \pi_\alpha^2 &= (A_0A_{\alpha+2}A_{\alpha+4}\ldots A_{m-\alpha-1}A_mA_{m-1}A_{m-2}\ldots A_{m-\alpha} \\ &\quad A_{m-\alpha-2}A_{m-\alpha-4}\ldots A_{\alpha+3}A_{\alpha+1}A_{\alpha-1}A_{\alpha-2}\ldots A_2A_1) \\ \pi_{\frac{m-1}{2}} &= (A_0A_mA_{m-1}\ldots A_2A_1) \end{aligned}$$

On a

$$\begin{aligned}
T(\pi_0) &= 2(m+1)\delta + 2(m+1)\epsilon + mp \\
T(\pi_0^1) &= T(\pi_0^2) = (3m+1)\delta + 2(m+1)\epsilon + p + \max(0, p - 4\delta - 2\epsilon) \\
T(\pi_\alpha^1) &= T(\pi_\alpha^2) = (3m + 2\alpha + 1)\delta + 2(m+1)\epsilon + \\
&\quad \max(0, p - 4\alpha\delta - 2\alpha\epsilon) + \max(0, p - 4(\alpha+1)\delta - 2(\alpha+1)\epsilon) \\
T(\pi_{\frac{m-1}{2}}) &= 4m\delta + 2(m+1)\epsilon + \max(0, p - 4(m-1)\delta - 2(m-1)\epsilon)
\end{aligned}$$

Lemme 9 *Les permutations* $\pi_0, \pi_0^1, \pi_1^1 \ldots \pi_{\frac{m-3}{2}}^1, \pi_{\frac{m-1}{2}}$ *dominent toutes les permutations pyramidales.*

Théorème 10 *La permutation pyramidale optimale est*

$$opt = \begin{cases} \pi_0 & si\ 0 \le p \le \delta\ ; \\ \pi_0^1 & si\ \delta \le p \le 2\delta \\ \pi_\alpha^1 & si\ (4\alpha - 2)\delta + 2(\alpha - 1)\epsilon \le p \le (4\alpha + 2)\delta + 2\alpha\epsilon\ ; \quad \alpha \in [1..\frac{m-3}{2}] \\ \pi_{\frac{m-1}{2}} & si\ (2m-4)\delta + (m-3)\epsilon \le p. \end{cases}$$

4.2.2 Validité de la conjecture pour m = 4

Nous avons prouvé [Théorème 9] que, pour le cas régulier équilibré, le meilleur 1-cycle parmi tous les 1-cycles pouvait être trouvé en temps constant et nous avons décrit les meilleurs 1-cycles. La dominance des 1-cycles sur les k-cycles n'en est que plus intéressante.

Nous étudions l'extension de la Conjecture des 1-cycles et nous présentons de nouveaux résultats sur la validité de cette conjecture.

Théorème 11 *Dans le cas régulier équilibré, la Conjecture des 1-cycles est vraie pour* $m = 4$.

Démonstration

D'après le Théorème 9, pour $m = 4$, le meilleur 1-cycle est

pour $0 \le p \le \delta$,
$\pi_0 = A_0A_1A_2A_3A_4$ et $T(\pi_0) = 10\delta + 10\epsilon + 4p$;
pour $\delta \le p \le 5\delta + 2\epsilon$
$\pi_1 = A_0A_2A_4A_3A_1$ et $T(\pi_1) = 14\delta + 10\epsilon + 2\max(0, p - 4\delta - 2\epsilon)$;
pour $5\delta + 2\epsilon \le p$
$\pi_2 = A_0A_4A_3A_2A_1$ et $T(\pi_2) = 16\delta + 10\epsilon + \max(0, p - 12\delta - 6\epsilon)$.

Soit C_k un k-cycle. Nous montrons que C_k est dominé par le meilleur 1-cycle. Pour des raisons de clarté, nous posons $u_i = u_i(C_k)$ et $|S|$ représente le nombre d'occurrences de la séquence d'activités S dans C_k. La Définition 1 (page 31) et

les Propositions 1 (page 31) et 3 (page 32) indiquent que

$$\begin{aligned} m_2(C_k) &\geq 4k - |A_1A_0A_2| - u_2 - |A_2A_4A_3| - u_3 && (4.4)\\ T(C_k) &\geq (2k + 4k - 2u_1 + m_2(C_k) + 4k - 2u_4 + 2k)\delta + 10k\epsilon \\ &\quad +(u_1 + u_2 + u_3 + u_4)p && (4.5)\\ &\quad +(|A_1A_0A_2| + |A_2A_4A_3|)\max(0, p - 4\delta - 2\epsilon) \\ u_i &\leq k \qquad \forall i \end{aligned}$$

La dernière ligne de l'inégalité (4.5) est due au fait que le temps d'attente lors de l'exécution de la séquence $A_1A_0A_2$ ou de la séquence $A_2A_4A_3$ est $\max(0, p - 4\delta - 2\epsilon)$. Les activités A_0 et A_4 apparaissent chacune exactement k fois dans C_k. Par conséquent,

$$\begin{aligned} k &\geq |A_1A_0A_2| + |A_0A_1| \\ k &\geq |A_2A_4A_3| + |A_3A_4| \end{aligned}$$

où, par définition, $|A_0A_1| = u_1(C_k)$ et $|A_3A_4| = u_4(C_k)$.

Nous considérons trois cas :

- **Cas 1** $0 \leq p \leq 4\delta + 2\epsilon$;
- **Cas 2** $4\delta + 2\epsilon \leq p \leq 12\delta + 6\epsilon$;
- **Cas 3** $12\delta + 6\epsilon \leq p$.

Les bornes des intervalles des différents cas correspondent aux valeurs pour lesquelles certains temps d'attente passent de 0 à une valeur positive. En fait, dans le cas 1, nous prouvons que pour $0 \leq p \leq \delta$, le cycle π_0 domine C_k et pour $\delta \leq p \leq 4\delta + 2\epsilon$, le cycle π_1 domine C_k. De même, dans le cas 2, nous prouvons que pour $4\delta + 2\epsilon \leq p \leq 5\delta + 2\epsilon$, le cycle π_1 domine C_k et pour $5\delta + 2\epsilon \leq p \leq 12\delta + 6\epsilon$, le cycle π_2 domine C_k.

Cas 1 $0 \leq p \leq 4\delta + 2\epsilon$

Les inégalités (4.4) et (4.5) indiquent que

$$\begin{aligned} T(C_k) &\geq (16k - |A_1A_0A_2| - u_2 - |A_2A_4A_3| - u_3 - 2u_1 - 2u_4)\delta + 10k\epsilon \\ &\quad +(u_1 + u_2 + u_3 + u_4)p \\ &\geq 16k\delta + 10k\epsilon - (|A_1A_0A_2| + |A_2A_4A_3| + u_1 + u_4)\delta \\ &\quad +(u_1 + u_2 + u_3 + u_4)(p - \delta) \\ &\geq 14k\delta + 10k\epsilon + (u_1 + u_2 + u_3 + u_4)(p - \delta). \end{aligned}$$

Or les inégalités $p - \delta \geq \min(0, p - \delta)$ et $\min(0, p - \delta) \leq 0$ et $u_1 + u_2 + u_3 + u_4 \leq 4k$ impliquent que

$$T(C_k) \geq 14k\delta + 10k\epsilon + (u_1 + u_2 + u_3 + u_4)\min(0, p - \delta)$$

$$\begin{aligned} &\geq 14k\delta + 10k\epsilon + 4k\min(0, p-\delta) \\ &\geq 10k\delta + 10k\epsilon + 4k\min(\delta, p). \end{aligned}$$

Comme $\min(T(\pi_0), T(\pi_1)) = 10\delta + 10\epsilon + 4\min(p, \delta)$, l'inégalité précédente implique que

$$T(C_k) \geq k\min(T(\pi_0), T(\pi_1)).$$

Cas 2 $\quad 4\delta + 2\epsilon \leq p \leq 12\delta + 6\epsilon$

Supposons que $m_2(C_k) = 2k + \alpha$ pour $(\alpha \geq 0)$. L'inégalité (4.4) devient

$$|A_1A_0A_2| + u_2 + |A_2A_4A_3| + u_3 \geq 2k - \alpha.$$

De plus,

$$|A_1A_0A_2| + u_2 + |A_2A_4A_3| + u_3 \geq 0.$$

Par conséquent,

$$|A_1A_0A_2| + u_2 + |A_2A_4A_3| + u_3 \geq \max(0, 2k - \alpha) \tag{4.6}$$

L'inégalité (4.5) implique que

$$\begin{aligned} T(C_k) \quad \geq \quad & (12k + 2k + \alpha - 2u_1 - 2u_4)\delta + 10k\epsilon \\ & +(u_1 + u_2 + u_3 + u_4)p + (|A_1A_0A_2| + |A_2A_4A_3|)(p - 4\delta - 2\epsilon) \\ & +(u_2 + u_3)(4\delta + 2\epsilon) - (u_2 + u_3)(4\delta + 2\epsilon) \\ \geq \quad & 14k\delta + 10k\epsilon + \alpha\delta + (u_2 + u_3 + |A_1A_0A_2| + |A_2A_4A_3|)(p - 4\delta - 2\epsilon) \\ & +(u_1 + u_4)(p - 2\delta) + (u_2 + u_3)(4\delta + 2\epsilon). \end{aligned}$$

Comme $(u_1 + u_4)(p - 2\delta) + (u_2 + u_3)(4\delta + 2\epsilon) \geq 0$, et $p - 4\delta - 2\epsilon \geq 0$, l'inégalité (4.6) implique que

$$T(C_k) \quad \geq \quad 14k\delta + 10k\epsilon + \alpha\delta + \max(0, 2k - \alpha)(p - 4\delta - 2\epsilon).$$

Nous considérons deux cas qui dépendent des valeurs de α. Si $\alpha \geq 2k$ alors le temps de cycle de C_k vérifie

$$\begin{aligned} T(C_k) \quad &\geq \quad 14k\delta + 10k\epsilon + \alpha\delta \\ &\geq \quad 16k\delta + 10k\epsilon \\ &\geq \quad kT(\pi_2). \end{aligned}$$

Sinon, $\alpha \leq 2k$ et

$$\begin{aligned} T(C_k) \quad &\geq \quad 14k\delta + 10k\epsilon + \alpha\delta + 2kp - 8k\delta - \alpha p + 4\alpha\delta - 4k\epsilon + 2\alpha\epsilon \\ &\geq \quad 6k\delta + 6k\epsilon + 2kp + \alpha(5\delta + 2\epsilon - p) \\ &\geq \quad 6k\delta + 6k\epsilon + 2kp + \alpha\min(0, 5\delta + 2\epsilon - p). \end{aligned}$$

Comme $\min(0, 5\delta + 2\epsilon - p) \leq 0$ et $\alpha \leq 2k$, le temps de cycle $T(C_k)$ vérifie

$$\begin{aligned} T(C_k) &\geq 6k\delta + 6k\epsilon + 2kp + 2k\min(0, 5\delta + 2\epsilon - p) \\ &\geq k\min(6\delta + 6\epsilon + 2p, 16\delta + 10\epsilon) \\ &\geq k\min(T(\pi_1), T(\pi_2)). \end{aligned}$$

Cas 3 $12\delta + 6\epsilon \leq p$

Dans ce cas, $T(\pi_2) = 4\delta + 4\epsilon + p$. D'après le Théorème 2 (page 39) la permutation pyramidale π_2 est optimale. □

Conclusion et perspectives de la deuxième partie

Dans cette partie, nous avons étudié la validité de la Conjecture des 1-cycles dans des cellules pour lesquelles les durées de transfert sont additives [Tableau 4.2]. Nous avons d'abord étudié le cas général puis, nous avons rajouté des contraintes sur les paramètres de la cellule. Pour le cas régulier où toutes les machines sont équidistantes, nous avons montré que la conjecture est vraie pour $m = 4$ et $k = 2$ puis qu'elle est fausse pour $m = 4$ et $k \geq 3$. Nous avons ensuite imposé, en plus, l'égalité des temps d'usinage sur toutes les machines. Dans ce cas, le meilleur 1-cycle peut être trouvé en temps constant et la Conjecture des 1-cycles est vraie pour $m = 4$.

Tableau 4.2: Validité de la Conjecture des 1-cycles

	$m = 2$	$m = 3$	$m = 4$		$m \geq 5$
			$k = 2$	$k \geq 3$	
durées de transfert additives	vraie [66, 29]		fausse		
cas régulier	vraie			fausse	
cas régulier équilibré	vraie				?

Pour le cas régulier équilibré, la validité de la Conjecture des 1-cycles est inconnue pour $m \geq 5$. Pour les autres cas, la conjecture est fausse pour des cellules à quatre machines et plus. Donc, plusieurs questions se posent :

- Quelle est la complexité de la recherche du meilleur cycle de production ?
- Pour un k fixé, quel est le meilleur cycle de production ?
- Peut-on borner k pour la recherche du meilleur cycle ?
- Quelle est la qualité (facteur de performance) des 1-cycles ?
- Peut-on caractériser des familles d'instances pour lesquelles les 1-cycles sont dominants ?

Concentrons-nous sur cette dernière question pour le cas régulier. En effet, on sait que, si l'un des temps d'usinage est grand ($\max(p_i) \geq 4(m-1)\delta$), alors la permutation descendante est optimale. Étudions maintenant le cas où les temps

d'usinage sont ¡¡ petits ¿¿.

Nous avons vu (Section 2.4.3) qu'une partie du temps de cycle pouvait être lue sur le line-graph, LG_m. Ainsi, puisque sous cette forme la Conjecture des 1-cycles est fausse, nous proposons une conjecture faible des 1-cycles. Définissons le temps de cycle réduit comme la somme des temps de trajet, des temps de chargement-déchargement et des temps d'attente générés par les séquences d'activités de la forme $A_h A_{h+1}$.

Conjecture faible des 1-cycles : *Pour les temps de cycle réduits, les 1-cycles dominent les k-cycles.*

Le temps de cycle réduit peut être lu directement sur le line-graph LG_m. Ainsi, le plus court circuit moyen du line-graph est le cycle de production optimal pour le temps de cycle réduit. La conjecture faible peut donc être reformulée comme suit : il existe un plus court circuit moyen de LG_m qui est un 1-cycle (circuit de longueur $m+1$).

L'intérêt de cette conjecture est que, si les temps d'usinage sont petits ($p_i \leq 4\delta$ pour tout i), alors le temps de cycle réduit est égal au temps de cycle. En effet, pour ce cas, les seuls temps d'attente sont ceux générés par les séquences $A_h A_{h+1}$.

Nous avons programmé la recherche du plus court circuit moyen dans les line-graphs, LG_m. Le module de test permet de tester la conjecture faible pour les instances telles que $\delta = 1$ et les temps d'usinage p_i prennent toutes les valeurs entières entre la valeur minimum (min) et la valeur maximum (max) entrées par l'utilisateur.

Nous avons testé la validité de la conjecture faible sur les instances suivantes

$$\begin{array}{lll} m=4 & min=0 & max=12\ ; \\ m=5 & min=0 & max=12\ ; \\ m=6 & min=0 & max=10\ ; \\ m=7 & min=0 & max=4. \end{array}$$

Parmi toutes ces instances, nous n'avons trouvé aucun contre-exemple à la Conjecture faible.

Nous avons démontré la conjecture faible des 1-cycles pour des systèmes à deux, trois et quatre machines. Il s'agit en fait d'une application directe des algorithmes décrits page 51. Par exemple, pour trois machines, les plus courtes distances entre les sommets pondérés par A_1 sont données dans la matrice suivante :

$$\begin{pmatrix} \min(4\delta + p_1 + p_2 + p_3, 8\delta + p_2, 6\delta + p_3) & \min(3\delta + p_1 + p_2, 5\delta) \\ \min(7\delta + p_1 + p_3, 9\delta + p_3, 11\delta) & \min(6\delta + p_1, 8\delta) \end{pmatrix}$$

Il reste alors à prouver que

$$\min(7\delta + p_1 + p_3, 9\delta + p_3, 11\delta) + \min(3\delta + p_1 + p_2, 5\delta)$$

$$\geq 2\min(4\delta + p_1 + p_2 + p_3, 8\delta + p_2, 6\delta + p_3, 6\delta + p_1, 8\delta)$$

La preuve pour $m = 4$ est similaire. Toutefois ces preuves sont très techniques et ne peuvent pas être généralisées à un nombre quelconque de machines. La validité de la Conjecture faible est donc un problème ouvert.

Part III

Autres formes de cellules robotisées

Introduction de la troisième partie

Dans les parties précédentes, les cellules robotisées vérifiaient les contraintes suivantes : les distances entre les machines étaient additives, les séjours des pièces sur les machines étaient de durées non limitées et il n'existait pas d'espace de stockage dans la cellule. Toutes ces contraintes peuvent être modifiées. Dans cette partie, nous allons aborder quelques modifications. Chaque modification est considérée indépendamment des autres. Pour chaque cas, nous étudions la dominance des permutations pyramidales et le rôle des 1-cycles. D'autres propriétés sont également décrites.

Dans le problème initial [5], les machines étaient disposées en cercle. Pour aller du lieu de déchargement au lieu de chargement, le robot devait passer par toutes les machines de la cellule. En effet, un convoyeur empêchait le robot de faire un tour complet. Dans un premier temps (Chapitre 5), nous considérons des cellules dans lesquelles le robot est autorisé à faire un tour complet. Les distances entre les machines restent additives mais sont définies différemment. Pour ce problème, nous démontrons que les permutations pyramidales ne sont pas dominantes et nous décrivons la liste des 1-cycles dominants pour des cellules à deux, trois et quatre machines.

Puis (Chapitre 6), nous étudions l'extension de certaines propriétés des cellules robotisées aux problèmes de type HSP (*Hoist Scheduling Problem*). Pour ces problèmes, le temps pendant lequel une pièce peut rester sur une machine doit appartenir à un intervalle. Le HSP est un problème classique, fréquemment rencontré dans l'industrie et pour lequel il existe une très vaste littérature. Nous démontrons que, pour le HSP, il n'existe pas toujours d'ordonnancement optimal actif, que les permutations pyramidales ne dominent pas les 1-cycles et que la Conjecture des 1-cycles est fausse à partir de trois machines.

La troisième extension considérée est l'ajout de zones de stockage entre les machines (Chapitre 7). En effet, cette solution peut être envisagée si le gain en production est important et le coût de mise en place de zones de stockage faible. Dans ce cas, la Conjecture des 1-cycles est valide. La description du cycle optimal nous permet d'établir le gain maximum induit par l'ajout de zones de stockage unitaires entre les machines.

Dans le dernier chapitre (Chapitre 8), nous considérons que les distances inter-machines ne sont plus nécessairement additives. Le robot peut, par exemple, prendre des raccourcis ou accélérer entre deux machines éloignées. Dans ce cas, les permutations pyramidales ne dominent pas les 1-cycles et la Conjecture des 1-cycles est fausse. De plus, trouver le meilleur 1-cycle est un problème NP-complet.

Chapter 5

Remarques sur les cellules circulaires

Dans ce chapitre, nous considérons une cellule robotisée dans laquelle les machines sont disposées en cercle [Figure 5.1] : les pièces entrent dans la cellule et en sortent au même endroit. Le robot central est bidirectionnel et prend la direction qui lui permet de faire le moins de trajets entre les machines. Ainsi, si δ est le temps de trajet entre deux machines consécutives alors la durée du trajet entre M_i et M_j est $\min(|i-j|, m+1-|i-j|)\delta$ unités de temps.

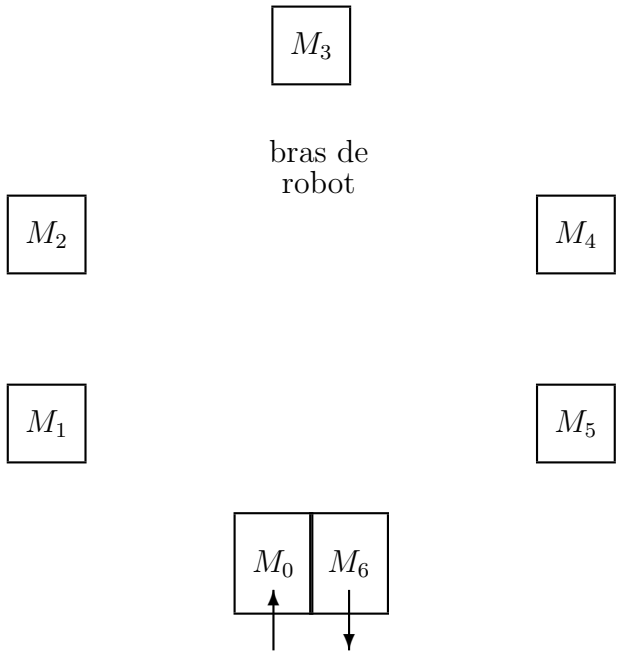

Figure 5.1: Une cellule robotisée circulaire à 5 machines

L'idée de cette configuration est que, si c'est physiquement possible, alors, il peut

être intéressant d'associer l'entrée et la sortie du système pour diminuer les mouvements du robot et pour n'avoir qu'un seul lieu de chargement et de déchargement (arrivée des matières premières et sortie des pièces finies au même endroit).

Dans ce chapitre, nous faisons quelques remarques et conjectures sur les cellules circulaires. Notre propos est d'introduire ce nouveau type de cellules pour de futures recherches. C'est pourquoi, certains résultats partiels ne sont pas approfondis ou sont présentés sous forme de conjecture. De plus, nous nous limitons, dans ce chapitre, aux 1-cycles. Par conséquent, ¡¡ optimal ¿¿ signifie ¡¡ optimal dans l'ensemble des 1-cycles ¿¿. Pour ne pas alourdir les calculs, nous posons que les temps de chargement et de déchargement sont nuls ($\epsilon = 0$).

Nous étudions les cas de deux machines (Section 5.1) puis de trois machines (Section 5.2). Puis, nous présentons quelques résultats pour des cellules à quatre machines et nous extrapolons ces résultats pour des cellules à m machines (Section 5.3) : nous abordons le calcul du gain par rapport à une cellule en ligne et nous faisons quelques remarques sur les cellules circulaires dans le cas régulier équilibré, c'est-à-dire, quand les temps d'usinage sont tous égaux.

5.1 Le cas de deux machines

Considérons d'abord le cas très simple de deux machines. Les temps de cycle des deux 1-cycles sont donnés par :

$$\begin{array}{ll} \pi_0 = (A_0 A_1 A_2) & T(\pi_0) = 3\delta + p_1 + p_2 \ ; \\ \pi_1 = (A_0 A_2 A_1) & T(\pi_1) = 6\delta + \max(0, p_1 - 3\delta, p_2 - 3\delta). \end{array}$$

Théorème 12 *Si $p_1 + p_2 \leq 3\delta$, alors π_0 est optimal, sinon π_1 est optimal.*

Démonstration Si $p_1 + p_2 \leq 3\delta$ alors $T(\pi_0) \leq 6\delta$ et $T(\pi_1) = 6\delta$. Donc $T(\pi_0) \leq T(\pi_1)$. Sinon, $T(\pi_0) \geq \max(6\delta, p_1 + 3\delta, p_2 + 3\delta)$. □

La Conjecture des 1-cycles est vraie dans le cas d'une cellule circulaire à deux machines. En effet, le raisonnement de la preuve du Théorème 4 (page 58) s'applique aussi au cas circulaire.

5.2 Le cas de trois machines

Pour $m = 3$, dans une cellule circulaire, les temps de cycle des six 1-cycles sont donnés par :

$$\begin{array}{rcl} \pi_0 & = & (A_0 A_1 A_2 A_3) \\ & & T(\pi_0) = 4\delta + p_1 + p_2 + p_3 \end{array}$$

$$
\begin{aligned}
\pi_1 &= (A_0A_1A_3A_2) \\
&\quad T(\pi_1) = 8\delta + p_1 + \max(0, p_2 - 4\delta, p_3 - 4\delta - p_1) \\
\pi_2 &= (A_0A_2A_1A_3) \\
&\quad T(\pi_2) = 8\delta + \max(0, p_1 - 4\delta, p_2 - 4\delta, p_3 - 4\delta, \frac{p_1 + p_2 + p_3}{2} - 6\delta) \\
\pi_3 &= (A_0A_2A_3A_1) \\
&\quad T(\pi_3) = 8\delta + p_3 + \max(0, p_1 - 4\delta - p_3, p_2 - 4\delta) \\
\pi_4 &= (A_0A_3A_2A_1) \\
&\quad T(\pi_4) = 12\delta + \max(0, p_1 - 8\delta, p_2 - 8\delta, p_3 - 8\delta) \\
\pi_5 &= (A_0A_3A_1A_2) \\
&\quad T(\pi_5) = 8\delta + p_2 + \max(0, p_1 - 4\delta, p_3 - 4\delta)
\end{aligned}
$$

Théorème 13 *Pour une cellule circulaire à trois machines, l'ensemble* $\{\pi_0, \pi_2, \pi_4\}$ *domine l'ensemble des 1-cycles.*

Démonstration Comme pour le cas de deux machines, il suffit de vérifier, pour chaque instance, que l'un des trois 1-cycles, π_0, π_2 ou π_4, est dominant. La Figure 5.2 indique, pour toutes les instances, le 1-cycle dominant. Nous ne détaillons pas plus la preuve car elle est assez calculatoire.

Par exemple, si $p_1 + p_2 + p_3 \leq 4\delta$ alors $T(\pi_0) \leq 8\delta$ et $T(\pi_i) \geq 8\delta$ pour tout $1 \leq i \leq 5$. □

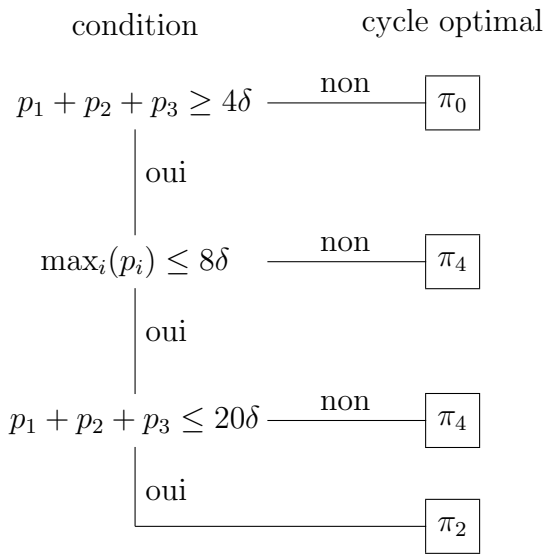

Figure 5.2: Les 1-cycles dominants dans une cellule circulaire à trois machines

Le Théorème 13 indique que, pour une cellule circulaire, les permutations pyramidales ne sont pas dominantes. En effet, π_2 n'est pas une permutation pyramidale et il existe des instances pour lesquelles π_2 domine strictement tous les 1-cycles.

5.3 Et pour quatre machines et plus ?

Nous commençons par étudier les cellules à quatre machines.

Théorème 14 *Pour $m = 4$, il existe seize 1-cycles dominants.*

Ces seize 1-cycles, ordonnés par temps de trajet croissant, sont décrits dans le Tableau 5.1.

Tableau 5.1: 1-cycles dominants pour $m = 4$

trajet	1-cycles optimaux
5δ	$(A_0A_1A_2A_3A_4)$
9δ	$(A_0A_2A_3A_4A_1)$ $(A_0A_4A_1A_2A_3)$ $(A_0A_1A_2A_4A_3)$ $(A_0A_1A_3A_2A_4)$ $(A_0A_2A_1A_3A_4)$
10δ	$(A_0A_2A_4A_1A_3)$
13δ	$(A_0A_3A_4A_2A_1)$ $(A_0A_4A_2A_3A_1)$ $(A_0A_2A_4A_3A_1)$ $(A_0A_3A_2A_4A_1)$ $(A_0A_4A_3A_1A_2)$ $(A_0A_1A_4A_3A_2)$ $(A_0A_2A_1A_4A_3)$ $(A_0A_3A_2A_1A_4)$
15δ	$(A_0A_4A_3A_2A_1)$

Le Théorème 14 signifie que, pour toutes les instances possibles, l'un des seize 1-cycles donnés dans le Tableau 5.1 est optimal et que pour chacun de ces seize 1-cycles, il existe une instance pour laquelle ce cycle domine strictement tous les autres.

Une extrapolation des résultats présentés précédemment permet de proposer la conjecture suivante.

Conjecture Considérons une cellule circulaire à m machines avec $m > 2$.

- Si $\sum p_i \leq 4\delta$, alors le cycle identité $\pi_0 = (A_0A_1 \dots A_m)$ est optimal et son temps de cycle est $T(\pi_0) = (m+1)\delta + \sum p_i$.
- Si $\max_i p_i \geq (3m-1)\delta$, alors la permutation descendante $\pi_d = (A_0A_mA_{m-1} \dots A_1)$ est optimale et son temps de cycle est $T(\pi_d) = 3(m+1)\delta + \max(0, p_i - (3m-1)\delta)$.

Étudions maintenant le gain que peut entraîner le fait d'associer l'entrée et la sortie du système. Pour une instance I, soit $T_l(I)$, le temps de cycle optimal pour une cellule où l'entrée et la sortie sont dissociées et $T_c(I)$ est le temps de cycle optimal pour une cellule circulaire. Le gain, G(I), est

$$G(I) = \frac{T_l(I) - T_c(I)}{T_l(I)}.$$

Le Tableau 5.2 présente une borne inférieure au gain pour une cellule à deux machines.

Tableau 5.2: Gain pour une cellule à deux machines

I	$T_l(I)$	$T_c(I)$	$G(I)$
$p_1 + p_2 \to 0$	6δ	3δ	$\to 1/2$
$p_1 + p_2 \leq 2\delta$	$6\delta + p_1 + p_2$	$3\delta + p_1 + p_2$	$3/8$
$2\delta \leq p_1 + p_2 \leq 3\delta$	8δ	$3\delta + p_1 + p_2$	$1/4$
$3\delta \leq p_1 + p_2$ $et \max(p_i) \leq 3\delta$	8δ	6δ	$1/4$
$3\delta \leq \max(p_i) \leq 4\delta$	8δ	$\max(p_i) + 3\delta$	$1/8$
$4\delta \leq \max(p_i)$	$\max(p_i) + 4\delta$	$\max(p_i) + 3\delta$	$\delta/(\max(p_i) + 4\delta)$
$\max(p_i) \to \infty$	$\max(p_i) + 4\delta$	$\max(p_i) + 3\delta$	$\to 0$

On remarque que plus les temps d'usinage sont petits par rapport aux temps de trajet et plus le gain est important. Par contre, quand les temps d'usinage sont grands, le layout a moins d'influence sur le temps de cycle et le gain tend vers zéro.

On peut généraliser cette remarque à des cellules à m machines. En effet quand les temps d'usinage tendent vers zéro, le gain tend vers 50% et quand les temps d'usinage tendent vers l'infini, le gain tend vers zéro.

Considérons maintenant le cas régulier équilibré pour une cellule circulaire : $p_i = p$ pour $i = 1, 2 \ldots m$. Dans ce cas, la cardinalité de l'ensemble des 1-cycles dominants est décrite dans le Tableau 5.3 (nous ne décrivons pas la preuve qui est une étude de cas) :

Tableau 5.3: Nombre de 1-cycles dominants pour le cas régulier équilibré

m	nombre de 1-cycles dominants
2	2
3	3
4	4

Pour m pair soit π, le 1-cycle suivant

$$\pi = (A_0 A_2 A_4 \ldots A_m A_1 A_3 A_5 \ldots A_{m-1})$$

et pour m impair

$$\pi = (A_0 A_2 A_4 \ldots A_{m-1} A_1 A_3 A_5 \ldots A_m).$$

Nous pensons que le temps de cycle de π est

$$T(\pi) = (m+1)\delta + \frac{2\alpha - 1}{2\alpha - 3} \max(0, p - (m+1)\delta)$$

où $m = 2\alpha$ si m est pair et $m = 2\alpha - 1$ si m est impair. Il semblerait que pour des temps d'usinage égaux, si $0 \leq p \leq \frac{m+1}{m}$ alors π_0 est optimal et pour un intervalle de la forme $\frac{m+1}{m} \leq p \leq$?? (la deuxième borne de l'intervalle n'est pas connue), π est optimal.

On peut remarquer que pour exécuter π_0, le robot fait une fois le tour complet de la cellule, pour exécuter π, il le fait deux fois, et lors de l'exécution de π_d, le robot parcourt trois fois la cellule. De plus, π ressemble à l'un des 1-cycles dominants pour le cas régulier équilibré (Chapitre 4, Section 4.2) en ¡¡ retournant ¿¿ la partie descendante.

5.4 Conclusion

Dans ce chapitre, nous avons proposé quelques remarques sur les 1-cycles optimaux dans des cellules circulaires. Pour ce type de cellules, nous ne connaissons pas de résultats plus détaillés et nous pensons qu'une étude approfondie peut s'avérer très intéressante. Ce chapitre constitue donc une ouverture vers d'autres configurations (ou layout) de cellules robotisées.

Chapter 6

Remarques sur les problèmes de type HSP

Dans ce chapitre, nous considérons les problèmes de type *Hoist Scheduling Problem* (HSP). Ces problèmes sont plus contraints que les problèmes d'ordonnancement dans des cellules robotisées. L'environnement est le même, mais le temps pendant lequel une pièce peut rester sur une machine possède une borne supérieure. Ce temps est donc compris dans un intervalle. Le HSP apparaît dans les processus chimiques comme la galvanoplastie [64]. Crama et van de Klundert ont prouvé que pour un HSP mono-produit, trouver le meilleur 1-cycle est un problème fortement NP-complet même si les durées de transfert sont additives, c'est-à-dire qu'elles vérifient l'égalité triangulaire [27]. Nous étudions l'extension au HSP de certaines propriétés intéressantes des cellules robotisées : existence d'ordonnancements actifs optimaux, dominance des permutations pyramidales et Conjecture des 1-cycles. [1]

6.1 Présentation du HSP

Le HSP cyclique mono-produit est généralement défini comme un problème d'ordonnancement dans un flow-shop robotisé.

Le vocabulaire du HSP est différent de celui des cellules robotisées. Les machines sont remplacées par des *cuves* qui contiennent des *bains de traitement*. Les pièces à traiter sont montées sur des *porteurs* et le bras de robot est remplacé par un robot qui se déplace sur un *rail* [Figure 6.1].

Les porteurs sont pris par le robot au lieu de chargement, noté IN ou M_0. Ils doivent passer dans toutes les cuves puis quitter la ligne au lieu de déchargement noté OUT

[1] les résultat présentés dans ce chapitre font partie d'un article publié dans les actes de la conférence IEPM'97 [13].

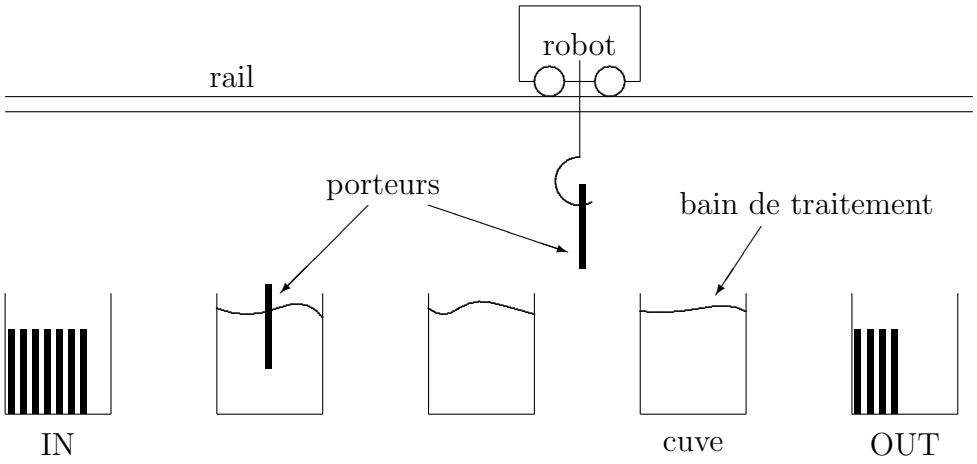

Figure 6.1: Représentation d'un HSP à trois cuves

ou M_{m+1}. Le temps pendant lequel une pièce doit rester dans une cuve admet une borne inférieure et une borne supérieure. Par exemple, les pièces ne doivent pas rester trop longtemps dans certains bains chimiques. Ainsi à chaque cuve, notée $M_i(i = 1, 2 \ldots m)$, est associé un intervalle ou fenêtre de temps : la durée de trempe d'un porteur dans une cuve doit appartenir à cet intervalle. Différentes formes de lignes ont été étudiées. Ces formes sont désignées par des lettres : U, I ou O si le rail forme un rectangle fermé. Nous considérons les lignes en I.

Il existe une très vaste littérature sur le HSP. Nous ne la détaillons pas dans ce document. Un état de l'art sur ce type de problème peut être trouvé dans [11] ou dans [59].

La cellule robotisée telle que nous la considérons dans les autres chapitres de ce document peut être définie comme un HSP cyclique mono-produit, mono-robot sur une ligne en I avec des durées opératoires non bornées.

6.2 Ordonnancements actifs

Un ordonnancement est actif si le robot exécute la prochaine activité dès que possible sans prendre en compte les mouvements futurs. Ainsi, les seules attentes du robot se produisent à des machines occupées lorsque le robot vide attend que la pièce soit prête afin d'exécuter l'activité suivante. Pour les cellules robotisées, il existe toujours un ordonnancement actif qui est optimal.

Si seuls les ordonnancements actifs sont considérés, alors la donnée de la séquence d'activités est suffisante pour décrire l'ordonnancement complet des mouvements du robot. Par contre, si on ne se restreint pas aux ordonnancements actifs, alors il

faut préciser, en plus de la séquence d'activités, les temps d'attente du robot aux machines ou les instants de début des activités.

Pour le HSP, il n'existe pas toujours un ordonnancement actif qui est optimal. En effet, pour l'instance suivante on ne peut pas trouver un ordonnancement actif optimal. On considère la ligne à deux cuves telle que :

- le temps de trajet entre deux cuves consécutives est égal à 1 ($\delta_i = 1$; $i = 0, 1 \ldots m$) ;
- les temps de chargement et de déchargement des cuves sont égaux à 0 ($\epsilon_i^u = \epsilon_{i+1}^l = 0$; $i = 0, 1 \ldots m$) ;
- la fenêtre de temps de M_1 est $[1, 4]$, c'est-à-dire qu'une pièce doit rester sur M_1 au moins 1 unité de temps et au plus 4 unités de temps ;
- la fenêtre de temps de M_2 est $[5, 5]$.

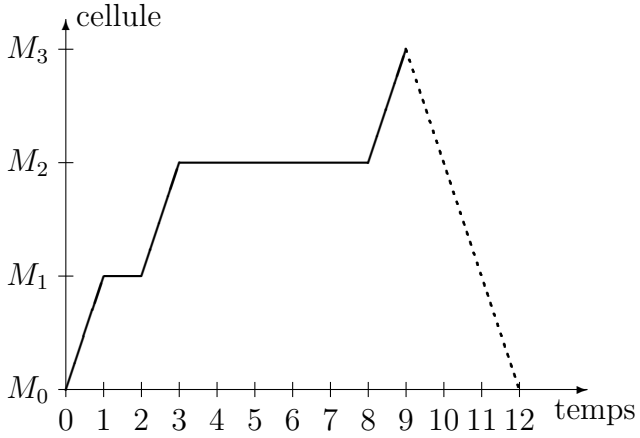

Figure 6.2: Le 1-cycle $(A_0A_1A_2)$ dure 12 unités de temps

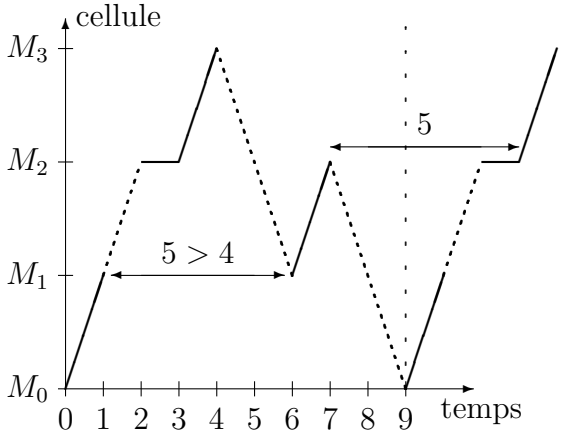

Figure 6.3: L'ordonnancement actif de $(A_0A_2A_1)$ n'est pas admissible

Il existe deux 1-cycles : $(A_0A_1A_2)$ et $(A_0A_2A_1)$. Le premier dure 12 unités de temps et est admissible et actif mais non optimal [Figure 6.2]. L'ordonnancement

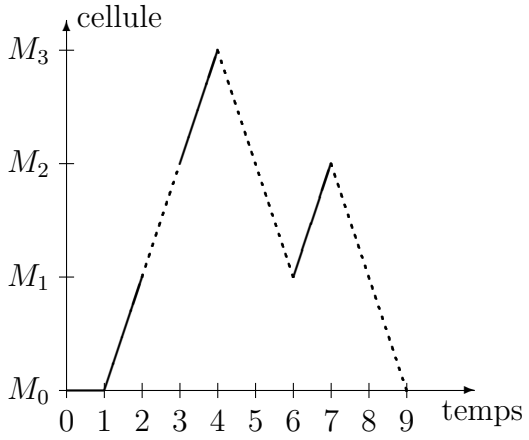

Figure 6.4: Cet ordonnancement non actif de $(A_0A_2A_1)$ est admissible et optimal

actif de $(A_0A_2A_1)$ n'est pas admissible car la pièce resterait au moins 5 unités de temps dans la cuve M_1, ce qui est supérieur au temps de trempe autorisé dans M_1 [Figure 6.3]. Il existe un ordonnancement optimal de $(A_0A_2A_1)$ qui dure 9 unités de temps et qui n'est pas actif car le robot attend une unité de temps avant d'exécuter A_0 alors que la pièce sur M_0 est disponible [Figure 6.4].

Ainsi, on ne peut pas se restreindre aux ordonnancements actifs et la donnée d'un cycle comme une séquence d'activités n'est pas suffisante pour déterminer l'ordonnancement des mouvements du robot.

6.3 Permutations pyramidales

Dans [27], les auteurs ont prouvé que, pour des cellules robotisées, les permutations pyramidales dominent les 1-cycles. Pour le HSP, les permutations pyramidales ne dominent pas les 1-cycles. Dans le cas de deux cuves, tous les 1-cycles sont des permutations pyramidales. À partir de trois cuves, certains 1-cycles ne sont pas des permutations pyramidales. Considérons par exemple l'instance suivante à trois cuves :

- le temps de trajet entre deux cuves consécutives est égal à 1 ($\delta_i = 1$; $i = 0, 1 \dots m$) ;
- les temps de chargement et de déchargement des cuves sont égaux à 0 ($\epsilon_i^u = \epsilon_{i+1}^l = 0$; $i = 0, 1 \dots m$) ;
- la fenêtre de temps de M_1 est $[6, 6]$;
- la fenêtre de temps de M_2 est $[1, 1]$;
- la fenêtre de temps de M_3 est $[6, 6]$.

On peut vérifier que, pour une telle cellule, les seuls 1-cycles admissibles sont ceux qui contiennent le motif A_1A_2. Dans le cas contraire, une pièce devrait rester

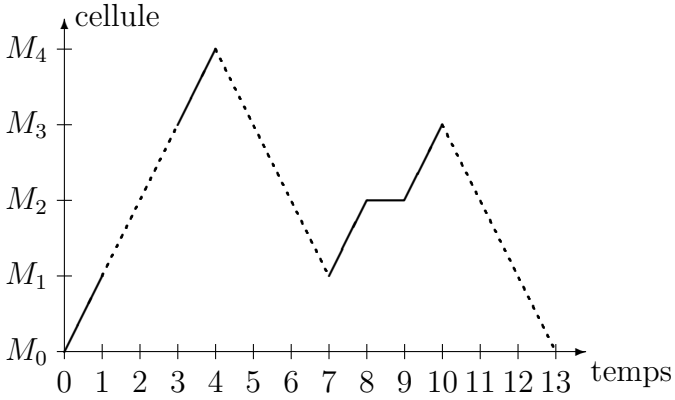

Figure 6.5: Le 1-cycle $(A_0A_3A_1A_2)$ dure 13 unités de temps

aux moins 4 unités de temps ce qui est supérieur à la valeur autorisée sur M_2 qui est de 1 unité de temps. Ainsi, les deux 1-cycles admissibles sont $(A_0A_1A_2A_3)$ et $(A_0A_3A_1A_2)$. Les temps de cycle de ces 1-cycles sont donnés par

$$\begin{aligned} T(A_0A_1A_2A_3) &= 21 \ ; \\ T(A_0A_3A_1A_2) &= 13. \end{aligned}$$

La Figure 6.5 décrit l'exécution du 1-cycle $(A_0A_3A_1A_2)$. Le meilleur 1-cycle est donc $(A_0A_3A_1A_2)$ et ce n'est pas une permutation pyramidale. Ainsi, les permutations pyramidales ne dominent pas les 1-cycles.

6.4 Conjecture des 1-cycles

Dans [55], les auteurs présentent un contre-exemple à la Conjecture des 1-cycles pour un HSP contenant 12 cuves. Pour le HSP, la conjecture des 1-cycles est en fait fausse à partir de trois cuves. En effet, pour l'instance à trois cuves suivante tous les 1-cycles sont dominés par un 2-cycle :

- le temps de trajet entre deux cuves consécutives est égal à 1 ($\delta_i = 1$; $i = 0, 1 \ldots m$) ;
- les temps de chargement et de déchargement des cuves sont égaux à 0 ($\epsilon_i^u = \epsilon_{i+1}^l = 0$; $i = 0, 1 \ldots m$) ;
- la fenêtre de temps de M_1 est $[2, 4]$;
- la fenêtre de temps de M_2 est $[1, 4]$;
- la fenêtre de temps de M_3 est $[1, 4]$.

Pour une telle cellule, il existe $3! = 6$ 1-cycles différents [66]. Seul l'un d'eux est admissible et son temps de cycle est $T(A_0A_1A_2A_3) = 12$. Les cinq autres 1-cycles ne sont pas admissibles car, même si on ne se limite pas aux ordonnancements actifs, une pièce resterait trop longtemps dans l'une des cuves. Par exemple pour

le 1-cycle $(A_0A_3A_1A_2)$, l'activité A_0 sépare A_2 et A_3. Par conséquent, entre la fin de A_2 (un porteur est déposé dans M_3) et le début de A_3 (le porteur est pris de M_3), il s'écoule 6 unités de temps (durée d'un trajet de M_3 à M_0 puis de M_0 à M_3). Le porteur reste dans M_3 un temps supérieur à 4. Donc, le 1-cycle $(A_0A_3A_1A_2)$ n'est pas admissible. Le même phénomène se répète pour les autres 1-cycles.

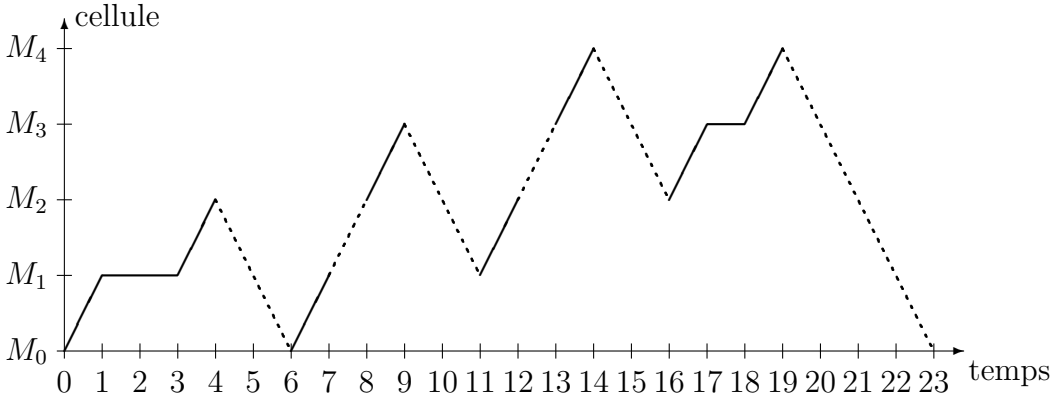

Figure 6.6: Le 2-cycle $(A_0A_1A_0A_2A_1A_3A_2A_3)$ dure 23 unités de temps

De plus, on peut remarquer que $T(A_0A_1A_0A_2A_1A_3A_2A_3) = 23$ [Figure 6.6]. Donc, $(A_0A_1A_0A_2A_1A_3A_2A_3)$ est un 2-cycle qui domine tous les 1-cycles admissibles $(23/2 = 11.5 < 12)$.

La Conjecture des 1-cycles est donc fausse à partir de trois cuves pour les problèmes de type HSP.

6.5 Conclusion

Dans ce chapitre, nous avons vu que les propriétés intéressantes des cellules robotisées ne pouvaient pas être étendues au HSP : il n'existe pas toujours un ordonnancement actif qui est optimal, les permutations pyramidales ne sont pas dominantes et la Conjecture des 1-cycles n'est pas vraie pour trois cuves et plus.

Les quelques exemples présentés dans ce chapitre montrent que le HSP est un problème beaucoup plus difficile à résoudre que les problèmes d'ordonnancement dans les cellules robotisées même si, dans leur description, les deux problèmes sont très proches. En effet des propriétés très intéressantes des cellules robotisées ne se retrouvent pas dans le HSP.

Chapter 7

Ajout d'espaces de stockage

Dans ce chapitre, nous étudions l'effet, sur le temps de cycle, de l'introduction dans l'atelier de production robotisé de zones de stockage aussi appelées buffers. En effet, dans un atelier en ligne sans espace de stockage entre les machines, les mouvements du robot sont contraints par le fait qu'il ne peut transporter une pièce sur une machine occupée. En ajoutant des stocks intermédiaires, nous donnons donc plus de liberté au robot. [1,2]

Dans la première section (Section 7.1), nous exposons les paramètres qui décrivent la cellule. Puis, nous identifions le cycle de production optimal pour une cellule robotisée avec buffers (Section 7.2). Ceci nous permet d'établir le gain maximum entraîné par l'ajout de buffers unitaires entre les machines (Section 7.3). Nous concluons ce chapitre (Section 7.4) en montrant les similitudes entre les cellules robotisées avec des espaces de stockage de capacités unitaires entre les machines et d'autres configurations de cellules de production.

Ces ateliers en ligne robotisés avec espaces de stockage ont surtout été étudiés pour la production de différents types de pièces avec comme objectif la minimisation du temps total de production (*makespan*). Par exemple, Kise a démontré que dans un flow-shop robotisé à deux machines, le robot peut être considéré comme une troisième machine intermédiaire [51]. Dans ce cas, même pour des temps de transport identiques pour toutes les pièces et un buffer intermédiaire infini, le problème est NP-difficile. King et al. [50] proposent un algorithme de séparation et évaluation pour résoudre ce problème dans une cellule à m machines avec des buffers intermédiaires infinis. Agnetis [3] a étudié certains cas polynomiaux de production par lots dans une cellule où les deux machines sont équipées d'un dispositif qui leur permet d'échanger la pièce qu'elles viennent d'usiner avec la pièce apportée par le

[1] Les travaux présentés dans ce chapitre ont été effectués en collaboration avec Cyrille Gueguen de l'École Centrale de Paris.

[2] Les résultats présentés dans la Section 7.2 ont été publiés dans les actes de la conférence ECCO IX [31]. Ceux présentés dans la Section 7.3 ont été publiés dans la revue JESA [18].

robot.

7.1 Description de l'atelier

Dans ce travail, nous nous intéressons à l'ajout, après chaque machine de la cellule, de zones de stockage de la plus petite taille possible, c'est-à-dire unitaires. Nous supposons que le transfert d'une pièce entre une machine et le buffer de cette machine ne nécessite pas le robot. Dans ce cas, le robot peut apporter une pièce sur une machine occupée dont le buffer est vide. Il attend alors la fin de l'usinage pour déposer la pièce.

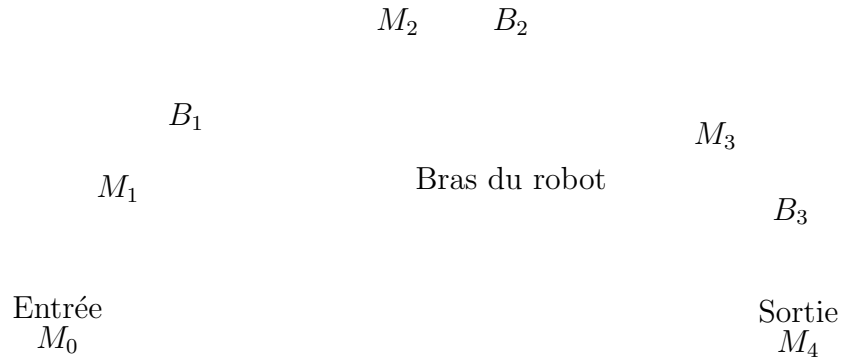

Figure 7.1: Cellule de production avec buffers

On considère la cellule de production robotisée classique dans laquelle on rajoute un buffer de capacité unitaire derrière chaque machine (Figure 7.1). Les pièces à produire sont toutes identiques. Dès qu'une pièce a fini d'être usinée sur la machine M_h, elle tombe instantanément dans le buffer B_h si celui-ci est vide. Sinon, elle reste sur la machine M_h jusqu'à ce que le buffer B_h se libère. Le couple (M_h, B_h) est appelé unité d'usinage et est noté U_h. Chaque machine et chaque buffer est de capacité unitaire.

Le processus de production est gouverné par les paramètres $p_h (h = 1, 2 \dots m)$, temps d'usinage sur la machine M_h, δ, durée du trajet entre deux unités de production adjacentes, et ϵ, temps de chargement ou de déchargement d'une unité de production. Les temps de trajet du robot sont additifs, c'est-à-dire pour se déplacer entre deux machines, le robot passe par toutes les machines intermédiaires. Lorsque le robot charge l'unité U_h, il dépose une pièce sur la machine M_h (si M_h est libre) et lorsqu'il la décharge, il prend une pièce du buffer B_h (si B_h contient une pièce).

Les résultats présentés dans ce chapitre peuvent être étendus à un système où les temps de chargement ou de déchargement de l'unité U_h sont respectivement ϵ_h^l et ϵ_h^u, et où le temps de transport de l'unité U_h à l'unité U_{h+1} est δ_h.

7.2 Cycle optimal de production

On étend la notion d'activité aux cellules de production avec buffers. L'activité $A_h(h = 0, 1 \dots m)$ est composée de la séquence d'actions suivante :
- le robot prend une pièce de l'unité U_h ;
- le robot transporte cette pièce de l'unité U_h à l'unité U_{h+1} ;
- le robot dépose la pièce sur l'unité U_{h+1}.

L'état du système, à l'instant t est défini par le vecteur $V(t)$ tel que chaque composante de $V(t)$ est un couple. Ainsi, la h-ème composante de $V(t)$, notée $V_h(t)$, est $(m_h(t), b_h(t))$, où $m_h(t)$ est égal à 1 si, à l'instant t, la machine M_h est occupée et à 0 sinon et $b_h(t)$ est égal à 1 si, à l'instant t, le buffer B_h est occupé et est égal à 0 sinon.

Lemme 10 *Le temps de cycle, $T(C_k)$, de tout k-cycle C_k est borné par*

$$T(C_k) \geq k \max(2(m+1)(\delta+\epsilon), \max_h(p_h) + \epsilon).$$

Démonstration Soit C_k un k-cycle. L'activité $A_h(h = 0, 1 \dots m)$ dure $\delta + 2\epsilon$ unités de temps. On exécute donc k fois toutes les activités A_h en $k(m+1)(\delta+2\epsilon)$ unités de temps. On considère, sans perte de généralité, que C_k commence par l'activité A_0. Entre deux occurrences de l'activité A_h, le robot doit parcourir au moins une fois le trajet de U_{h+1} à U_h. Entre la dernière occurrence de A_h et la fin du cycle, le robot doit retourner en M_0 ce qui rajoute un trajet de U_{h+1} à U_h. Ainsi, les trajets de U_{h+1} à U_h durent au moins $k(m+1)\delta$ unités de temps. On a donc

$$T(C_k) \geq 2k(m+1)(\delta+\epsilon).$$

De plus, soit j un indice tel que $p_j = \max_h(p_h)$. Durant l'exécution d'un k-cycle, k pièces doivent être usinées sur la machine M_j. Le temps écoulé entre deux chargements de U_j doit donc être au moins $p_j + \epsilon$. Ainsi, $T(C_k) \geq k(\max_h(p_h) + \epsilon)$. Ce qui conclut la démonstration. □

Théorème 15 *Dans une cellule de production avec des espaces de stockage de capacités unitaires derrière les machines, le taux maximum de production est obtenu en exécutant le 1-cycle $Id = (A_0 A_1 ... A_m)$. La durée $T_{buff}(I)$ du cycle Id, pour l'instance I, est donnée par*

$$T_{buff}(I) = \max(2(m+1)(\delta+\epsilon), \max_h(p_h) + \epsilon). \quad (7.1)$$

Démonstration Au début du cycle Id, l'unité $U_h(h = 1, 2 \dots m)$, c'est-à-dire M_h ou B_h, contient une pièce. Soient $T_1 = 2(m+1)(\delta+\epsilon), T_2 = \max_h(p_h) + \epsilon$ et $T = \max(T_1, T_2)$.

Cas 1 $T = T_1$

T est le temps nécessaire au robot pour exécuter le 1-cycle *Id* sans avoir à attendre à une machine. Le temps écoulé entre deux passages à l'unité $U_h(h = 1, 2 \dots m)$ est $T - \epsilon$. Or $T - \epsilon \geq p_h$. Donc, la pièce sur M_h a fini d'être usinée et est dans le buffer B_h lorsque le robot arrive en U_h. Ainsi, le robot peut prendre la pièce dans B_h sans avoir à attendre. Le cycle *Id* est alors admissible et son temps de cycle est T_1.

Cas 2 $T = T_2$

Le Lemme 10 indique que, pour tout k-cycle C_k, $T(C_k) \geq k(p_h + \epsilon)$. Soit j le plus petit indice qui satisfait $T = p_j + \epsilon$. Soit $\pi = (A_j \dots A_m A_0 \dots A_{j-1})$ le cycle équivalent à *Id* commençant par A_j. On définit le vecteur d'état $V(t_j)$ où t_j est l'instant de début de l'activité A_j lors de la première exécution du cycle π,

- $V_j(t_j) = (1, 1)$ et
- $V_i(t_j) = (0, 1)$ pour tout $i \neq j$.

On exécute A_j une première fois en prenant la pièce de B_j. Les machines $M_h(h \neq j)$ sont atteintes la deuxième fois au bout d'un temps égal à $p_j + \epsilon \geq p_h + \epsilon$. Le cycle *Id* est donc admissible et son temps de cycle est égal à T_2.

Le Lemme 10 permet alors d'affirmer que le taux maximum de production est obtenu en exécutant le cycle *Id*. □

Le Théorème 15 indique que la Conjecture des 1-cycles est vérifiée pour une cellule de production munie d'un buffer de capacité unitaire derrière chaque machine.

7.3 Gains

Pour une instance I donnée, soit $T_{opt}(I)$ le meilleur temps de cycle des 1-cycles dans un système sans buffer et $T_{buff}(I)$ le meilleur temps de cycle dans un système muni d'un buffer de capacité unitaire derrière chaque machine. Pour une instance I, le gain $G(I)$ est défini par :

$$G(I) = \frac{T_{opt}(I) - T_{buff}(I)}{T_{opt}(I)}.$$

Dans le pire des cas, pour des instances artificielles, le gain est égal à zéro. Par exemple, on considère une instance I, telle que les temps d'usinage sont négligeables par rapport aux temps de trajet ou de chargement/déchargement. Dans le système sans buffer, on veut minimiser le trajet. Le cycle optimal est donc l'identité *Id* :

$$T_{opt}(I) = T(Id) = 2(m + 1)(\delta + \epsilon) + \sum_{h=1}^{m} p_h = 2(m + 1)(\delta + \epsilon).$$

Pour un système avec buffer, on a $T_{buff}(I) = 2(m+1)(\delta+\epsilon)$ ce qui donne $G(I) = 0$.

On s'intéresse donc au gain maximum G, défini par

$$G = \inf\{g \mid G(I) \leq g \text{ pour toutes les instances } I\}.$$

On veut évaluer le gain, qui se trouve dans l'intervalle $[0, G]$, afin de décider s'il est rentable de rajouter un buffer derrière chaque machine. On simule les différentes instances pour avoir une idée du gain moyen et des intervalles de gains possibles.

Figure 7.2: Nombre d'instances pour lesquelles $100 \times \lceil G(I) \rceil$ a une valeur donnée

La Figure 7.2 décrit les valeurs de la partie entière supérieure de $100 \times G(I)$ dans une cellule composée de trois machines pour toutes les instances possibles telles que

- δ et p_h prennent toutes les valeurs entières entre 1 et v ;
- $\epsilon = 0$;

pour différentes valeurs de v. Par exemple, le point P qui est sur la courbe "$v = 20$" signifie qu'il existe environ 10 000 instances I différentes, sur les $v^{m+1} = 20^4$ instances possibles décrites ci-dessus, telles que la partie entière supérieure de $G(I)$ soit égale à 16%. On peut remarquer que le gain est significatif puisque $G(I)$ est presque toujours supérieur à 12%.

Le théorème suivant donne la valeur exacte du gain maximum que l'on peut obtenir, sur toutes les instances, en ajoutant au système une zone de stockage de capacité unitaire derrière chaque machine dans un système à m machines. Il donne également des familles d'instances pour lesquelles ce gain est atteint.

Théorème 16 *Le gain G satisfait*

$$G = \frac{2}{m+3} \quad \textit{pour } m \leq 3 \textit{ et}$$

$$G = \frac{1}{2} - \frac{1}{2m} \quad \textit{pour } m \geq 3.$$

Pour des raisons de clarté, la démonstration de ce théorème est décomposée en différentes étapes.

Lemme 11 *Pour une instance I donnée, le temps de cycle optimal, $T_{opt}(I)$, pour un système sans buffer vérifie*

$$T_{opt}(I) \leq T_{buff}(I) + max(4\delta + 3\epsilon, 2(m-1)\delta). \tag{7.2}$$

Démonstration Si $4(m-1)\delta + (2m-1)\epsilon \leq 2(m+1)(\delta+\epsilon)$ alors, en comparant les équations (2.5) (page 40) et (7.1), on obtient

$$T(\pi_d) \leq T_{buff}(I) + 4\delta + 3\epsilon.$$

Sinon, on a $4(m-1)\delta + (2m-1)\epsilon \geq 2(m+1)(\delta+\epsilon)$ et

$$T(\pi_d) \leq T_{buff}(I) + 4(m-1)\delta + (2m-1)\epsilon - 2(m+1)(\delta+\epsilon) + 4\delta + 3\epsilon$$

ce qui implique que

$$T(\pi_d) \leq T_{buff}(I) + 2(m-1)\delta.$$

De plus, comme, $T_{opt}(I) \leq T(\pi_d)$, on a

$$T_{opt}(I) \leq T_{buff}(I) + max(4\delta + 3\epsilon, 2(m-1)\delta).$$

□

Lemme 12 *Le gain G vérifie*

$$G \leq \frac{2}{m+3} \quad \textit{pour } m \leq 3 + \frac{3\epsilon}{2\delta} \textit{ et}$$

$$G \leq \frac{1}{2} - \frac{1}{2m} \quad \textit{pour } m \geq 3 + \frac{3\epsilon}{2\delta}.$$

Démonstration En multipliant les deux membres de l'inégalité (7.2) par $2(m+1)(\delta+\epsilon)$, on obtient

$$2(m+1)(\delta+\epsilon)T_{opt}(I) \leq 2(m+1)(\delta+\epsilon)[T_{buff}(I) + max(4\delta + 3\epsilon, 2(m-1)\delta)].$$

Or, l'équation (7.1) indique que $T_{buff}(I) \geq 2(m+1)(\delta+\epsilon)$. Donc, en reportant dans l'équation précédente, on obtient

$$2(m+1)(\delta+\epsilon)T_{opt}(I) \leq T_{buff}(I)[2(m+1)(\delta+\epsilon) + \max(4\delta+3\epsilon, 2(m-1)\delta)]$$

ce qui donne

$$1 - \frac{T_{buff}(I)}{T_{opt}(I)} \leq 1 - \frac{2(m+1)(\delta+\epsilon)}{2(m+1)(\delta+\epsilon) + \max(4\delta+3\epsilon, 2(m-1)\delta)}.$$

Or, l'inégalité $4\delta + 3\epsilon \geq 2(m-1)\delta$ est équivalente à ($\delta > 0$)

$$m \leq 3 + \frac{3\epsilon}{2\delta}.$$

Donc, si $m \leq 3 + \frac{3\epsilon}{2\delta}$ alors

$$\begin{aligned} 1 - \frac{T_{buff}(I)}{T_{opt}(I)} &\leq \frac{4\delta+3\epsilon}{2(m+3)\delta + (2m+5)\epsilon} \\ &\leq \frac{2}{m+3} - \frac{(m+1)\epsilon}{2(m+3)^2\delta + (2m+5)(m+3)\epsilon} \\ &\leq \frac{2}{m+3}. \end{aligned}$$

Si $m \geq 3 + \frac{3\epsilon}{2\delta}$ alors

$$\begin{aligned} 1 - \frac{T_{buff}(I)}{T_{opt}(I)} &\leq \frac{(m-1)\delta}{2m\delta + (m+1)\epsilon} \\ &\leq \frac{(m-1)\delta}{2m\delta} \\ &\leq \frac{1}{2} - \frac{1}{2m}. \end{aligned}$$

□

Lemme 13 *Soit F_1 une famille d'instances telles que $\epsilon = 0$ et $\max_h(p_h) = 2(m+1)\delta$. Pour I un élément de F_1 et pour $m \leq 3$, on a*

$$G(I) = \frac{2}{m+3}.$$

Démonstration Soit I un élément de F_1. Pour $m \leq 3$, on a

$$T(\pi_d) = 4\delta + \max_h(p_h).$$

Or, on sait, d'après le Théorème 1 (page 38) que $T_{opt}(I) \geq 4\delta + \max_h(p_h)$. Donc π_d est optimale. De plus, $T_{buff}(I) = \max_h(p_h)$ donc, pour I un élément de F_1, on a

$$\begin{aligned} G(I) &= \frac{4\delta}{4\delta + 2(m+1)\delta} \\ &= \frac{2}{m+3}. \end{aligned}$$

□

Lemme 14 *Soit F_2 la famille d'instances telles que*
- *$\epsilon = 0$,*
- *$p_h = 2(m+1)\delta$, quel que soit $h = 1, 2 \ldots m$.*

Pour I un élément de F_2 et pour $m \geq 3$, on a

$$\frac{T(\pi_d) - T_{buff}(I)}{T(\pi_d)} = \frac{1}{2} - \frac{1}{2m}.$$

Démonstration Soit I, un élément de F_2, on a , $T_{buff}(I) = 2(m+1)\delta$ et $T(\pi_d) = 4m\delta$ ce qui implique

$$\frac{T(\pi_d) - T_{buff}(I)}{T(\pi_d)} = \frac{1}{2} - \frac{1}{2m}.$$

□

Pour démontrer que, pour une instance I de F_2, $G(I) = 1/2 - 1/2m$, il suffit de démontrer que π_d est optimale pour toute instance I de F_2 dans un système sans buffer. Le lemme suivant est une conséquence immédiate des Théorèmes 9 et 10, pages 95 et 97.

Lemme 15 *Pour I, un élément de F_2, la permutation π_d est optimale dans un système sans buffer.*

Démonstration du Théorème 16

Étape 1 Les Lemmes 12 et 13 indiquent que pour $m \leq 3$,

$$G \leq \frac{2}{m+3}$$

et que pour I, une instance de la famille F_1, on a

$$G(I) = \frac{2}{m+3}.$$

On peut donc conclure que pour $m \leq 3$,

$$G = \frac{2}{m+3}.$$

Étape 2 Le Lemme 12 indique que pour $3 \leq m \leq 3 + \frac{3\epsilon}{2\delta}$,

$$G \leq \frac{2}{m+3}.$$

Or, dans ce cas,

$$\frac{1}{2} - \frac{1}{2m} \geq \frac{2}{m+3}.$$

Donc, pour $3 \leq m \leq 3 + \frac{3\epsilon}{2\delta}$, on a

$$G \leq \frac{1}{2} - \frac{1}{2m}.$$

Étape 3 Le Lemme 12 implique que, pour $m \geq 3$,

$$G \leq \frac{1}{2} - \frac{1}{2m}$$

et les Lemmes 14 et 15 indiquent que, pour $m \geq 3$ et pour I, une instance de la famille F_2, on a,

$$G(I) = \frac{1}{2} - \frac{1}{2m}.$$

Donc, pour $m \geq 3$, on a $G = \frac{1}{2} - \frac{1}{2m}$. □

Le temps de cycle, $T_{opt}(I)$, dans l'expression du gain ne concerne que les 1-cycles. Les résultats précédents peuvent être partiellement étendus aux k-cycles. En effet, seul le Lemme 15 ne peut pas être étendu aux k-cycles puisqu'il dépend de la validité de la Conjecture des 1-cycles pour le cas régulier équilibré. Or ce problème est encore ouvert pour $m \geq 5$. Donc, pour $m \geq 5$, on peut seulement affirmer que

$$G \leq \frac{1}{2} - \frac{1}{2m}.$$

Il est à noter que le minimum de gain est atteint pour 3 machines. Pour $m = 2$, le gain est inférieur à 40%, pour $m = 3$, il est inférieur à 33,3 %. Lorsque $m \geq 4$, pour certaines instances, le gain augmente asymptotiquement vers 50% si on se limite aux 1-cycles. Ces résultats montrent qu'il peut être très rentable de rajouter un buffer derrière chaque machine.

7.4 Conclusion

Dans ce chapitre, nous avons étudié l'effet de l'ajout de zones de stockage de capacité unitaire après chaque machine. Nous avons démontré que le cycle optimal

de production est l'identité et que, par conséquent, la Conjecture des 1-cycles est vraie pour un tel système. Nous avons ensuite établi le gain maximum entraîné par l'ajout de buffers unitaires aux machines.

Ces résultats sur le gain maximum peuvent être étendus à des configurations du système différentes. Par exemple, dans [71] les auteurs considèrent un système sans zone de stockage où le robot possède deux pinces qui lui permettent de transporter deux pièces à la fois. Comme pour le système avec zones de stockage, le cycle optimal est l'identité.

En fait, pour les deux systèmes, les mouvements optimaux du robot sont les mêmes : il peut arriver chargé à une machine qui contient une pièce. Les buffers des machines ou la deuxième pince du robot ne servent en fait qu' permettre l'échange d'une pièce entre la machine et le robot. Les ressources bloquantes restent les machines qui ne peuvent, dans les deux cas, usiner qu'une pièce à la fois et le robot dont le temps de déplacement n'est pas négligeable.

Pour les mêmes raisons, le cycle optimal est aussi l'identité dans des cellules avec, entre les machines, des zones de stockage de capacités supérieures ou égales à un.

Les résultats sur le gain sont donc identiques pour les systèmes avec :
- des zones de stockage unitaires aux machines, ou
- des zones de stockage de capacités supérieures ou égales à un, ou
- un robot avec deux pinces, ou
- des dispositifs d'échange de pièces entre le robot et les machines (étudiés dans [3] pour le cas non cyclique).

Chapter 8

Généralisation des durées de transfert

Dans les chapitres précédents, les durées de transfert étaient additives et la vitesse du robot constante. Ainsi, les temps de transfert entre les machines vérifiaient l'égalité triangulaire. Dans ce chapitre, nous levons cette contrainte et autorisons des temps de transfert inter-machines quelconques. Ceci se produit, par exemple, lorsque le robot peut prendre des raccourcis dans la cellule ou a le temps d'accélérer entre deux machines éloignées.[1]

Dans la section suivante, nous montrons que la Conjecture des 1-cycles est valide pour des cellules à deux machines quels que soient les temps de transfert (Section 8.1). Puis, nous étudions des cellules à trois machines et plus lorsque les temps de transfert vérifient l'inégalité triangulaire (Section 8.2). Dans la dernière section (Section 8.3), nous considérons des temps de transfert quelconques et analysons la complexité de la recherche du meilleur 1-cycle.

La matrice des temps de transfert est notée $D = (\delta_{i,j})_{0 \leq i,j \leq m+1}$. Ainsi, $\delta_{i,j}$ désigne le temps nécessaire au robot pour se déplacer, vide ou chargé, de la machine M_i à la machine M_j. On impose des durées de transfert nulles sur la diagonale ($\delta_{i,i} = 0$). Le robot n'effectue jamais certains trajets dans la cellule. Par exemple, le robot ne va en M_0 que pour charger une pièce puis aller en M_1. Ainsi, il n'effectue jamais le trajet de M_0 à M_i pour $i \geq 2$. Par conséquent, nous ne définissons pas systématiquement les valeurs $\delta_{0,i}$ pour $i \geq 2$. De même, les valeurs de $\delta_{i,n+1}$ pour $i \leq n-1$ n'ont pas besoin d'être définies.

[1] Les travaux présentés dans la Section 8.3 ont été faits en collaboration avec Wieslaw Kubiak, de la *Memorial University of Newfoundland* au Canada.

8.1 Cellule à deux machines

Nous considérons, dans cette section, des cellules à deux machines dont les temps de transfert inter-machines sont quelconques. Pour ce cas très simple, nous montrons que la généralisation des durées de transfert ne modifie pas la validité de la Conjecture des 1-cycles.

Théorème 17 *La Conjecture des 1-cycles est valide pour une cellule à deux machines quels que soient les temps de transfert inter-machines.*

Démonstration Lorsqu'il exécute l'activité A_1, le robot transporte une pièce de M_1 à M_2 et ces deux machines sont vides. Ainsi, à chaque exécution de A_1 l'état du système est le même. On peut donc appliquer le Théorème 3 (page 43) qui affirme que, dans ce cas, la durée d'un k-cycle est égale à la somme des durées des sous-cycles qui le composent. Ceci implique que les 1-cycles sont dominants. □

Le cas de deux machines étant résolu, nous considérons, dans la suite, des cellules de trois machines et plus.

8.2 Inégalité triangulaire

Nous considérons dans cette section des cellules robotisées pour lesquelles les temps de transfert ne sont pas nécessairement additifs mais vérifient l'inégalité triangulaire et sont symétriques, c'est-à-dire

$$\begin{aligned} &\delta_{i,j} \leq \delta_{i,k} + \delta_{k,j} \qquad \forall i,j,k \ ; \\ &\delta_{i,j} = \delta_{j,i}. \end{aligned}$$

Nous étudions d'abord la dominance des 1-cycles afin de déterminer s'il est possible de limiter la recherche du meilleur cycle aux 1-cycles. Puis, nous restreignons le problème aux 1-cycles et discutons la dominance des permutations pyramidales.

Hertz [43] a prouvé que la Conjecture des 1-cycles est fausse même si les durées de transfert vérifient l'inégalité triangulaire. Il propose l'exemple suivant pour lequel tous les 1-cycles sont dominés par un 2-cycle :

- le nombre de machines, $m = 3$;
- la matrice des temps de transfert,

$$D = \begin{pmatrix} 0 & 3 & 5 & 6 & 9 \\ 3 & 0 & 3 & 5 & 6 \\ 5 & 3 & 0 & 3 & 5 \\ 6 & 5 & 3 & 0 & 3 \\ 9 & 6 & 5 & 3 & 0 \end{pmatrix}$$

- les temps de chargement et de déchargement $\epsilon_{i+1}^l = \epsilon_i^u = 0$ pour $i = 0, 1, 2, 3$;
- les temps d'usinage $p_i = 11$ pour $i = 1, 2, 3$.

Les durées de transfert définies par la matrice D vérifient l'inégalité triangulaire et sont symétriques. Pour cette instance, les durées des 1-cycles sont

$$\begin{array}{ll} T(A_0A_1A_2A_3) = 54 \ ; & T(A_0A_2A_3A_1) = 37 \ ; \\ T(A_0A_1A_3A_2) = 37 \ ; & T(A_0A_3A_1A_2) = 40 \ ; \\ T(A_0A_2A_1A_3) = 32 \ ; & T(A_0A_3A_2A_1) = 32. \end{array}$$

Or, le 2-cycle $(A_0A_3A_1A_0A_2A_1A_3A_2)$ ne dure que 62 unités de temps et $62/2 = 31 < 32$. Il est donc strictement meilleur que tous les 1-cycles. La Conjecture des 1-cycles est donc fausse à partir de trois machines même si les temps de transfert vérifient l'inégalité triangulaire.

Hertz [43] a donc prouvé que les 1-cycles ne sont pas dominants. Toutefois, il peut être intéressant de déterminer si trouver le meilleur 1-cycle reste un problème facile pour lequel les permutations pyramidales sont dominantes. Pour deux machines, tous les 1-cycles sont des permutations pyramidales. Donc le problème de la dominance des permutations pyramidales ne se pose pas. Pour une cellule à trois machines, nous avons modifié l'instance précédente pour obtenir un contre-exemple à la dominance des permutations pyramidales :

- le nombre de machines, $m = 3$;
- la matrice des temps de transfert,

$$D = \begin{pmatrix} 0 & 3 & 5 & 6 & 8 \\ 3 & 0 & 3 & 5 & 6 \\ 5 & 3 & 0 & 3 & 5 \\ 6 & 5 & 3 & 0 & 3 \\ 8 & 6 & 5 & 3 & 0 \end{pmatrix}$$

- les temps de chargement et de déchargement, $\epsilon_{i+1}^l = \epsilon_i^u = 0$ pour $i = 0, 1, 2, 3$;
- les temps d'usinage $p_i = 11$ pour $i = 1, 2, 3$.

Les durées de transfert définies par la matrice D vérifient l'inégalité triangulaire et sont symétriques. Les temps de cycle des 1-cycles sont

$$\begin{array}{ll} T(A_0A_1A_2A_3) = 53 \ ; & T(A_0A_2A_3A_1) = 37 \ ; \\ T(A_0A_1A_3A_2) = 37 \ ; & T(A_0A_3A_1A_2) = 40 \ ; \\ T(A_0A_2A_1A_3) = 31 \ ; & T(A_0A_3A_2A_1) = 32. \end{array}$$

Ainsi, le meilleur 1-cycle est $(A_0A_2A_1A_3)$ et ce n'est pas une permutation pyramidale. La dominance des permutations pyramidales est donc liée à l'additivité des durées de transfert.

Pour les cellules robotisées à partir de trois machines, la dominance des permutations pyramidales est liée au fait que les durées de transfert sont additives (c'est-à-dire, vérifient l'égalité triangulaire). Cette propriété ne peut donc pas être étendue au cas où les durées de transfert ne vérifient que l'inégalité triangulaire.

8.3 Complexité

Dans cette section, nous étudions la complexité du problème du meilleur 1-cycle dans une cellule robotisée avec des durées de transfert généralisées. Nous n'imposons donc aucune restriction sur la matrice D. Nous transformons ce problème d'optimisation en un problème de décision que nous désignons comme le problème des Cellules Robotisées Généralisées ou CRG. Ce problème prend en entrée

- un nombre de machines m ;
- des temps de chargement et de déchargement $\epsilon^l_{i+1}, \epsilon^u_i$ pour $i = 0, 1 \dots m$;
- une matrice D des temps de transfert ;
- des temps d'usinage p_i pour $i = 1, 2 \dots m$;
- une borne C.

La question est : existe-t-il un 1-cycle de durée inférieure à C ?

Définissons l'hypothèse suivante :

(H1) Le robot n'attend jamais à une machine qui contient une pièce prête.

À notre connaissance, il existe deux preuves de NP-complétude dans la littérature des cellules robotisées pour le cas mono-produit : une de Lei et Wang [56] et une autre de Crama et van de Klundert [28]. La première viole l'hypothèse (H1) qui est habituellement imposée en pratique. La deuxième concerne le problème du HSP (*Hoist Scheduling Problem*) avec des durées de transfert additives. Ainsi, ces deux preuves ne sont pas satisfaisantes pour le cas de production de pièces mécaniques dans des cellules robotisées. Rappelons que Crama et van de Klundert ont prouvé que lorsque les durées de transfert sont additives, le problème CRG peut être résolu à l'optimalité en temps polynomial [27].

Théorème 18 *Le problème CRG est fortement NP-complet.*

Démonstration Dans [56], les auteurs ont prouvé que le problème CRG est dans NP. Pour montrer la NP-complétude forte de ce problème, nous utilisons le voyageur de commerce ou TSP (*Traveling Salesman Problem*).

Le TSP prend en entrée :

- un graphe orienté $G = (V, A)$ où $V = \{0, 1 \dots n\}$, $n > 1$, et $A \subseteq V \times V$;
- une fonction distance $d : A \rightarrow R_+ \cup \{0\}$ qui associe $d_{i,j}$ à l'arc $(i, j) \in A$;

- une borne $B > 0$.

La question est : existe-t-il un circuit hamiltonien dans G de longueur inférieure à B ?

Cette question peut être reformulée de la manière suivante : existe-t-il une permutation ρ des sommets $\{0, 1 \dots n\}$ telle que $(\rho(i), \rho(i+1)) \in A$ pour $i = 0, 1 \dots n-1$, et $(\rho(n), \rho(0)) \in A$ et

$$d_{\rho(n),\rho(0)} + \sum_{i=0}^{n-1} d_{\rho(i),\rho(i+1)} \leq B \ ?$$

Nous montrons dans un premier temps que le TSP reste fortement NP-complet même pour des graphes sans arc de la forme $(i, i+1)$ ni arc de la forme $(i-1, i+1)$.

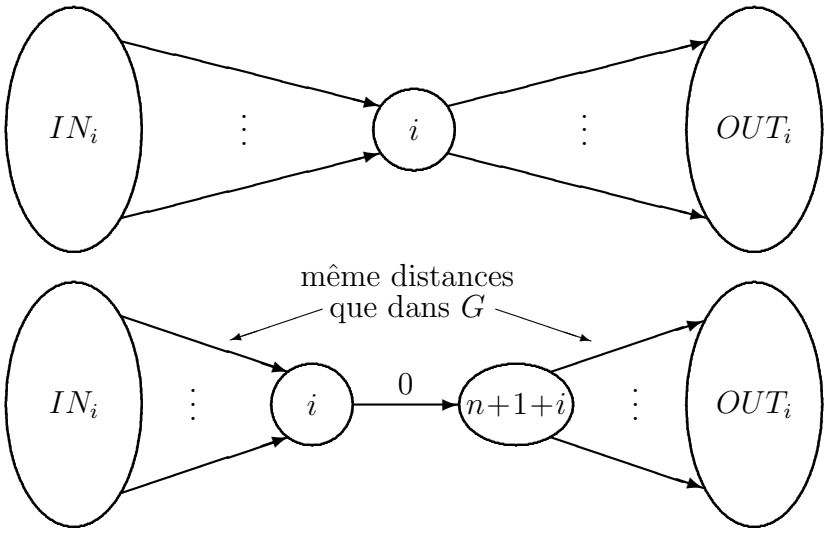

Figure 8.1: Transformation de G en G'

Considérons un graphe orienté G. Chaque sommet i de G est coupé en deux sommets i et $n+i+1$ comme indiqué sur la Figure 8.1. Les arcs entrants de i dans G' sont les arcs entrants de i dans G. Les arcs sortants de $n+i+1$ dans G' sont les arcs sortants de i dans G. La distance associée à l'arc $(i, n+1+i)$ est égale à 0.

Les sommets de G' sont numérotés de 0 à $2n+1$. Les arcs de G' sont de la forme (i, j) où $n+1 \leq i \leq 2n+1$ et $0 \leq j \leq n$ ou de la forme $(i, n+1+i)$ pour $0 \leq i \leq n$. Ainsi, G', ne contient aucun arc de la forme $(i, i+1)$ et aucun arc de la forme $(i-1, i+1)$. Il est clair qu'il existe un circuit hamiltonien dans G de longueur inférieure à B si et seulement si il existe une circuit hamiltonien dans G' de longueur inférieure à B.

Nous réduisons le TSP ainsi modifié au problème CRG. Étant donnés un graphe $G' = (\{0, 1 \dots n\}, A)$ sans arc $(i, i+1)$ ni arc $(i-1, i+1)$, des distances $d_{i,j}$ pour $(i, j) \in A$, et une borne B, nous définissons une instance du problème CRG de la manière suivante :

- le nombre de machines, $m = n$;
- les temps d'usinage, $p_i = B + 1$ pour $i = 1, 2 \dots n$;
- les temps de trajet, pour $i, j = 0, 1 \dots n$ et $i + 1 \neq j$

$$\delta_{i+1,j} = \begin{cases} d_{i,j} & \text{si } (i, j) \in A \ ; \\ B + 1 & \text{si } (i, j) \notin A \ ; \end{cases}$$

- les temps de trajet, $\delta_{0,1} = B + 1$ et $\delta_{m,m+1} = B + 1$;
- les temps de chargement et de déchargement $\epsilon^l_{i+1} = \epsilon^u_i = 0$.

La question est : existe-t-il un 1-cycle de longueur inférieure à $C = B+(n+1)(B+1)$?

Comme $(i-1, i+1) \notin A$, la durée de transfert $\delta_{i,i+1}$ est égale à $B + 1$. Donc, la durée d'une activité est aussi $B + 1$.

Nous montrons d'abord que si le TSP admet une réponse positive, alors il existe un 1-cycle dont la durée est inférieure à C. Considérons un circuit hamiltonien ρ de G' de longueur inférieure à B. Comme il n'existe pas d'arc $(i, i+1)$ dans G', ce circuit vérifie

$$\rho(i) + 1 \neq \rho(i + 1) \quad \text{pour } i = 0, 1 \dots n - 1.$$

Définissons la permutation des activités π comme suit :

$$\pi(i) = \rho(i) \text{ pour } i = 0, 1 \dots n.$$

Comme la permutation ρ, la permutation π vérifie $\pi(i + 1) \neq \pi(i) + 1$. Notons que la durée d'une activité est égale aux temps d'usinage. Or A_{i+1} ne suit pas immédiatement A_i dans π. Donc, le robot n'attend jamais à une machine. De plus, la durée de l'activité $A_{\pi(i)}$ est $B + 1$ et après l'exécution de $A_{\pi(i)}$, le robot doit se déplacer de $M_{\pi(i)+1}$ à $M_{\pi(i+1)}$. Ainsi, le temps de cycle $T(\pi)$ de π vérifie

$$T(\pi) = (n + 1)(B + 1) + \sum_{i=0}^{m-1} \delta_{\pi(i)+1,\pi(i+1)} + \delta_{\pi(m)+1,\pi(0)}.$$

Le premier terme correspond au temps nécessaire pour exécuter les $m+1$ activités et les termes suivants correspondent aux mouvements à vide du robot. Comme

$(\pi(i), \pi(i+1)) \in A$ et $(\pi(m), \pi(0)) \in A$, nous pouvons réécrire $T(\pi)$ comme suit

$$\begin{aligned} T(\pi) &= (n+1)(B+1) + \sum_{i=0}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(0)} \\ &= (n+1)(B+1) + \sum_{i=0}^{n-1} d_{\rho(i),\rho(i+1)} + d_{\rho(n),\rho(0)} \\ &\leq (n+1)(B+1) + B. \end{aligned}$$

Supposons maintenant que le problème CRG admette une réponse positive. Considérons une permutation des activités $(A_{\pi(0)} A_{\pi(1)} \dots A_{\pi(n)})$ dont le temps de cycle $T(\pi)$ vérifie

$$T(\pi) \leq (n+1)(B+1) + B \tag{8.1}$$

S'il existe i_0 tel que $(\pi(i_0), \pi(i_0+1)) \notin A$, avec $\pi(n+1) = \pi(0)$, alors la durée entre la fin de $A_{\pi(i_0)+1}$ et le début de $M_{\pi(i_0+1)}$ est égale à $B+1$ unités de temps. Comme le robot exécute chaque activité une fois, on obtient

$$T(\pi) \geq (n+1)(B+1) + B + 1$$

ce qui contredit l'inégalité (8.1). Donc, $(\pi(i), \pi(i+1)) \in A$, et on peut définir un circuit hamiltonien ρ_π dans G' comme suit : $\rho_\pi(i) = \pi(i)$ pour $i = 0, 1 \dots n$. Le graphe G' ne contient pas d'arc $(i, i+1)$, pour tout $0 \leq i < n$, ce qui implique que $\pi(i) + 1 \neq \pi(i+1)$. Par conséquent, le robot n'attend jamais aux machines et le temps de cycle $T(\pi)$ vérifie

$$\begin{aligned} T(\pi) &= (m+1)(B+1) + \sum_{i=0}^{m-1} \delta_{\pi(i)+1,\pi(i+1)} + \delta_{\pi(n)+1,\pi(0)} \\ &= (n+1)(B+1) + \sum_{i=0}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(0)}. \end{aligned}$$

Donc, ρ_π est un circuit hamiltonien qui vérifie

$$\sum_{i=0}^{n-1} d_{\rho_\pi(i),\rho_\pi(i+1)} + d_{\rho_\pi(n),\rho_\pi(0)} \leq B.$$

Ceci conclut la preuve. □

Le Théorème 18 reste valide si la matrice D des temps de transfert vérifie l'inégalité triangulaire. En effet, dans la preuve précédente, il suffit de rajouter une valeur M assez grande aux temps d'usinage, à tous les éléments de la matrice D sauf aux élément de la diagonale et de rajouter $2(m+1)M$ à la borne C.

8.4 Conclusion

Dans ce chapitre, nous avons démontré que, pour des temps de transfert inter-machines qui vérifient l'inégalité triangulaire, trouver le meilleur 1-cycle est un problème fortement NP-complet. Or, si les temps de transfert vérifient l'égalité triangulaire (c'est-à-dire, sont additives) alors ce problème est polynomial. Reste le cas intermédiaire où les durées de transfert vérifient l'inégalité triangulaire et sont symétriques mais ne sont pas additives. Dans ce cas, dès trois machines, la dominance des permutations pyramidales n'est plus vraie et la complexité de la recherche du meilleur 1-cycle est un problème ouvert.

Conclusion

Dans ce mémoire, nous avons analysé la production cyclique de pièces identiques dans des flow-shops munis d'un robot chargé du transfert des pièces entre les machines. Les deux thèmes principaux que nous avons abordés sont la validité de la Conjecture des 1-cycles et la complexité de la recherche du meilleur 1-cycle. Ces thèmes ont été analysés en fonction des durées de transfert dans les flow-shops robotisés, en ligne, sans espace de stockage et pour des extensions de ce problème.

Le Tableau 8.1 décrit la validité de la Conjecture des 1-cycles en fonction des durées de transfert. Soit $\delta_{i,j}$ la durée du transfert entre les machines M_i et M_j. Les durées de transfert sont quelconques lorsqu'aucune restriction sur $\delta_{i,j}$ n'est imposée (Chapitre 8). Si les durées de transfert vérifient l'inégalité triangulaire, alors on a :

$$\delta_{i,k} + \delta_{k,j} \geq \delta_{i,j} \quad \text{pour} \quad i < k < j.$$

Lorsque les durées de transfert sont additives (Chapitre 3), on a l'égalité triangulaire :

$$\delta_{i,k} + \delta_{k,j} = \delta_{i,j} \quad \text{pour} \quad i < k < j.$$

Dans le cas régulier (Chapitre 4), on a

$$\delta_{i,i+1} = \delta_{i+1,i} = \delta \quad \text{et} \quad \delta_{i,j} = |j - i|\delta.$$

Enfin, le cas régulier équilibré est un cas particulier du cas régulier : les temps d'usinage sont égaux sur toutes les machines.

Le Tableau 8.1 indique que la validité de la conjecture n'est pas connue dans le cas régulier équilibré pour des cellule à cinq machines et plus.

Nous avons également étudié la dominance des permutations pyramidales et la complexité de la recherche du meilleur 1-cycle en fonction des durées de transfert [Tableau 8.2].

Nous avons étudié des extensions de la cellule de base. Nous avons énoncé quelques remarques concernant les cellules circulaires, c'est-à-dire, lorsque l'entrée et la sortie de la cellule sont associées (Chapitre 5). Nous avons identifié les meilleurs 1-cycles pour des cellules à deux, trois et quatre machines. Nous savons que certains 1-cycles dominent d'autres 1-cycles, mais nous ne savons pas identifier l'ensemble

Tableau 8.1: Validité de la Conjecture des 1-cycles en fonction des durées de transfert

	$m = 2$	$m = 3$	$m = 4$		$m \geq 5$
			$k = 2$	$k \geq 3$	
transferts quelconques	vraie	fausse			
inégalité triangulaire	vraie	fausse [43]			
transferts additifs	vraie [66, 29]		fausse		
cas régulier	vraie			fausse	
cas régulier équilibré	vraie				?

Tableau 8.2: Recherche du meilleur 1-cycle en fonction des durées de transfert

	dominance des permutations pyramidales	complexité de la recherche du meilleur 1-cycle
transferts quelconques	non	NP-complet
inégalité triangulaire	non	?
transferts additifs	oui [27]	$O(m^3)$ [27]
cas régulier	oui	$O(m^3)$
cas régulier équilibré	oui	constant

des 1-cycles dominants pour $m \geq 5$. La validité de la conjecture pour les cellules circulaires est également un problème ouvert.

Pour le HSP, nous avons démontré que les permutations pyramidales ne dominent pas les 1-cycles et que la Conjecture des 1-cycles est fausse pour des cellules de plus de trois machines (Chapitre 6).

Nous avons prouvé que, lorsqu'on ajoute un espace de stockage unitaire derrière chaque machine, la Conjecture des 1-cycles est vraie et nous avons analysé le gain entraîné par l'ajout d'espaces de stockage. Nous avons identifié des types de problèmes proches pour lesquels les méthodes de résolution pour les cellules robotisées avec zones de stockage s'appliquent également (Chapitre 7).

Pour le problème de base, d'autres questions ouvertes comme l'ergodicité des durées d'exécution des cycles ou la stabilité des permutations pyramidales ont été soulevées dans ce mémoire. Nous avons également proposé une conjecture faible des 1-cycles qui permet d'identifier le meilleur cycle de production pour des temps d'usinage relativement petits.

Le problème essentiel qu'il reste à résoudre est de trouver le meilleur cycle de production. Ce travail peut également être un point de départ pour des recherches sur les cellules robotisées où le nombre de types de pièces à produire est considéré comme une donnée.

Bibliography

[1] A. Agnetis. Scheduling no-wait robotic cells with two and three machines. Technical Report 21-97, Università degli studi di Roma, La Sapienza, 1997.

[2] A. Agnetis and D. Pacciarelli. Part sequencing in three machine no-wait robotic cells. Technical Report 10-98, Università degli studi di Roma, La Sapienza, 1998.

[3] A. Agnetis, D. Pacciarelli, and F. Rossi. Batch scheduling in a two-machine cell with swapping devices. Technical Report 28-94, Università degli studi di Roma, La Sapienza, 1994.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows, theory, algorithms, and applications*. Prentice Hall, New Jersey, 1993.

[5] C. R. Asfahl. *Robots and manufacturing automation*. John Wiley & Sons, New York, NY, 1985.

[6] J. Błażewicz, R. E. Burkard, G. Finke, and G. J. Woeginger. Vehicle scheduling in two-cycle flexible manufacturing systems. *Mathematical and Computer Modelling*, 20(2):19–31, 1994.

[7] J. Błażewicz, W. Cellary, R. Slowinski, and J. Węglarz. *Scheduling under resource constraints: deterministic models*. Baltzer, Basel, 1987.

[8] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Węglarz. *Scheduling computer and manufacturing processes*. Springer, Berlin, Heidelberg, New York, 1996.

[9] J. Błażewicz and G. Finke. Scheduling with resource management in manufacturing systems. *European Journal of Operational Research*, 76:1–14, 1994.

[10] J. Błażewicz, G. Finke, M.-L. Espinouse, and G. Pawlak. Scheduling vehicles in a cyclic flexible flowshop. *European Journal of Automation / Journal Européen des Systèmes Automatisés - APII-JESA*, 32(4):441–451, 1998.

[11] C. Bloch. *Contribution à l'ordonnancement dynamique de lignes de traitement de surface*. Thèse de doctorat, Université de Franche-Comté, U.F.R. Sciences et Techniques, Besançon, 1999.

[12] C. Bloch and M.-A. Manier. Notation et typologie du problème de l'ordonnancement d'une ligne de traitement de surface (Hoist Scheduling Problem). In *Deuxième congrès de la société Française de Recherche Opérationnelle et Aide à la Décision, ROADEF 99*, Autrans, France, 1999.

[13] N. Brauner and G. Finke. Cyclic scheduling in a robotic flowshop. In *Proceedings International Conference on Industrial Engineering and Production Management (IEPM'97)*, pages 439–449, Lyon, France, 1997.

[14] N. Brauner and G. Finke. On cycles and permutations in robotic cells. Internal Note, 1998.

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

[16] N. Brauner and G. Finke. On cycles and permutations in robotic cells. *Mathematical and Computer Modelling*, 1999. Submitted.

[17] N. Brauner and G. Finke. Optimal moves of the material handling system in a robotic flow-shop. In *Proceedings International Conference on Industrial Engineering and Production Management (IEPM'99)*, volume 1, pages 409–417, Glasgow, 1999.

[18] N. Brauner, G. Finke, and C. Gueguen. Atelier en ligne robotisé avec zones de stockage. *European Journal of Automation / Journal Européen des Systèmes Automatisés - APII-JESA*, 32(7-8):875–891, 1998.

[19] P. Brucker and S. Knust. `http://www.mathematik.uni-osnabrueck.de/research/OR/class/`.

[20] H. Chen, C. Chu, and J.-M. Proth. Cyclic hoist scheduling based on graph theory. In *IEEE Symposium on emerging technologies and factory automation*, Paris, October 1995.

[21] H. Chen, C. Chu, and J.-M. Proth. Sequencing of parts in robotic cells. *International Journal of Flexible Manufacturing Systems*, 9(1):81–103, 1997.

[22] P. Chrétienne, E. G. Coffman, J. K. Lenstra, and Z. Liu. *Scheduling theory and its applications*. John Wiley and Sons, Chichester, New York, Brisbane, Toronto, Singapore, 1997.

[23] C. Chu, J.-M. Proth, and S. Sethi. Heuristic procedures for minimizing makespan and the number of required pallets. *European Journal of Operational Research*, 86:491–502, 1995.

[24] Y. Crama. Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research*, 99(1):136–153, 1997.

[25] Y. Crama, V. Kats, J. van de Klundert, and E. Levner. Cyclic scheduling in robotic flowshops. *Annals of Operations Research: Mathematics of Industrial Systems*, in press, 1999.

[26] Y. Crama and J. van de Klundert. Cyclic scheduling in 3-machine robotic flow shops. Research Memorandum RM/97/018, Maastricht University, Maastricht, 1997.

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

[28] Y. Crama and J. van de Klundert. Robotic flowshop scheduling is strongly NP-complete. In *Ten Years LNMB, W. K. Klein Haneveld, O.J. Vrieze and L.C.M. Kallenberg (eds.), CWI Tract 122*, pages 277–286, Amsterdam, 1997.

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

[30] M.-L. Espinouse. *Flow-shop et extensions : chevauchement des tâches, indisponibilité des machines et système de transport.* Thèse de doctorat, Université Joseph Fourier, Grenoble, 1998.

[31] G. Finke, C. Gueguen, and N. Brauner. Robotic cells with buffer space. In *Proceedings ECCO IX, Conference of the European Chapter on Combinatorial Optimization*, 9 pages unnumbered, Dublin, Ireland, 1996.

[32] T. Ganesharajah, N. G. Hall, and C. Sriskandarajah. Design and operational issues in AGV-served manufacturing systems. Working Paper 95-13, University of Toronto, 1995.

[33] M. R. Garey and D. S. Johnson. *Computers and intractability, A guide to the theory of NP-completeness.* W. H. Freeman and Company, New York, 1979.

[34] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.

[35] P. C. Gilmore and R. E. Gomory. Sequencing a one state-variable machine: a solvable case of the traveling salesman problem. *Operations Research*, 12:655–679, 1964.

[36] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:169–231, 1979.

[37] O. Grunder, P. Baptiste, and D. Chappe. 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):2189–2211, 1997.

[38] N. G. Hall, H. Kamoun, and C. Sriskandarajah. Scheduling in robotic cells : Two machine cells and identical parts. Working Paper 93-06, University of Toronto, 1993.

[39] N. G. Hall, H. Kamoun, and C. Sriskandarajah. Scheduling in robotic cells: Large cells. Working Paper 93-07, University of Toronto, 1993.

[40] N. G. Hall, H. Kamoun, and C. Sriskandarajah. Scheduling in robotic cells: Classification, two and three machine cells. *Operations Research*, 45(3):421–439, 1997.

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

[42] C. Hanen and A. Munier. Ordonnancement cyclique d'un robot sur une ligne de galvanoplastie : modèles et algorithmes. Rapport de recherche, LITP/IBP 93.30, 1993.

[43] A. Hertz. Private communication, 1999.

[44] A. Hertz, Y. Mottet, and Y. Rochat. On a scheduling problem in a robotized analytical system. *Discrete Applied Mathematics*, 65(1-3):285–318, 1996.

[45] J. Hurink and S. Knust. Flow-shop problems with transportation times and a single robot. Working Paper 201, University of Osnabrück, 1998.

[46] I. Ioachim and F. Soumis. Schedule efficiency in a robotic production cell. *International Journal of Flexible Manufacturing Systems*, 7:5–26, 1995.

[47] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1:61–68, 1954.

[48] H. Kamoun, N. G. Hall, and C. Sriskandarajah. Scheduling in robotic cells: heuristics and cell design. Working Paper 93-08, University of Toronto, 1994.

[49] R. M. Karp. A characterization of the minimum cycle mean in a diagraph. *Discrete mathematics*, 23:309–311, 1978.

[50] R. E. King, T. J. Hodgson, and F. W. Chafee. Robot task scheduling in a flexible manufacturing cell. *IIE Transactions*, 25(2):80–87, 1993.

[51] H. Kise. On an automated two-machine flowshop scheduling problem with infinite buffer. *Journal of the Operations Research Society of Japan*, 34(3):354–361, 1991.

[52] H. Kise, H. Matsuo, and R. S. Sullivan. Optimal cyclic scheduling for automated two-machine flowshops with no buffer. In *Symposium on Flexible Automation*, Kyoto, Japan, 1990.

[53] H. Kise, T. Shioyama, and T. Ibaraki. Automated two-machine flowshop scheduling: a solvable case. *IIE Transactions*, 23(1):10–16, 1991.

[54] K. Kogan and E. Levner. A polynomial algorithm for scheduling small-scale manufacturing cells served by multiple robots. *Computers and Operations Research*, 25(1):53–62, 1998.

[55] L. Lei and T. J. Wang. On the optimal cyclic schedules of single hoist electroplating processes. Working Paper 89-06, Graduate School of Management, Rutgers University, 1989.

[56] L. Lei and T. J. Wang. A proof: the cyclic hoist-scheduling problem is NP-complete. Working Paper #89-0016, Graduate School of Management, Rutgers University, 1989.

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

[58] E. Levner, L. Meyzin, and A. Ptuskin. Fuzzy periodic scheduling of a production line with a robot. Working Paper, Center for Technology and Education, Holon, 199.

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

[60] S. T. McCormick and U. S. Rao. Some complexity results in cyclic scheduling. *Mathematical and Computer Modelling*, 20(2):107–122, 1994.

[61] W. C. Ng. Determining the optimal number of duplicated process tanks in a single-hoist circuit board production line. *Computers and Industrial Engineering*, 28(4):681–688, 1995.

[62] W. C. Ng. A branch and bound algorithm for hoist scheduling of a circuit board production line. *International Journal of Flexible Manufacturing Systems*, 8:45–65, 1996.

[63] C. H. Papadimitriou and P. C. Kanellakis. Flowshop scheduling with limited temporary storage. *Journal of the Association for Computing Machinery*, 27(3):533–549, 1980.

[64] L. W. Phillips and P. S. Unger. Mathematical programming solution of a hoist scheduling program. *AIIE Transactions*, 8(2):219–225, 1976.

[65] M. Pinedo. *Scheduling, theory, algorithms and systems.* Prentice Hall, New Jersey, 1995.

[66] S. P. Sethi, C. Sriskandarajah, G. Sorger, J. Błażewicz, and W. Kubiak. Sequencing of parts and robot moves in a robotic cell. *International Journal of Flexible Manufacturing Systems*, 4:331–358, 1992.

[67] G. W. Shapiro and H. L. W. Nuttle. Hoist scheduling for a PCB electroplating facility. *IIE Transactions*, 20(2):157–167, 1988.

[68] W. Song, R. L. Storch, and Z. B. Zabinsky. An example for scheduling a chemical processing tank line. In *IEEE Symposium on Emerging Technologies and Factory Automation*, pages 475–482, Paris, 1995.

[69] W. Song, Z. B. Zabinsky, and R. L. Storch. An algorithm for scheduling a chemical processing tank line. *Production Planning and Control*, 4(4):323–332, 1993.

[70] V. A. Strusevich and C. M. Zwaneveld. On non-permutation solutions to some two machine flow shop scheduling problems. *Zeitschrift fur Operations Research*, 39(3):305–319, 1994.

[71] Q. Su and F. F. Chen. Optimally sequencing of double-gripper gantry robot moves in tightly-coupled serial production systems. Technical Report, Florida International University, 1995.

[72] V. S. Tanaev, Y. N. Sotskov, and V. A. Strusevich. *Scheduling theory. Multi-stage systems*. Kluwer Academic Publishers, Dordrecht, Boston, London, 1994.

[73] V. S. Tanaev, Y. N. Sotskov, and V. A. Strusevich. *Scheduling theory. Single-stage systems*. Kluwer Academic Publishers, Dordrecht, Boston, London, 1994.

[74] J. van de Klundert. *Scheduling problems in automated manufacturing*. PhD thesis, Faculty of Economics and Business Administration, University of Limburg, Maastricht, The Netherlands, 1996.

[75] A. Vignier, J.-C. Billaut, and C. Proust. Les problèmes d'ordonnancement de type flow-shop hybride : état de l'art. *RAIRO-RO*, 33(2):117–183, 1999.

Appendix A

Conjecture des 1-cycles pour m = 4 et k = 2

Dans cette annexe, nous considérons les cellules régulières à quatre machines.

Théorème 19 *Dans une cellule régulière à quatre machines, les 1-cycles dominent les 2-cycles.*

Démonstration Pour prouver ce théorème, nous utilisons l'approche algébrique et plus particulièrement la Proposition 9 (page 68). Comme cette preuve est longue, nous la décomposons en cinq étapes qui couvrent tous les cas possibles. Nous montrons que

- **Étape 0** certains cas triviaux pour lesquels (C2) est vraie peuvent être éliminés ;
- **Étape 1** si, pour un k-cycle C_k, $u_4(C_k) = k$ ou $u_1(C_k) = k$ alors la condition (C1) est vraie (dans les étapes suivantes, nous considérons donc que $u_1(C_k) < k$ et $u_4(C_k) < k$) ;
- **Étape 2** si le k-cycle C_k vérifie $m_2(C_k) = 2k$, alors (C4) est vraie ;
- **Étape 3** si le 2-cycle C_2 vérifie $m_2(C_2) = 6$, alors (C1) ou (C2) ou (C4) est vraie ;
- **Étape 4** si le 2-cycle C_2 vérifie $m_2(C_2) \geq 8$, alors (C2) ou (C4) est vraie.

Les temps de cycle des permutations pyramidales sont donnés dans le Tableau A.1 Le temps de chargement/déchargement vérifie $T_L(\pi_j) = 10\epsilon$ pour toute permutation pyramidale $\pi_j (j = 0, 1 \ldots 7)$. Pour le k-cycle C_k, le système $S_4(C_k)$ est donné

Tableau A.1: Temps de cycle des permutations pyramidales pour $m = 4$

j	π_j	$T_T(\pi_j)$	$T_W(\pi_j)$
0	$(A_0A_1A_2A_3A_4)$	10δ	$p_1 + p_2 + p_3 + p_4$
1	$(A_0A_1A_2A_4A_3)$	12δ	$p_1 + p_2 +$ $\max(0, p_3 - 4\delta - 2\epsilon, p_4 - 8\delta - 6\epsilon - p_1 - p_2)$
2	$(A_0A_1A_3A_4A_2)$	12δ	$p_1 + p_4 +$ $\max(0, p_2 - 6\delta - 4\epsilon - p_4, p_3 - 6\delta - 4\epsilon - p_1)$
3	$(A_0A_1A_4A_3A_2)$	14δ	$p_1 + \max(0, p_2 - 8\delta - 4\epsilon,$ $p_3 - 10\delta - 6\epsilon - p_1, p_4 - 10\delta - 6\epsilon - p_1)$
4	$(A_0A_2A_3A_4A_1)$	12δ	$p_3 + p_4 +$ $\max(0, p_1 - 8\delta - 6\epsilon - p_3 - p_4, p_2 - 4\delta - 2\epsilon)$
5	$(A_0A_2A_4A_3A_1)$	14δ	$\max(0, p_1 - 10\delta - 6\epsilon, p_2 - 4\delta - 2\epsilon,$ $p_3 - 4\delta - 2\epsilon, p_4 - 10\delta - 6\epsilon, p_2 + p_3 - 8\delta - 4\epsilon)$
6	$(A_0A_3A_4A_2A_1)$	14δ	$p_4 + \max(0, p_1 - 10\delta - 6\epsilon - p_4,$ $p_2 - 10\delta - 6\epsilon - p_4, p_3 - 8\delta - 4\epsilon)$
7	$(A_0A_4A_3A_2A_1)$	16δ	$\max_i(0, p_i - 12\delta - 6\epsilon)$

par

$$S_4(C_k) = \begin{cases} x_0 + x_1 + x_2 + x_3 = u_1(C_k) \\ x_0 + x_2 + x_4 + x_6 = u_4(C_k) \\ x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = k \\ 2x_0 + 2x_1 + 4x_2 + 4x_3 + 2x_4 + 2x_5 + 4x_6 + 4x_7 = min(m_2(C_k), 4k) \\ x_i \text{ entier} \quad \text{pour } i = 0, 1 \dots 7 \end{cases}$$

Étape 0 Supposons que l'une des conditions suivantes soit vérifiée :

$$\begin{aligned} T_W(\pi_1) &= p_4 - 8\delta - 6\epsilon \ ; \\ T_W(\pi_3) &= \max(p_3 - 10\delta - 6\epsilon, p_4 - 10\delta - 6\epsilon) \ ; \\ T_W(\pi_4) &= p_1 - 8\delta - 6\epsilon \ ; \\ T_W(\pi_5) &= \max(p_1 - 10\delta - 6\epsilon, p_4 - 10\delta - 6\epsilon) \ ; \\ T_W(\pi_6) &= \max(p_1 - 10\delta - 6\epsilon, p_2 - 10\delta - 6\epsilon) \ ; \\ T_W(\pi_7) &= \max_i(p_i - 12\delta - 6\epsilon). \end{aligned}$$

Dans ce cas, la condition (C2) est vraie. Supposons par exemple que $T_W(\pi_4) = p_1 - 8\delta - 6\epsilon$, alors on a $T(\pi_4) = 4\delta + 4\epsilon + p_1$.

Nous réduisons donc notre étude aux autres cas. C'est-à-dire que nous considérons que toutes les conditions suivantes sont vérifiées :

$$T_W(\pi_0) = p_1 + p_2 + p_3 + p_4 \ ;$$

$$
\begin{aligned}
T_W(\pi_1) &= w_3 + p_1 + p_2 \ ; \\
T_W(\pi_2) &= max(0, p_2 - 6\delta - 4\epsilon - p_4, p_3 - 6\delta - 4\epsilon - p_1) + p_1 + p_4 \ ; \\
T_W(\pi_3) &= max(0, p_2 - 8\delta - 4\epsilon) + p_1 \ ; \\
T_W(\pi_4) &= w_2 + p_3 + p_4 \ ; \\
T_W(\pi_5) &= w_2 + w_3 \ ; \\
T_W(\pi_6) &= max(0, p_3 - 8\delta - 4\epsilon) + p_4 \ ; \\
T_W(\pi_7) &= 0 \ ;
\end{aligned}
$$

où $w_2 = max(0, p_2 - 4\delta - 2\epsilon)$ et $w_3 = max(0, p_3 - 4\delta - 2\epsilon)$.

Étape 1 *Nous montrons que dans une cellule robotisée à quatre machines si, pour un k-cycle C_k, $u_4(C_k) = k$ ou $u_1(C_k) = k$, alors, pour toute instance I, la condition (C1) est vraie.*

Soit C_k, un k-cycle tel que $u_4(C_k) = k$. L'idée de la preuve est de se ramener à une cellule à trois machines en remplaçant, dans C_k, chaque motif A_3A_4 par l'activité A_3.

Pour une instance donnée, définissons la cellule à trois machines S' suivante : le temps de trajet inter-machines est δ ; le temps d'usinage sur la machine M_i est $p_i (i = 1, 2, 3)$; le temps de chargement de la machine M_4 est $p_4 + 2\epsilon + 2\delta$ et les autres temps de chargement ou de déchargement sont ϵ.

Soit C'_k le cycle C_k réduit à un système à trois machines en enlevant l'activité A_4. Il est aisé de vérifier que le cycle C'_k ainsi généré est bien un k-cycle. Le temps de cycle de C'_k dans S' est égal au temps de cycle de C_k. Or, on sait que C'_k est dominé par une permutation pyramidale π' (Théorème 5, page 58). Dans π', remplaçons l'activité A_3 par le motif A_3A_4 pour obtenir la permutation pyramidale π. Le temps de cycle de π' dans S' est égal au temps de cycle de π. Donc la permutation pyramidale π domine C_k ce qui implique que la condition (C1) est vraie.

La preuve est identique si $u_1(C_k) = k$.

Dans la suite, nous considérons que $u_1(C_k) < k$ et $u_4(C_k) < k$.

Étape 2 *Nous montrons que, dans une cellule à quatre machines, si le k-cycle C_k vérifie $m_2(C_k) = 2k$ alors (C4) est vraie.*

Dans cette preuve, u_i désigne $u_i(C_k)$. Nous prouvons que le vecteur x suivant est une W-couverture pyramidale de C_k (Condition (C4)) :

$$
\begin{aligned}
x_0 &= \min(u_1, u_4); \\
x_1 &= u_1 - \min(u_1, u_4); \\
x_4 &= u_4 - \min(u_1, u_4); \\
x_5 &= k - u_1 - u_4 + \min(u_1, u_4);
\end{aligned}
$$

$$x_2 = x_3 = x_6 = x_7 = 0.$$

Rappelons que $|S|$ représente le nombre d'occurrences de la séquence d'activités S dans C_k. Comme $m_2(C_k) = 2k$, pendant l'exécution de C_k, le robot fait exactement k fois le trajet de M_2 à M_3 et exactement k fois le trajet de M_3 à M_2. L'activité A_2 apparaît k fois dans C_k et pendant l'exécution de cette activité, le robot va de M_2 à M_3. Donc, le robot ne se déplace entre M_2 et M_3 que pendant l'exécution de A_2. Par conséquent, l'activité A_4 est toujours précédée soit par A_3 soit par A_2 ce qui implique que

$$|A_2A_4| + |A_3A_4| = k \tag{A.1}$$

Mais la Proposition 3 (page 32) indique que

$$|A_2A_3| + |A_2A_4A_3| \geq k$$

qui devient une égalité car A_2 est répété exactement k fois dans C_k. Par conséquent, A_2 est suivie soit de A_3 soit de A_4 ce qui implique que

$$|A_2A_3| + |A_2A_4| = k. \tag{A.2}$$

Les équations (A.1) et (A.2) entraînent que

$$|A_2A_3| = |A_3A_4|$$

et, par conséquent, $u_3 = u_4$ et $|A_2A_4A_3| = k - u_4$.

De la même manière, on peut prouver que $|A_1A_0A_2| = k - u_1$ et $u_1 = u_2$. De plus, les motifs $A_1A_0A_2$ et $A_2A_4A_3$ entraînent des temps d'attente de w_2 et w_3 respectivement. On a donc

$$\begin{aligned}
\sum_j x_j T_W(\pi_j) &= x_0(p_1 + p_2 + p_3 + p_4) + x_1(p_1 + p_2 + w_3) + \\
&\quad x_4(p_3 + p_4 + w_2) + x_5(w_2 + w_3) \\
&= u_1p_1 + u_1p_2 + u_4p_3 + u_4p_4 + (k - u_4)w_3 + (k - u_1)w_2 \\
&= u_1p_1 + u_2p_2 + u_3p_3 + u_4p_4 + |A_2A_4A_3|w_3 + |A_1A_0A_2|w_2 \\
&\leq T_W(C_k).
\end{aligned}$$

Le vecteur x est donc une W-couverture pyramidale de C_k.

Dans la suite de cette annexe, u_i désigne $u_i(C_2)$ et $|S|$ est le nombre d'occurrences de la séquence d'activités S dans C_2.

Étape 3 *Dans une cellule à quatre machines, soit C_2 un 2-cycle. Si $m_2(C_2) = 6$, alors (C1) ou (C2) ou (C4) est vraie.*

Considérons le vecteur $x = (x_i)(i = 0, 1 \ldots \mu)$, solution de $S_4(C_2)$, défini comme suit

$$x_0 = \min_i(u_i)$$

$$\begin{aligned}
x_1 &= \min(u_1, u_2) - \min_i(u_i) \\
x_2 &= 1 \text{ si } u_1 = 1 \text{ et } u_4 = 1 \text{ et } u_2 = 0 \text{ et } u_3 = 0 \\
&= 0 \text{ sinon} \\
x_3 &= u_1 - \min(u_1, u_2) - x_2 \\
x_4 &= \min(u_3, u_4) - \min_i(u_i) \\
x_5 &= 1 - \min(u_1, u_2) - \min(u_3, u_4) + \min_i(u_i) \\
x_6 &= u_4 - \min(u_3, u_4) - x_2 \\
x_7 &= 1 - u_1 + \min(u_1, u_2) - u_4 + \min(u_3, u_4) + x_2
\end{aligned}$$

Nous démontrons que (C1) est vraie ou que x est une W-couverture pyramidale de C_2. D'après la Proposition 1 (page 31) et la Proposition 2 (page 31) et comme $u_1 \leq 1$ et $u_4 \leq 1$ et $m_2(C_2) = 6$, le temps de cycle de C_2 vérifie

$$T(C_2) \geq 26\delta + u_1p_1 + u_2p_2 + u_3p_3 + u_4p_4 + |A_1A_0A_2|w_2 + |A_2A_4A_3|w_3.$$

La Proposition 3 indique que $u_2 + |A_1A_0A_2| \geq 1$ et $u_3 + |A_2A_4A_3| \geq 1$. Par conséquent, comme $p_2 \geq w_2$ et $p_3 \geq w_3$, l'équation précédente devient

$$T(C_2) \geq 26\delta + u_1p_1 + u_4p_4 + w_2 + w_3.$$

Si $p_2 \geq 8\delta + 4\epsilon$ et $u_1 = 1$, alors $T(C_2) \geq 22\delta + p_1 + p_2$ et $T(\pi_3) = 12\delta + 2p_1 + 2p_2$. Dans ce cas, si $p_1 + p_2 \geq 10\delta$ alors $22\delta + p_1 + p_2 \geq 32\delta = 2T(\pi_7)$. Sinon, $T(C_k) \geq 22\delta + p_1 + p_2 \geq 12\delta + 2p_1 + 2p_2$. Donc, C_2 est dominé par π_3 ou par π_7 et (C1) est vraie. On peut donc considérer que $u_1 \max(0, p_2 - 8\delta - 4\epsilon) = 0$. Or $x_3 \leq u_1$ implique que $x_3T_W(\pi_3) \leq x_3p_1$. De la même manière, on peut prouver que (C1) est vraie ou $x_2T_W(\pi_2) = x_2(p_1 + p_4)$ et $x_6T_W(\pi_6) = x_6p_4$. Par conséquent, on a

$$\begin{aligned}
\sum_{j=0}^{7} x_jT_W(\pi_j) &\leq x_0(p_1 + p_2 + p_3 + p_4) + x_1(p_1 + p_2 + w_3) + x_2(p_1 + p_4) + \\
&\quad x_3p_1 + x_4(p_3 + p_4 + w_2) + x_5(w_2 + w_3) + x_6p_4 \\
&\leq p_1(x_0 + x_1 + x_2 + x_3) + p_2(x_0 + x_1) + \\
&\quad p_3(x_0 + x_4) + p_4(x_0 + x_2 + x_4 + x_6) + \\
&\quad w_2(x_4 + x_5) + w_3(x_1 + x_5) \\
&\leq u_1p_1 + \min(u_1, u_2)p_2 + \min(u_3, u_4)p_3 + u_4(p_4) + \\
&\quad w_2(1 - \min(u_1, u_2)) + w_3(1 - \min(u_3, u_4)) \\
&\leq \sum u_ip_i + \\
&\quad (\min(u_1, u_2) - u_2)p_2 + (1 - \min(u_1, u_2))w_2 + \\
&\quad (\min(u_3, u_4) - u_3)p_3 + (1 - \min(u_3, u_4))w_3
\end{aligned}$$

Mais $p_2 \geq w_2$ et $\min(u_1, u_2) \leq u_2$ et $p_3 \geq w_3$ et $\min(u_3, u_4) \leq u_3$ impliquent que

$$\sum_{j=0}^{7} x_j T_W(\pi_j) \leq \sum u_i p_i + (1 - u_2)w_2 + (1 - u_3)w_3.$$

La Proposition 3 indique que $|A_1A_0A_2| \geq 1 - u_2$ et $|A_2A_4A_3| \geq 1 - u_3$. L'équation précédente devient donc

$$\begin{aligned} \sum_{j=0}^{7} x_j T_W(\pi_j) &\leq \sum u_i p_i + |A_1A_0A_2|w_2 + |A_2A_4A_3|w_3 \\ &\leq T_W(C_2) \end{aligned}$$

ce qui implique que (C4) est vraie.

Étape 4 *Dans une cellule à quatre machines, soit C_2 un 2-cycle. Si $m_2(C_2) \geq 8$ alors (C2) ou (C4) est vraie.*

Considérons le vecteur $x = (x_i)(i = 0, 1 \ldots \mu)$, solution de $S_m(C_2)$, défini comme suit :

$$x_0 = x_1 = x_4 = x_5 = 0$$

$$\begin{aligned} x_2 &= 1 \text{ si } u_1 = 1 \text{ et} u_4 = 1 \text{ et } [(\text{si } A_2A_0A_1A_3 \text{ et non } A_0A_1A_3A_0A_2) \text{ ou} \\ &\quad (\text{si } A_1A_3A_4A_2 \text{ et non } A_2A_4A_1A_3) \text{ ou } (\text{si } A_1A_3A_2 \text{ et } A_2A_1A_3)] \\ &= 0 \text{ sinon} \\ x_3 &= u_1 - x_2 \\ x_6 &= u_4 - x_2 \\ x_7 &= 2 - u_1 - u_4 + x_2 \end{aligned}$$

Nous prouvons que la condition (C2) est vraie ou que x est une W-couverture pyramidale de C_2. Nous considérons les cas suivants :
Cas 1 $u_1 = 0$ et $u_4 = 0$;
Cas 2 $u_1 = 1$ et $u_4 = 0$; (symétrique à $u_1 = 0$ et $u_4 = 1$)
Cas 3 $u_1 = 1$ et $u_4 = 1$.

Cas 1 $u_1 = 0$ et $u_4 = 0$.
Dans ce cas, $x = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 2)$ et $T_W(\pi_7) = 0$. Comme $T_W(C_2) \geq 0$, x est une W-couverture pyramidale et (C4) est vraie.

De manière à simplifier la preuve, nous posons $\epsilon = 0$.

Cas 2 $u_1 = 1$ et $u_4 = 0$.
Dans ce cas $x = (00010001)$. Nous prouvons que x est une W-couverture pyramidale de C_2. Le cycle C_2 peut être écrit sous la forme : $(A_0A_1 \ldots A_2 \ldots A_1yA_2)$ où

y représente une séquence d'activités. La séquence y peut contenir les activités A_3 et A_4. En effet, y ne peut pas contenir A_0 (entre deux occurrences de A_1, il y a exactement une occurrence de A_0) et y peut contenir au plus une fois A_3 (entre deux occurrences de A_2, il y a exactement une occurrence de A_3). De plus, $u_4 = 0$ implique que y contient au plus une fois l'activité A_4. Par conséquent seuls les cas suivants peuvent se produire :

$$\begin{array}{lll} y = \emptyset & \Rightarrow & T_W(C_2) \geq p_1 + p_2 \\ y = A_3 & \Rightarrow & T_W(C_2) \geq p_1 + max(0, p_2 - 4\delta) \\ y = A_4 & \Rightarrow & T_W(C_2) \geq p_1 + max(0, p_2 - 6\delta) \\ y = A_4A_3 & \Rightarrow & T_W(C_2) \geq p_1 + max(0, p_2 - 8\delta) \end{array}$$

Dans tous les cas, $T_W(C_2) \geq p_1 + max(0, p_2 - 8\delta)$. Par conséquent $T_W(\pi_3) + T_W(\pi_7) \leq T_W(C_2)$ ce qui implique que (C4) est vraie.

Cas 3 $u_1 = 1$ et $u_4 = 1$
Dans ce cas $x = (00100001)$ ou $x = (00010010)$. Il suffit de prouver que $T_W(\pi_3) + T_W(\pi_6) \leq T_W(C_2)$ ou $T_W(\pi_2) + T_W(\pi_7) \leq T_W(C_2)$.

Entre toutes les occurrences de A_1 et l'occurrence suivante de A_2, on peut avoir au plus les activités A_0, A_3, A_3, A_4 et A_4. Par conséquent, il existe un motif entre A_1 et A_2 avec moins de deux activités. Notons y ce motif qui peut être de la forme :

$$\begin{array}{lll} \emptyset & \Rightarrow & W_2 = p_2 \\ A_0 & \Rightarrow & W_2 = max(0, p_2 - 4\delta) \\ A_3 & \Rightarrow & W_2 = max(0, p_2 - 4\delta - W_3') \\ A_4 & \Rightarrow & W_2 = max(0, p_2 - 6\delta) \\ A_0A_3 & \Rightarrow & W_2 = max(0, p_2 - 8\delta - W_3') \\ A_0A_4 & \Rightarrow & \text{l'autre motif entre } A_1 \text{ et } A_2 \text{ est nécessairement } A_3A_4 \\ A_3A_0 & \Rightarrow & W_2 = max(0, p_2 - 8\delta - W_3') \\ A_3A_4 & \Rightarrow & W_2 = max(0, p_2 - 6\delta - p_4 - W_3') \\ A_4A_0 & \Rightarrow & \text{l'autre motif entre } A_1 \text{ et } A_2 \text{ est nécessairement } A_3A_4 \\ A_4A_3 & \Rightarrow & u_4 = 0 \text{ (contradiction)} \end{array}$$

où W_2 est le temps d'attente du robot avant l'exécution de A_2 dans le motif A_1yA_2 et W_3' est le temps d'attente du robot avant d'exécuter A_3 si y contient l'activité A_3.

De la même manière, entre A_2 et A_3, on peut avoir les activités A_0, A_0, A_1, A_1 et A_4. Par conséquent, il existe un motif entre A_2 et A_3 avec moins de deux activités. Notons z ce motif qui peut être de la forme :

$$
\begin{array}{lcl}
\emptyset & \Rightarrow & W_3 = p_3 \\
A_0 & \Rightarrow & W_3 = max(0, p_3 - 6\delta) \\
A_1 & \Rightarrow & W_3 = max(0, p_3 - 4\delta) \\
A_4 & \Rightarrow & W_3 = max(0, p_3 - 4\delta) \\
A_0A_1 & \Rightarrow & W_3 = max(0, p_3 - 6\delta - p_1) \\
A_0A_4 & \Rightarrow & \text{l'autre motif entre } A_2 \text{ et } A_3 \text{ est nécessairement } A_0A_1 \\
A_1A_0 & \Rightarrow & u_1 = 0 \text{ (contradiction)} \\
A_1A_4 & \Rightarrow & W_3 = max(0, p_3 - 8\delta) \\
A_4A_0 & \Rightarrow & \text{l'autre motif entre } A_2 \text{ et } A_3 \text{ est nécessairement } A_0A_1 \\
A_4A_1 & \Rightarrow & W_3 = max(0, p_3 - 8\delta)
\end{array}
$$

où W_3 est le temps d'attente du robot avant d'exécuter A_3 dans le motif A_2zA_3. On peut remarquer que l'occurrence de A_3 dans A_1yA_2, si y contient A_3, peut être l'occurrence de A_3 qui finit le motif A_2zA_3 (ce qui implique que $W_3 = W_3'$). Ceci ne peut pas se produire si $y = A_0A_3$. Dans ce cas, $z = A_1A_0$ ce qui implique que $u_1 = 0$. On a $T_W(C_2) \geq W_2 + W_3 + p_1 + p_4$ ou $T_W(C_2) \geq W_2 + W_3 + W_3' + p_1 + p_4$ si W_3 et W_3' ne viennent pas de la même occurrence de A_3. Nous étudions différentes bornes possibles pour $W_2 + W_3$ puis nous décrivons le cas pertinent pour chaque valeur de y et de z.

- **Cas 1** si $W_2 + W_3 \geq \max(0, p_2 - 8\delta) + \max(0, p_3 - 8\delta)$ alors $W_2 + W_3 + p_1 + p_4 \geq T_W(\pi_3) + T_W(\pi_6)$;

- **Cas 2** si $W_2 + W_3 \geq \max(0, p_3 - 6\delta - p_1, p_2 - 6\delta - p_4)$ alors $W_2 + W_3 + p_1 + p_4 \geq T_W(\pi_2) + T_W(\pi_7)$;

- **Cas 3** si C_2 contient $A_2A_0A_1A_3$ et $A_0A_1A_3A_0A_2$ alors C_2 contient la séquence $A_2A_0A_1A_3A_0A_2$. Il reste à placer les activités A_1, A_3 et deux fois A_4. Donc, entre la deuxième occurrence de A_2 et le A_3 suivant, on ne peut avoir que A_4 ou A_1A_4 ou A_4A_1 et donc $W_2 + W_3 + W_3' \geq \max(0, p_2 - 8\delta) + \max(0, p_3 - 8\delta)$ qui est résolu dans les cas 1.

- **Cas 4** si C_2 contient $A_1A_3A_4A_2$ et $A_2A_4A_1A_3A_4$ alors, de la même manière, entre l'autre occurrence de A_1 et le A_2 suivant, on ne peut avoir que A_0 ou A_0A_3 ou A_3A_0 ce qui est également résolu dans le cas 1.

Le tableau suivant indique un cas pertinent, pour chaque valeur possible de y et de z. Par exemple, pour $z = A_1$ et $y = A_3A_4$, on a $W_2 = max(0, p_2 - 6\delta - p_4 - W_3')$ et $W_3 = max(0, p_3 - 4\delta)$ où W_3' peut être W_3. Donc $T_W(C_2) \geq max(0, p_2 - 6\delta - p_4, p_3 - 4\delta)$. Comme $p_3 - 4\delta \geq p_3 - 6\delta$, les attentes W_2 et W_3 vérifient le cas 2.

	12	0102	132	142	01032	01302	1342
23	1	1	1	1	1	1	2
203	1	inc.	1	1	inc.	inc.	2
213	1	1	2 *	1	1	1	2 *
2434	1	1	1	inc.	1	1	inc.
2013	2	inc.	2 *	2	inc.	3 *	2 *
21434	1	1	1	inc.	1	1	inc.
24134	1	1	1	inc.	1	1	4 *

où "inc." signifie que les deux motifs sont incompatibles et "*" signifie que l'activité A_3 peut être commune aux deux motifs.

Pour toutes les valeurs compatibles de y et de z, nous avons montré que $T_W(\pi_3) + T_W(\pi_6) \leq T_W(C_2)$ ou que $T_W(\pi_2) + T_W(\pi_7) \leq T_W(C_2)$. Par conséquent, x est une W-couverture pyramidale de C_2 ce qui implique que (C4) est vraie. □

Appendix B

Stabilité des permutations pyramidales

Dans cette annexe, nous considérons les cellules à deux, trois et quatre machines. Nous analysons la stabilité (définie dans la Section 2.2 page 34) et nous calculons les temps de cycle des permutations pyramidales.

Rappelons que $w_h^i(\pi)$ est le vecteur (matrice de taille $1 \times m$) des temps d'attente du robot à la machine M_h pendant la i-ème exécution de la permutation pyramidale π. Dans la suite, nous supprimons π dans $w_h^i(\pi)$ lorsqu'il n'y a pas de confusion possible. Pour prouver la stabilité de la permutation pyramidale π, nous prouvons que les suites $w_h^i(\pi)$ sont stationnaires, et indépendantes des temps d'usinage restants $P^0(\pi)$, c'est-à-dire que les temps de cycle et la stabilité des permutations pyramidales ne dépendent pas de la phase d'initialisation.

B.1 Propriétés générales

Nous présentons d'abord quelques lemmes et propriétés afin de simplifier les preuves de stabilité et d'éliminer quelques cas triviaux ou généraux.

Définition 4 *Une suite de nombres u_i est dite stationnaire s'il existe un i_0 tel que pour tout $i \geq i_0$, on a $u_i = u_{i+1}$.*

Lemme 16 *Soit u_i et v_i des suites entières définies par*

$$u_i = \max(0, a - v_i) \ ; \qquad v_i = \max(0, b - u_{i-1}) \ ; \qquad u_0 \geq 0;$$

où u_0, a et b sont des nombres entiers. Il existe un indice i_0 tel que, pour tout $i \geq i_0$, les suites u_i et v_i sont constantes et vérifient $u_i + v_i = \max(0, a, b)$.

Démonstration En remplaçant v_i par son expression dans u_i on obtient

$$\begin{aligned} u_i &= \max(0, a - \max(0, b - u_{i-1})) \\ &= \max(0, a + \min(0, u_{i-1} - b)). \end{aligned}$$

Nous considérons trois cas.

Cas 1 $a \leq 0$

Comme $\min(0, u_{i-1} - b) \leq 0$ et $a \leq 0$, on a $u_i = \max(0, a + \min(0, u_{i-1} - b)) = 0$, pour tout i.

Cas 2 $a > 0$ et $a \geq b$

Si $u_i \geq b$ pour un i, alors $u_j = a$ pour tout $j \geq i$. Donc, si $u_0 \geq b$ alors $u_i = a$ pour tout i. Si $u_0 < b$ alors $u_1 = u_0 + a - b$; $u_2 = u_0 + 2(a - b)$... La séquence augmente de $a - b$ à chaque itération. Si $a > b$, alors u_i dépasse b pour un i suffisamment grand. Sinon (c'est-à-dire $a = b$), $u_i = u_0$ pour tout i.

Cas 3 $a > 0$ et $a < b$

On a $v_i = \max(0, b + \min(0, v_{i-1} - a))$. En échangeant a et b dans la démonstration précédente, on obtient $v_i = b$ pour i assez grand.

La Figure B.1 résume les différentes valeurs de u_i et v_i (pour i assez grand), en fonction de a et b. On déduit, à partir de cette figure, que $u_i + v_i = \max(0, a, b)$.

□

Considérons maintenant une permutation pyramidale très simple : l'identité, $\pi_0 = (A_0 A_1 \dots A_m)$.

Proposition 15 *L'identité* $\pi_0 = (A_0 A_1 \dots A_m)$ *est stable et son temps de cycle est donné par*

$$T(\pi_0) = \sum_{h=0}^{m} (2\delta_h + \epsilon_h^u + \epsilon_{h+1}^l) + \sum_{h=1}^{m} p_h.$$

Démonstration Toutes les activités de π_0 sont montantes. Donc $m_h(\pi_0) = 2$ pour tout $h = 0, 1 \dots m$. Par conséquent, le temps de trajet vérifie

$$T_T(\pi_0) = \sum_{h=0}^{m} m_h(\pi_0)\delta_h = 2\sum_{h=0}^{m} \delta_h.$$

Or, le temps de chargement/déchargement vérifie

$$T_L(\pi_0) = \sum_{h=0}^{m} (\epsilon_h^u + \epsilon_{h+1}^l).$$

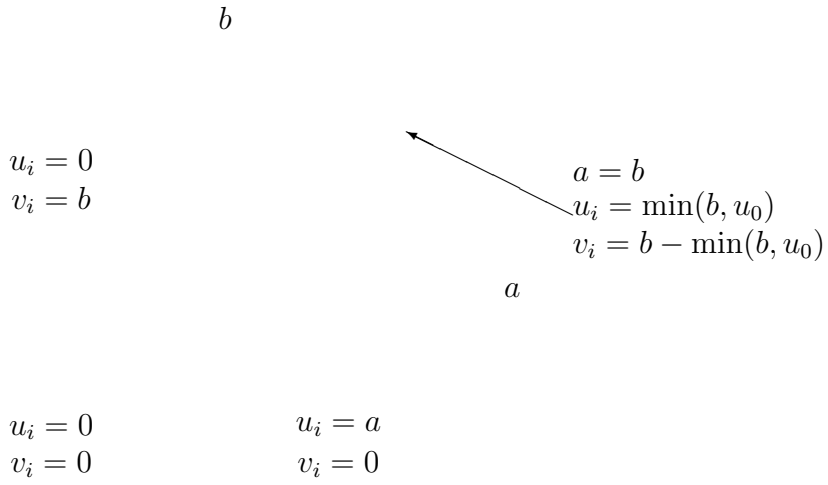

Figure B.1: Valeurs des suites u_i et v_i

Comme A_h suit immédiatement A_{h-1} pour $h = 1, 2 \dots m$, on a $w_h^i(\pi_0) = p_h$, pour tout $h = 1, 2 \dots m$ et pour tout i. Par conséquent, les temps d'attente $w_h^i(\pi_0)$ sont constants et

$$T_W(\pi_0) = \sum_{h=1}^{m} p_h.$$

Ceci conclut la preuve. □

B.2 Le cas de deux machines

Dans une cellule à deux machines, il y a $\mu+1 = 2^{m-1} = 2$ permutations pyramidales : l'identité $\pi_0 = (A_0 A_1 A_2)$ et la permutation descendante $\pi_1 = (A_0 A_2 A_1)$.

Proposition 16 *Dans une cellule à deux machines les deux permutations pyramidales sont stables et leurs temps de cycles sont donnés par*

$$\begin{aligned} T(\pi_0) &= \sum_{h=0}^{2} (2\delta_h + \epsilon_h^u + \epsilon_{h+1}^l) + p_1 + p_2 \ ; \\ T(\pi_1) &= \sum_{h=0}^{2} (2\delta_h + \epsilon_h^u + \epsilon_{h+1}^l) + 2\delta_1 \end{aligned}$$

$$+max(0, p_1 - 2\delta_1 - 2\delta_2 - \epsilon_2^u - \epsilon_3^l, p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l).$$

Démonstration Pour π_0, la Proposition 16 est une conséquence immédiate de la Proposition 15. Il reste à calculer le temps de cycle de π_1 et de prouver que π_1 est stable. On peut remarquer que, pour π_1, on a

$$\begin{aligned} w_1^i &= max(0, p_1 - 2\delta_1 - 2\delta_2 - \epsilon_2^u - \epsilon_3^l - w_2^i) \ ; \\ w_2^i &= max(0, p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l). \end{aligned}$$

Donc, w_2^i est constant. Comme w_1^i ne dépend que de w_2^i, il est aussi constant. Donc, π_1 est stable. De plus,

$$\begin{aligned} T_W(\pi_1) &= w_1^i + w_2^i \\ &= max(0, p_1 - 2\delta_1 - 2\delta_2 - \epsilon_2^u - \epsilon_3^l, p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l). \end{aligned}$$

Nous savons que

$$T_L(\pi_1) = \sum_{h=0}^{2} (\epsilon_h^u + \epsilon_{h+1}^l).$$

Dans π_1, seule A_1 est descendante. On a donc $T_T(\pi_1) = 2\delta_0 + 4\delta_1 + 2\delta_2$. Ceci conclut la preuve. □

B.3 Le cas de trois machines

Les temps de cycles des permutations pyramidales, dans une cellule à trois machines, ont été calculés pour la première fois dans [66]. Rappelons que $\Delta_h = 2\delta_{h-1} + 2\delta_h + \epsilon_{h-1}^u + \epsilon_h^l + \epsilon_h^u + \epsilon_{h+1}^l$.

Proposition 17 *Dans une cellule à trois machines, les quatre permutations pyramidales sont stables et leurs temps de cycles sont donnés par*

$$\begin{aligned} T(\pi_0) &= \Delta_1 + \Delta_3 + p_1 + p_2 + p_3 \\ T(\pi_1) &= \Delta_1 + \Delta_3 + 2\delta_2 + p_1 + \\ &\quad max(0, p_3 - \Delta_1 - 2\delta_2 - p_1, p_2 - 2\delta_2 - 2\delta_3 - \epsilon_3^u - \epsilon_4^l) \\ T(\pi_2) &= \Delta_1 + \Delta_3 + 2\delta_1 + p_3 + \\ &\quad max(0, p_1 - 2\delta_1 - \Delta_3 - p_3, p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l) \\ T(\pi_3) &= \Delta_1 + \Delta_3 + 2\delta_1 + 2\delta_2 + max(0, p_1 - 2\delta_1 - 2\delta_2 - \Delta_3, \\ &\quad p_2 - 2\delta_1 - 2\delta_2 - \Delta_1 - \Delta_3 + \Delta_2, p_3 - 2\delta_1 - 2\delta_2 - \Delta_1) \end{aligned}$$

où $\pi_0 = (A_0A_1A_2A_3)$, $\pi_1 = (A_0A_1A_3A_2)$, $\pi_2 = (A_0A_2A_3A_1)$, *et* $\pi_3 = (A_0A_3A_2A_1)$.

Démonstration Pour π_0, la Proposition 17 est une conséquence immédiate de la Proposition 15. Nous savons que, pour une permutation pyramidale π, si A_h est

montante, alors $m_h(\pi) = 2$ et si A_h est descendante, alors $m_h(\pi) = 4$. Comme $T_T(\pi) = \sum_{h=0}^{m} m_h(\pi)\delta_h$, on calcule facilement le temps de trajet. Dans une cellule à trois machines, on a $T_L(\pi) = \sum_{h=0}^{3}(\epsilon_h^u + \epsilon_{h+1}^l)$ pour toute permutation pyramidale π. Il reste à calculer les temps d'attente $w_h^i(\pi_j)$ pour $j = 1, 2, 3$ et de prouver qu'ils sont constants.

Pour π_1, on a

$$\begin{aligned} w_1^i &= p_1 \ ; \\ w_2^i &= \max(0, p_2 - 2\delta_2 - 2\delta_3 - \epsilon_3^u - \epsilon_4^l - w_3^i) \ ; \\ w_3^i &= \max(0, p_3 - \Delta_1 - 2\delta_2 - w_1^i). \end{aligned}$$

Ces équations peuvent être réécrites comme suit

$$\begin{aligned} w_3^i &= \max(0, p_3 - \Delta_1 - 2\delta_2 - p_1) \ ; \\ w_2^i &= \max(0, p_2 - 2\delta_2 - 2\delta_3 - \epsilon_3^u - \epsilon_4^l - \max(0, p_3 - \Delta_1 - 2\delta_2 - p_1)). \end{aligned}$$

Comme w_1^i, w_2^i et w_3^i peuvent être exprimés indépendamment de i, il sont constants. Les temps d'attente vérifient

$$\begin{aligned} T_W(\pi_1) &= w_1^i + w_2^i + w_3^i \\ &= p_1 + max(0, p_3 - \Delta_1 - 2\delta_2 - p_1, p_2 - 2\delta_2 - 2\delta_3 - \epsilon_3^u - \epsilon_4^l). \end{aligned}$$

Pour π_2, les temps d'attente vérifient

$$\begin{aligned} w_1^i &= \max(0, p_1 - 2\delta_1 - \Delta_3 - w_2^i - w_3^i) \ ; \\ w_2^i &= \max(0, p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l) \ ; \\ w_3^i &= p_3. \end{aligned}$$

Comme pour π_1, les temps d'attente de π_2 sont constants et on obtient

$$\begin{aligned} T_W(\pi_2) &= w_1^i + w_2^i + w_3^i \\ &= p_3 + max(0, p_1 - 2\delta_1 - \Delta_3 - p_3, p_2 - 2\delta_0 - 2\delta_1 - \epsilon_0^u - \epsilon_1^l). \end{aligned}$$

Pour π_3, l'argument est différent car nous ne pouvons pas supprimer simplement i dans l'expression de w_h^i. On a

$$\begin{aligned} w_1^i &= \max(0, p_1 - 2\delta_1 - 2\delta_2 - \Delta_3 - w_2^i - w_3^i) \ ; \\ w_2^i &= \max(0, p_2 - 2\delta_1 - 2\delta_2 - \Delta_1 - \Delta_3 + \Delta_2 - w_3^i) \ ; \\ w_3^i &= \max(0, p_3 - 2\delta_1 - 2\delta_2 - \Delta_1 - w_1^{i-1}). \end{aligned}$$

Nous décrivons, pour w_1^i, w_2^i et w_3^i, une transformation qui mène aux suites u_i et v_i et à une application du Lemme 16. Les temps d'attente de π_3 vérifient :

$$\begin{aligned} w_1^i &= \max(0, p_1 - 2\delta_1 - 2\delta_2 - \Delta_3 - (w_2^i + w_3^i)) \\ w_2^i + w_3^i &= \max(w_3^i, p_2 - 2\delta_1 - 2\delta_2 - \Delta_1 - \Delta_3 + \Delta_2) \\ &= \max(0, p_2 - 2\delta_1 - 2\delta_2 - \Delta_1 - \Delta_3 + \Delta_2, p_3 - 2\delta_1 - 2\delta_2 - \Delta_1 - w_1^{i-1}). \end{aligned}$$

Ces temps d'attente peuvent être écrits sous la forme de u_i et v_i avec les valeurs suivantes

$$\begin{array}{ll} \alpha = \max(0, p_2 - 2\delta_1 - 2\delta_2 - \Delta_1 - \Delta_3 + \Delta_2) \ ; & \\ a = p_1 - 2\delta_1 - 2\delta_2 - \Delta_3 - \alpha \ ; & u_i = w_1^i \ ; \\ b = p_3 - 2\delta_1 - 2\delta_2 - \Delta_1 - \alpha \ ; & v_i = w_2^i + w_3^i - \alpha. \end{array}$$

Comme u_i et v_i sont constant à partir d'un certain rang, w_1^i et $w_2^i + w_3^i$ deviennent aussi constants. Comme w_1^i est stationnaire et w_3^i ne dépend que de w_1^i, le temps d'attente w_3^i est aussi stationnaire. Comme $w_2^i + w_3^i$ et w_3^i sont stationnaires, w_2^i est aussi stationnaire. Donc, la permutation pyramidale π_3 est stable. Le Lemme 16 nous permet de trouver l'expression de $T_W(\pi_3)$:

$$\begin{array}{rcl} T_W(\pi_3) & = & w_1^i + w_2^i + w_3^i = u_i + v_i + \alpha = \max(0, a, b) + \alpha \\ & = & max(0, p_1 - 2\delta_1 - 2\delta_2 - \Delta_3, \\ & & p_2 - 2\delta_1 - 2\delta_2 - \Delta_1 - \Delta_3 + \Delta_2, p_3 - 2\delta_1 - 2\delta_2 - \Delta_1). \end{array}$$

Ceci conclut la preuve. □

B.4 Le cas de quatre machines

Nous considérons une cellule à quatre machines pour le cas régulier (pour alléger les calculs). Les résultats de stabilité peuvent être étendus au cas additif général.

Proposition 18 *Les permutations pyramidales sont stables et leurs temps de cycles sont donnés dans le Tableau 4.1, page 89.*

Démonstration Nous savons que [27] le temps de cycle de la permutation descendante π_7 satisfait $T(\pi_7) = 16\delta + 10\epsilon + \max_i(0, p_i - 12\delta - 6\epsilon)$. Donc π_7 est stable. Pour les autres permutations pyramidales, on peut calculer les temps d'attente de $\pi_j (j = 0, 1, \ldots, 6)$ et trouver les expressions suivantes

$$\begin{array}{ll} T(\pi_0) = 10\delta + 10\epsilon + w_1^i + w_2^i + w_3^i + w_4^i & \\ w_1^i = p_1 & w_2^i = p_2 \\ w_3^i = p_3 & w_4^i = p_4 \end{array}$$

$$\begin{array}{ll} T(\pi_1) = 12\delta + 10\epsilon + w_1^i + w_2^i + w_3^i + w_4^i & \\ w_1^i = p_1 & w_2^i = p_2 \\ w_3^i = \max(0, p_3 - 4\delta - 2\epsilon - w_4^i) & w_4^i = \max(0, p_4 - 8\delta - 6\epsilon - w_1^i - w_2^i) \end{array}$$

$$\begin{array}{ll} T(\pi_2) = 12\delta + 10\epsilon + w_1^i + w_2^i + w_3^i + w_4^i & \\ w_1^i = p_1 & w_2^i = \max(0, p_2 - 6\delta - 4\epsilon - w_3^i - w_4^i) \\ w_3^i = \max(0, p_3 - 6\delta - 4\epsilon - w_1^i) & w_4^i = p_4 \end{array}$$

$T(\pi_3) = 14\delta + 10\epsilon + w_1^i + w_2^i + w_3^i + w_4^i$
$w_1^i = p_1$ $\qquad w_2^i = \max(0, p_2 - 8\delta - 4\epsilon - w_3^i - w_4^i)$
$w_3^i = \max(0, p_3 - 10\delta - 6\epsilon - w_1^i - w_4^i)$
$w_4^i = \max(0, p_4 - 10\delta - 6\epsilon - w_1^i - w_2^{i-1})$

$T(\pi_4) = 12\delta + 10\epsilon + w_1^i + w_2^i + w_3^i + w_4^i$
$w_1^i = \max(0, p_1 - 8\delta - 6\epsilon - w_2^i - w_3^i - w_4^i)$ $\qquad w_2^i = \max(0, p_2 - 4\delta - 2\epsilon)$
$w_3^i = p_3$ $\qquad w_4^i = p_4$

$T(\pi_5) = 14\delta + 10\epsilon + w_1^i + w_2^i + w_3^i + w_4^i$
$w_1^i = \max(0, p_1 - 10\delta - 6\epsilon - w_2^i - w_3^i - w_4^i)$ $\qquad w_2^i = \max(0, p_2 - 4\delta - 2\epsilon)$
$w_3^i = \max(0, p_3 - 4\delta - 2\epsilon - w_4^i)$
$w_4^i = \max(0, p_4 - 10\delta - 6\epsilon - w_1^{i-1} - w_2^i)$

$T(\pi_6) = 14\delta + 10\epsilon + w_1^i + w_2^i + w_3^i + w_4^i$
$w_1^i = \max(0, p_1 - 10\delta - 6\epsilon - w_2^i - w_3^i - w_4^i)$
$w_2^i = \max(0, p_2 - 10\delta - 6\epsilon - w_3^i - w_4^i)$
$w_3^i = \max(0, p_3 - 8\delta - 4\epsilon - w_1^{i-1})$ $\qquad w_4^i = p_4$

Considérons, par exemple, la permutation pyramidale π_6. Afin de calculer w_2^i, nous considérons les événements entre la fin de la $(i-1)$-ème exécution de A_1 (quand une pièce est déposée sur M_2) et le début de la i-ème exécution de A_2 (quand la pièce est déchargée de M_2) [Figure B.2], c'est-à-dire

- le robot est en M_2 et va en M_0 [2δ unités de temps] ;
- le robot exécute A_0 [$\delta + 2\epsilon$ unités de temps] ;
- le robot va de M_1 à M_3 [2δ unités de temps] ;
- le robot attend que la pièce sur M_3 soit prête [w_3^i unités de temps] ;
- le robot exécute A_3 [$\delta + 2\epsilon$ unités de temps] ;
- le robot attend que la pièce sur M_4 soit prête [w_4^i unités de temps] ;
- le robot exécute A_4 [$\delta + 2\epsilon$ unités de temps] ;
- le robot va de M_5 à M_2 [3δ unités de temps].

Donc, le temps d'attente en M_2 pendant la i-ème exécution de π_6 est $w_2^i = \max(0, p_2 - 10\delta - 6\epsilon - w_3^i - w_4^i)$.

Pour π_0, La Proposition 18 est une conséquence immédiate de la Proposition 15.

Pour π_1, on a

$$\begin{aligned} w_1^i &= p_1 \qquad w_2^i = p_2 \\ w_3^i &= \max(0, p_3 - 4\delta - 2\epsilon - w_4^i) \\ w_4^i &= \max(0, p_4 - 8\delta - 6\epsilon - w_1^i - w_2^i) \\ &= \max(0, p_4 - 8\delta - 6\epsilon - p_1 - p_2). \end{aligned}$$

Donc, w_4^i est stationnaire et par conséquent, w_3^i l'est également. Donc π_1 est stable. Le temps de cycle de π_1 est

$$\begin{aligned} T_W(\pi_1) &= w_1^i + w_2^i + w_3^i + w_4^i \\ &= p_1 + p_2 + \max(0, p_3 - 4\delta - 2\epsilon - w_4^i) + w_4^i \\ &= p_1 + p_2 + \max(w_4^i, p_3 - 4\delta - 2\epsilon) \\ &= p_1 + p_2 + \max(0, p_3 - 4\delta - 2\epsilon, p_4 - 8\delta - 6\epsilon - p_1 - p_2). \end{aligned}$$

Les preuves pour π_2 et π_4 sont similaires.

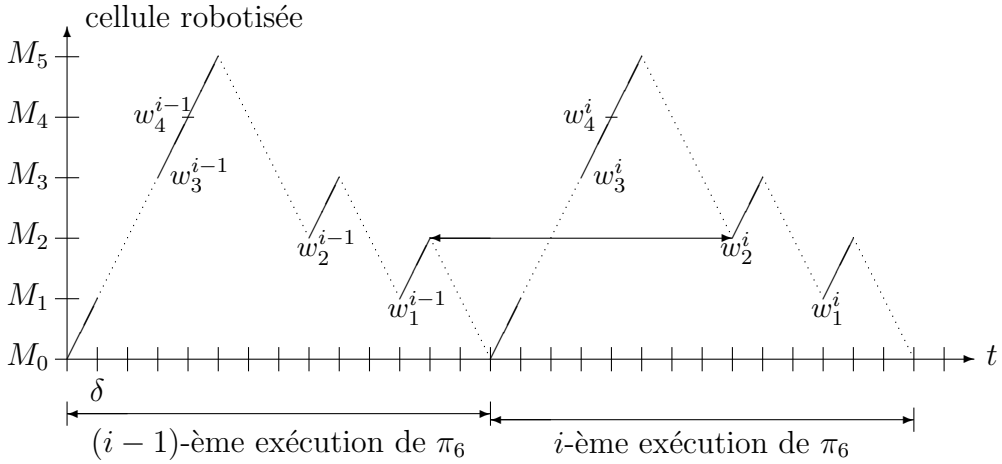

Figure B.2: La permutation pyramidale $\pi_6 = (A_0A_3A_4A_2A_1)$

Nous montrons que les temps d'attente des permutations pyramidales π_3, π_5 et π_6 peuvent être écrites sous la forme de u_i et v_i.

Les temps d'attente de π_3 sont :

$$\begin{aligned} w_1^i &= p_1 \\ w_2^i &= \max(0, p_2 - 8\delta - 4\epsilon - w_3^i - w_4^i) \\ w_3^i + w_4^i &= \max(0, p_3 - 10\delta - 6\epsilon - w_1^i - w_4^i) + w_4^i \\ &= \max(w_4^i, p_3 - 10\delta - 6\epsilon - p_1) \\ &= \max(0, p_3 - 10\delta - 6\epsilon - p_1, p_4 - 10\delta - 6\epsilon - p_1 - w_2^{i-1}). \end{aligned}$$

On peut vérifier qu'ils peuvent être écrits sous la forme de u_i et v_i avec les valeurs suivantes :

$$\begin{aligned} &\alpha = \max(0, p_3 - 10\delta - 6\epsilon - p_1); \\ &a = p_2 - 8\delta - 4\epsilon - \alpha; && u_i = w_2^i; \\ &b = p_4 - 10\delta - 6\epsilon - p_1 - \alpha; && v_i = w_3^i + w_4^i - \alpha. \end{aligned}$$

Par conséquent, w_2^i est stationnaire, donc w_1^i l'est aussi. Ceci implique que w_3^i est également stationnaire. Pour obtenir $T_W(\pi_3)$, il suffit d'appliquer la transformation

précédente :

$$\begin{aligned}
T_W(\pi_3) &= w_1^i + w_2^i + w_3^i + w_4^i \\
&= w_1^i + u_i + v_i + \alpha \\
&= p_1 + \max(0, a, b) + \alpha \\
&= p_1 + \max(0, p_2 - 8\delta - 4\epsilon - \alpha, p_4 - 10\delta - 6\epsilon - p_1 - \alpha) + \alpha \\
&= p_1 + \max(\alpha, p_2 - 8\delta - 4\epsilon, p_4 - 10\delta - 6\epsilon - p_1) \\
&= p_1 + \max(0, p_3 - 10\delta - 6\epsilon - p_1, p_2 - 8\delta - 4\epsilon, p_4 - 10\delta - 6\epsilon - p_1).
\end{aligned}$$

Les preuves pour π_5 et pour π_6 sont similaires avec les transformations suivantes pour π_5,

$$\begin{array}{l}
\alpha = \max(0, p_2 - 4\delta - 2\epsilon, p_3 - 4\delta - 2\epsilon, p_2 + p_3 - 8\delta - 4\epsilon); \\
a = p_1 - 10\delta - 6\epsilon - \alpha; \quad u_i = w_1^i; \\
b = p_4 - 10\delta - 6\epsilon - \alpha; \quad v_i = w_2^i + w_3^i + w_4^i - \alpha.
\end{array}$$

et pour π_6,

$$\begin{array}{ll}
\alpha = \max(0, p_2 - 10\delta - 6\epsilon - p_4); & \\
a = p_1 - 10\delta - 6\epsilon - p_4 - \alpha; & u_i = w_1^i; \\
b = p_3 - 8\delta - 4\epsilon - \alpha; & v_i = w_2^i + w_3^i - \alpha.
\end{array}$$

□

Résumé : Ce travail concerne la production cyclique de pièces identiques dans un flow-shop robotisé. La Conjecture des 1-cycles, proposée par Sethi et al., suppose que le taux maximum de production peut être atteint en répétant un cycle particulier qui produit une seule pièce. Cette conjecture simplifie la recherche du meilleur cycle de production.

Nous présentons de nouvelles preuves (approche par les graphes et approche algébrique) de la validité de cette conjecture pour des cellules à 2 et 3 machines et nous montrons qu'elle est fausse à partir de 4 machines. Nous délimitons ensuite plus précisément son cadre de validité en imposant des restrictions sur les paramètres : distances inter-machines égales ou temps d'usinage égaux.

Puis, nous étudions d'autres formes de cellules robotisées en relaxant des contraintes de la cellule robotisée de base. La première variante est l'association de l'entrée et de la sortie de la cellule. Nous proposons quelques remarques sur la recherche du meilleur cycle de production. La deuxième variante est le HSP (*Hoist Scheduling Problem*) : le temps pendant lequel une pièce peut rester sur une machine admet une borne supérieure. Nous montrons que des propriétés des cellules robotisées ne peuvent pas être étendues au HSP. La troisième variante est l'ajout de zones de stockage entre les machines. Nous montrons que la Conjecture des 1-cycles est vraie et nous analysons le gain par rapport à une cellule sans stockage. Enfin, nous supposons que les distances inter-machines sont quelconques. Nous montrons que trouver le meilleur cycle de production d'une pièce est un problème NP-complet.

Ce travail a permis de résoudre complètement une conjecture ouverte depuis 1989 et de décrire l'influence de la relaxation de certaines contraintes des cellules robotisées sur la recherche du meilleur cycle de production. La principale perspective est, pour les cas où la conjecture est fausse, de trouver le meilleur cycle de production.

Mots-clés : ordonnancement, flow-shop, système de transport, production cyclique, conjecture des 1-cycles.

Scheduling in robotic cells

Abstract: This work concerns the cyclic production of identical parts in a robotic flow-shop. The one-cycle Conjecture, proposed by Sethi et al., claims that the repetition of the best one-unit production cycle will yield the maximum throughput rate in the set of all possible robotic moves. The interest of this conjecture is that it simplifies the search of the best production cycle.

We present new proofs (using graphs and an algebraic approach) of the validity of this conjecture for 2- and 3-machine cells and we prove that it is false for cells with 4 and more machines. We define more precisely its validity range by imposing restrictions on the parameters: equidistant machines or equal processing times on all machines.

Then we study other forms of robotic cells by relaxing some constraints of the initial cell. The first modification is the association of the input and the output station. We make some remarks on the validity of the one-cycle Conjecture and on the search of the best production cycle. The second modification is the HSP (Hoist Scheduling Problem): the time a part can stay on a machines is upper bounded. We prove that some interesting properties of the robotic cells cannot be extended to the HSP. The third modification is the addition of buffer space at the machines. We show that the one-cycle Conjecture is true and we analyze the gain in production compared to a system with no buffer space. Finally, we suppose that the inter-machine distances can be of any form. Then the conjecture is false and we prove that finding the best one-unit production cycle is NP-hard.

This work allowed us to settle completely a conjecture open since 1989 and to describe how certain relaxed constraints of robotic cells influence the search of the best production cycle. The most challenging open question is, whenever the one-cycle Conjecture is false, to find the overall best production cycle.

Key words: scheduling, flow-shop, material handling system, cyclic production, one-cycle conjecture.

Laboratoire Leibniz-IMAG, 46 avenue Félix Viallet, 38031 Grenoble Cedex, FRANCE.