Scheduling handling resources : robotic flowshops with circular layout and a practical railway problem

Florence Thiard

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Florence THIARD

Thèse dirigée par **Nadia BRAUNER**,
codirigée par **Nicolas CATUSSE**

préparée au sein du **Laboratoire des Sciences pour la Conception, l'Optimisation et la Production de Grenoble** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Ordonnancement de ressources de transports : flow-shops robotisés circulaires et un problème pratique de gestion ferroviaire

Scheduling handling resources: robotic flowshops with circular layout and a practical railway problem

Thèse soutenue publiquement le **21 novembre 2017**,
devant le jury composé de :

Monsieur Alessandro AGNETIS
Professeur, Université de Sienne, Italie, Président
Madame Safia KEDAD-SIDHOUM
Maître de conférences, Université Pierre et Marie Curie, Rapportrice
Monsieur Ameur SOUKHAL
Professeur, École polytechnique de l'université de Tours, Rapporteur
Monsieur Fabien MANGIONE
Maître de conférences, Grenoble INP, Examinateur
Madame Nadia BRAUNER
Professeur, Université Grenoble Alpes, Directrice de thèse
Monsieur Nicolas CATUSSE
Maître de conférences, Grenoble INP, Co-encadrant de thèse

The idea is like grass. It craves light,
likes crowds, thrives on crossbreeding,
grows better for being stepped on.

Ursula K. Le Guin, *The Dispossessed*

Contents

Remerciements

Une thèse, c'est comme l'éternité : un peu long, surtout vers la fin. En revanche, c'est sans doute moins solitaire. Dans ces pages, déjà trop longues et malgré tout trop courtes pour remercier toutes celles et ceux qui ont eu leur rôle à jouer, inutile de chercher un plan, fut-il chronologique, thématique, encore moins didactique. Lecteur, lectrice : débrouillez-vous.

Commençons par la fin : je tiens à remercier Safia Kedad-Sidhoum et Ameur Soukhal d'avoir accepté de rapporter ce travail et de lire avec attention mes divague-ations (c'était facile) en pleine période de rentrée. Merci à Fabien Mangione d'avoir replongé dans sa petite enfance pour faire partie de mon jury, et merci à Alessandro Agnetis de m'avoir fait l'honneur de présider celui-ci. Grazie anche per il tuo accoglio, et per avere corretto i miei congiuntivi. A tou·te·s, merci pour vos remarques, vos questions, votre intérêt et votre bienveillance qui ont contribué à faire de la clôture de ces années un bon souvenir.

On recommencerait presque ? Quand même pas. Revenons tout de même aux débuts, et donc à Nadia Brauner et Nicolas Catusse. Nadia, la toute première fois que tu m'as parlé de cellules robotisées avec des pièces de 5 centimes, elles tournaient déjà en rond : ce n'est que plus tard que j'ai compris pourquoi. Merci de m'avoir fait confiance quand je ne l'aurais pas fait moi-même, et d'avoir inlassablement tenté de me convaincre que ma thèse est une thèse (maintenant, je vais bien finir par le croire). Merci aussi d'avoir patiemment repris mes « s » et mes calculs – pour les virgules, par contre, je ne céderai pas. Merci Nicolas pour ton enthousiasme contagieux, tes intuitions bluffantes et ton pragmatisme salutaire. Merci à tou·te·s les deux pour votre disponibilité touchante (malgré la rude compétition avec Caseine). Et surtout, si finir une thèse, c'est avoir appris, alors grâce à vous celle-ci est réussie.

Quatre ans de thèse dont deux d'enseignement, et il faut croire que j'y ai pris goût : merci à toute l'équipe du département MMI, Vanda Luengo, Jean-François Berdjugin et Vincent Lestideau avec qui j'ai eu le plaisir de travailler, et merci bien sûr à mes étudiant·e·s toujours là pour me rappeler que « mais Madame, c'est pas logique ! » – c'était pour voir si vous suiviez.

* * *

Avant l'histoire, il y a la préquelle : sans remonter à ma première pâte à sel (encore que...), je tiens à remercier Madame Cornet pour avoir fait des « maths » un joli monde traversé de petites et grandes histoires parsemées de ponctuations interrogatives, suspendues ou enthousiastes, sans jamais sacrifier une rigueur qui m'a fait gagner beaucoup de temps par la suite.

Suivent les ami·e·s qui ont balisé mon chemin jusqu'ici. Cécile, bien sûr : après tout c'est pour le plaisir de ta conversation (et pas pour la science ni les beaux yeux de Justin !) que j'ai atterri dans mon premier cours de « Combinatoire et Graphes ». Pour reprendre tes mots, on n'aurait pas cru à l'époque, il y a 8 ou 9 ans d'ici, finir un jour en haut de la tour Zamansky (ou dans mon cas, plus modestement, du bâtiment F) « pompettes, et doctoresses » ! Tu remercieras aussi Thomas pour ses « conseils de vieux » que je n'ai, bien évidemment, jamais suivis (je ne dis pas que ce n'était pas un tort). Je n'oublie pas Sophie, à qui je pense encore à chaque fois que ma bouilloire chauffe pour une tasse de thé nocturne (sur ces dernières années, ça veut dire « souvent » !), Laure sans qui la prépa est beaucoup moins drôle, Mélissa avec qui j'ai (énergiquement !) fait mes armes et mes griffes de bébé-informaticienne et qui s'entête malgré tout à me supporter régulièrement le temps d'une pause en bord de mer.

A propos de binôme, Eli : sans nos démêlées avec la coloration de Juliette, la décomposition de Benders et même une transformée de Laplace égarée, sans nos doutes partagés, je ne me serais sans doute pas dit que peut-être, après tout... une thèse... tu crois qu'on peut ? Enfin, voilà le reste de la dream-team : Hugo, co-auteur, co-équipier, co-ADOC, co-procrastinateur, co- à peu près tout sauf bureau, en fait, et Lucas, co-bureau adoptif aux talents de portraitiste incontestés (mais pas que). Depuis les après-midi à élaborer des plans de carrière alternatifs entre deux preuves par déchargement sur les tableaux de l'UFR IM^{2}AG, on a pas mal grandi, et les plans ont changé[1]. Je suis heureuse d'avoir commencé et de terminer cette histoire à vos côtés – à ce propos, Hugo : c'est bon, tu peux soutenir maintenant.

* * *

Me voilà avec vous revenue aux portes de G-SCOP : merci Maria et Marie-Laure d'y avoir accompagné mes premiers pas – quelque peu maladroits. Merci à mes co-bureau successifs et simultanés, d'abord Boris et Saleh Eddine tout au bout du bout du 2C, puis Michaël qui a placé la barre drôlement haut, enfin Kean et ses implacables 10 pages/semaine qui n'empêchent pas de rigoler pour autant, et puis Nicolas qui entre ses deux trains, ses quatre stagiaires et ses vingt problèmes à gérer trouve toujours le temps pour une petite bataille de chansons agaçantes à se coincer en tête.

Hors des bureaux, il y a des couloirs animés, une kfet pleine à craquer (et parfois d'autres lieux mieux irrigués : selon votre génération, Saint-Vincent, Re-

[1] Alors, ce labo indépendant dans le Trièves, on s'y met quand ?

naissance ou Sun Valley). J'ai pu y faire la connaissance, au gré de discussions plus ou moins scientifiques, plus ou moins échauffées, d'un paquet de personnalités. Permanentes : entre autres Bernard, Hadrien[2], Louis, Nicolas (encore un), Olivier, Vincent (préviens-moi quand même si tu casses ma conjecture), Zoltán... et moins permanentes : les matheux qu'il ne faut pas placer côte-à-côte dans les dîners où l'on cause, Quentin, Rémi, Yohann, les coincheur·se·s qui m'ont tolérée malgré tout et tout·e·s les autres, Alex, Clément, Élodie, Franck, Justine, Lisa, Matthieu, Nico la tête me tourne et j'en oublie. Merci Tom de lire mes mails-presque-sérieux, merci Aurélie et Lucie grâce à qui je me suis parfois sentie moins seule, et bon courage à mon p'tit frère Florian pour venir à bout du cas 4.2.42bis (hein que c'est chouette, les preuves d'ordonnancement ?).

Pour rester en famille, un merci à part (désolée) pour mon très-grand frère de thèse, Pierre avec et sans initiale, d'avoir relevé ces années de nouveaux paysages, de jolies découvertes, et même de (vrais) petits lutins dansant entre les lignes de LaTeX[3]. De m'avoir, aussi, prêté un peu de sa bienveillance quand la mienne me faisait défaut. Et puisqu'il a bien fallu penser à « l'après », merci à Julie pour ses conseils (parfois par procuration), et sur une autre note, la franchise d'un sourire contagieux (même si ses Gourdon, eux, me regardent un peu de travers).

Un labo peuplé uniquement de chercheur·se·s (fussent-illes enseignant·e·s) sombrerait vite dans le chaos. Merci donc à celles et ceux qui font que pourtant, il tourne : Marie Jo grâce à qui je sais maintenant (entre autre) charger correctement une agrafeuse, Fadila, Myriam, ainsi que les membres passés et présents des services informatique et financier – en particulier Amandine et Christine pour leur aide lors de l'organisation des journées G-SCOP.

* * *

Repassons la porte. Merci aux Grenobloi·se·s, Téo qui n'a *jamais* abandonné la bière du vendredi soir malgré mes réponses sporadiques en 3ème année et plus, Quentin pour la pastèque (désolée, pour cette soutenance-ci ce n'était plus la saison), Kévin pour nos trop rares synchronisations compensées par la densité des discussions, Irina, toujours une bonne raison pour faire le trajet jusqu'au campus, et bien sûr Seb, malgré ses infidélités avec la Savoie et l'Ardèche.

Dézoomons encore un peu : de leurs labos français ou européens (ou d'un bateau en Antarctique), un grand merci aux habituées du topic doctorat, et tout particulièrement au « bureau » des Licornes à Paillettes (voilà, c'est dit) : Anaïs, Cori, Dom, Emilie, Léa... de râlages en verbalisations, de soutien moral en digressions, on n'a pas *toujours* fait avancer la Science et la Recherche, mais depuis qu'ils jouent gentiment entre eux, nos Gonzague, Charles-Henri, Gérard et autres Raichu nous fichent un peu plus la paix, non ?

[2]Si, si, Hadrien, tu es bien un permanent.

[3]Même si pour rédiger ce manuscrit, il a fallu se passer de `\faeries`...

Allez, tant qu'on y est, traversons l'Atlantique : Emeline, tu rentres dans tellement de catégories à la fois (et en même temps aucune) que je ne sais pas où te placer. Merci d'être là depuis maintenant 13 ans, pour ta fenêtre ouverte malgré le décalage horaire, pour ton fauteuil encore mieux qu'un canapé-lit et tes tentatives de nourrir « une thésarde en rédaction » ; merci à Émile de te maintenir en forme parce que sinon je serais quand même super embêtée. Et puis courage : on dirait pas comme ça, mais c'est bientôt fini.

* * *

Retour aux sources (géographiquement instables). « Mieux vaut Thiard que jamais », dit l'adage, et cette thèse n'y échappe pas. Merci à ma maman qui ne s'est pas trop fichue de moi en me voyant traîner de mois d'août en mois d'août un chapitre 1 toujours à commencer, à mon papa sans qui je n'aurais pas poussé si loin la métaphore maritime (et me serais contentée en bonne Grenobloise de parler de montagnes et de cols), et à tou·te·s deux de m'avoir envoyée dans le vaste monde équipée, en sus d'une trousse à couture, d'un tournevis et de conseils plus ou moins avisés, du réconfort de se savoir soutenue. Renaud, Jérôme, Sophie : si je peux parler 45 minutes d'affilée déguisée en madame presque sans rougir, c'est sans doute grâce à Rires en Folies. Et si ces remerciements commencent à déborder, c'est bien sûr à cause de la propension familiale au discours plutôt qu'à la synthèse. Merci enfin Grand-maman de m'avoir accueillie pendant mes études parisiennes, et ne t'inquiètes pas : je crois que cette thèse n'a mis personne au chômage (sauf moi). Je n'oublie pas non plus les additions, Anne, Céline, J.-B. (ni la multiplication, Alice qui m'a gentiment attendue), ainsi que la famille adverse : merci Félix, Sophie pour votre accueil chaleureux et vos encouragements, et Sacha d'avoir supporté mes humeurs de fin de rédaction.

* * *

Il y en aurait bien d'autres (Guillaume pour m'avoir aidé à résoudre ce problème crucial : « le jury, deuxième ou troisième rang ? », MamieCaro pour la meilleure citation de tableau au monde, et même un plombier bio pas étranger à ma décision de commencer une thèse...), mais pour ne pas précéder ce manuscrit d'un roman, il va falloir conclure. Je finirai donc en mêlant ici et peut-être ailleurs, passé, présent et je l'espère futur : merci David, d'avoir été, d'être et de rester, celui auprès de qui j'ai envie de me réveiller chaque matin et de m'endormir chaque soir (mais jamais sur mes lunettes). Eh, monsieur, tu sais quoi ? T'es quand même chouette.

Introduction et guide de lecture

Ordonnancer consiste à décider quand et comment allouer des ressources (machines, processeurs...) à un ensemble de tâches, en respectant certaines contraintes (capacité, précédence...), afin d'optimiser un ou plusieurs objectif ou indicateurs de performance (temps d'achèvement, retards, coûts de traitement...).

Dans le cas de ressources physiques, si les tâches sont constituées de plusieurs opérations s'effectuant sur différentes ressources, on est amené à se poser la question du transport des objets ou matériaux impliqués entre les ressources affectées. La prise en compte du transport génère alors des contraintes propres, liées à l'utilisation éventuelle de ressources spécifiques pour assurer les déplacements, et à la disposition physique des ressources de traitement : temps de déplacements, capacités et disponibilités des ressources de transport, connexions et accessibilité d'une ressource de traitement depuis une autre...

Dans ce travail, nous nous intéressons à deux problèmes associant ordonnancement et transport, dans deux contextes différents : la production manufacturière (Chapitres 1 à 4) et la gestion ferroviaire (Chapitre 5).

Le premier problème concerne les cellules robotisées : il s'agit d'un modèle d'atelier en ligne (flow-shop), où le transport des pièces entre les machines est assuré par un bras robotisé. Les pièces à produire doivent être récupérées à l'entrée de la cellule, être transportées sur chacune des machines pour y être traitées, et enfin déposées à la sortie de la cellule. On cherche à déterminer une programmation cyclique du robot afin d'optimiser le débit de la cellule. En fonction du type de robot utilisé et de la disposition des machines, on peux distinguer deux types de configuration : linéaire et circulaire. Dans le cas de la configuration linéaire, l'entrée et la sortie de la cellule sont séparées, le robot ne peut se rendre de l'une à l'autre sans traverser l'ensemble de la cellule (cas, par exemple, d'un robot circulant sur un rail rectiligne). Dans le cas circulaire, l'entrée et la sortie de la cellules sont jointes ou très proches, et le robot peut faire le tour complet de la cellule (cas de machines disposées en cercle et desservies par un bras robotisé situé au centre) : ceci permet d'économiser du temps de transport, au prix, semble-t-il, d'une plus grande complexité. En effet, les principaux résultats théoriques établis pour la configuration linéaire ne s'appliquent pas à la configuration circulaire, et

celle-ci demeure moins bien comprise. Notre objectif est donc d'en d'améliorer la compréhension théorique ; nous nous intéressons pour cela à un cas particulier, celui des cellules régulières et équilibrées.

Les Chapitres 1 à 4 sont consacrés à ce premier problème. Le Chapitre 1 présente une introduction aux cellules robotisées : nous y exposons les concepts nécessaires à la modélisation d'une cellule robotisée, les principales questions théoriques ainsi qu'un état de l'art sur le sujet ; nous y précisons ensuite le cadre choisi pour les travaux menés dans les chapitres suivants. Le Chapitre 2 présente des outils utilisés par la suite pour l'analyse des programmations (cycles) possibles : sa lecture est recommandée avant d'aborder les Chapitres 3 et 4. Ces deux chapitres abordent, dans le cas des cellules robotisées circulaires régulières et équilibrées, deux problèmes théoriques classiques (introduits dans le Chapitre 1) concernant les cycles de production d'une pièce exactement (1-cycles) : la conjecture des 1-cycles et le problème des meilleurs 1-cycles. Dans le Chapitre 3 nous établissons certaines propriétés des cycles performants, qui nous permettent de fermer le problème du meilleur 1-cycle pour des cellules de petite taille (jusqu'à 8 machines) ; nous donnons également un contre-exemple à la conjecture des 1-cycles pour 6-machines. Dans le Chapitre 4 nous nous intéressons aux cellules arbitrairement grandes (plus de 8 machines), et cherchons à dégager des propriétés structurelles des programmations dominantes. Sans trop en dévoiler, disons qu'on y parle de vagues – et qu'il vaut mieux bien s'amarrer.

Le Chapitre 5 est consacré au second problème, concernant la gestion ferroviaire – il peut donc être lu indépendamment des précédents. Il s'agit d'un problème industriel présenté dans le cadre du Challenge EURO/ROADEF 2014 ; ce chapitre présente notre contribution. On s'intéresse à la gestion des trains au sein d'une gare. Une gare est constituée de ressources de différents types : ressources de traitement (plate-forme pour le traitement des arrivées et des départs, ressources de maintenance), de stockage (parkings et voies de garage) et enfin ressources de transition (aiguillages). Le déplacement des trains sur et entre ces ressources obéit à des contraintes liées aux ressources elles-mêmes (capacité, temps de parcours), à leur disposition (adjacence et connexité des ressources), et aux autres trains (collisions et blocages). La formulation originale mêle un grand nombre de sous-problèmes (affectation des départs et arrivées, affectation des plateformes, routage, maintenance des trains, jonctions et disjonction des rames), contraintes et objectifs (nombre de départs satisfaits, coûts de performance et d'occupation des ressources, de maintenances...) reflétant la complexité du problème en pratique. Nous simplifions la formulation originale afin de nous concentrer sur le problème du routage, pour lequel nous proposons un algorithme par propagation de contraintes multi-intervalles, suffisamment souple pour permettre, couplé à un algorithme d'affectation glouton, d'obtenir une solution réalisable.

Chapter 1

An introduction to robotic cells

Scheduling consists in deciding how and when to allocate available *resources* (servers) to a set of *tasks* (or jobs) to be performed, subject to certain constraints (capacity, precedence between tasks...) in order to optimize one or more objective function (total completion time, cycle time, lateness, cost...). In the context of manufacturing production, the tasks may be parts to be produced, and the resources the machines on which the processing takes place.

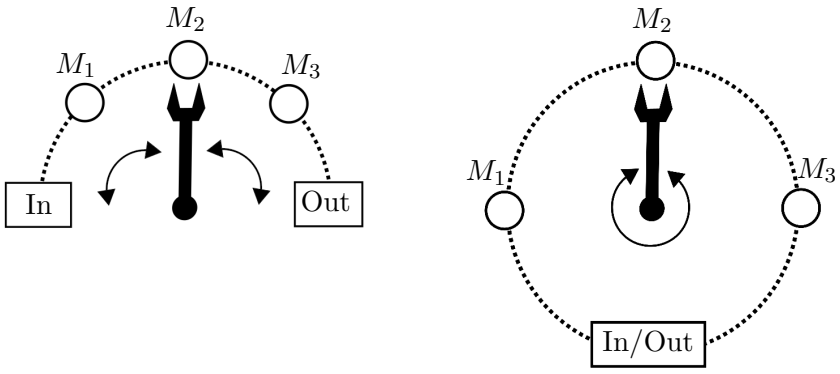
Classical scheduling theory distinguishes between scheduling problems on *parallel servers*, and on *dedicated servers*. The second class is itself divided into 3 categories: job-shops, open-shops and flow-shops. Several operations have to be performed on a job: in a flow-shop, the operations sequence is fixed and is the same for all the jobs; in a job-shop, it is still fixed but job-dependent; in an open-shop, the operations sequence is up to the scheduler to decide. In terms of manufacturing production, in a flow-shop, all parts are processed on the machines in the same order. The minimization of the makespan in a flow-shop with 3 machines or more is NP-Hard, with and without buffer between machines (Garey et al., 1976, Hall and Sriskandarajah, 1996). For 2 machines, it is polynomial both for the case with unlimited buffering capacity (Johnson, 1954) and the no-buffer case (Gilmore and Gomory, 1964, Hall and Sriskandarajah, 1996). However, it is NP-hard when the buffer is of limited non-zero capacity (Papadimitriou and Kanellakis, 1980)). Pinedo (2008) and Leung et al. (2004) are, for instance, reference books on classical scheduling theory and algorithms; the web page `http://www2.informatik.uni-osnabrueck.de/knust/class/` maintains an inventory of classical scheduling problems and their complexity status.

However, classical scheduling theory (and, specifically, the classical flow-shop model) does not fully account for modern manufacturing systems. Additionally to the processing machines, auxiliary resources must be considered for example human operators in the case of semi-automated systems, and handling resources such as automated guided vehicles or mobile robots.

Section 1.1 presents robotic cells, one of the models which take into account handling resources (namely handling robots). Section 1.2, Section 1.3 and Section 1.4 present a literature review on three types of robotic cells: with no-wait condition, with unbounded waiting times, and with flexibility. Section 1.5 states the scope of our work and the notations used throughout this document.

1.1 Robotic cells: generalities

A robotic cell consists of a flow-shop where the m machines $M_1, \ldots, M_m$ are served by one or several robotic arms. The cell also contains an input buffer, where parts to be processed are usually supposed to be available in infinite quantity, and an output buffer of similarly infinite capacity where fully processed parts are to be transferred. The input and output buffer are generally represented by two auxiliary machines $M_0 = \text{In}$ and $M_{m+1} = \text{Out}$. Figure 1.1 shows two examples of three-machine cells with two different layouts, semi-circular (where the input and input buffers are separated) and circular (where the input and output buffers coincide).



(a) semi-circular layout (b) circular layout

Figure 1.1: Three-machine robotic cells

Each part must be taken from the input buffer, transferred successively on each machine (M_1, then M_2, etc. until M_m) to complete their processing, then it is transferred to the output buffer.

The model was first introduced by Asfahl (1985) to describe a production cell for truck differentials, while Sethi et al. (1992) provided the first formal study for small dimension cells.

In the general case, where multiple part types must be produced, two types of decisions must be made: sequencing the parts, and scheduling the robot moves. In the case where only one type of part is to be produced, there is no decision to make for the part sequencing: the problem reduces to finding an optimal robot move schedule.

In the classical model, the cell is equipped with a single robot able to handle only one part at a time (single gripper), the machines are specialized and can only carry out one type of operation (non-flexibility), and machines are without buffer facility. In this set-up, a *blocking* constraint is enforced: the robot has to be empty in order to pick-up a part, and any machine (except for the input and output buffers) has to be unloaded before it can be loaded again. Hall and Sriskandarajah (1996) presents a survey of flow-shop problems with blocking constraints.

Other models allow to relax this blocking constraint, by using different types of robots (multi-gripper robot, robot with swapping ability), or adding machine buffers.

The following two sections describe the different models for processing policies and travel policies in robotic cells. They are largely extracted from the survey (Brauner, 2008).

1.1.1 Processing and pick-up criterion

The processing starts as soon as a part is loaded on a machine. The *processing time* represents the minimum time a part must remain on a machine. If all parts are different, p_{ij} denotes the processing time of part j on machine M_i. If one wants to produce one large batch of *identical parts*, the processing times of the parts on machine M_i are $p_i = p_{ij}$. In the *balanced* case[1], all processing times are equal, i.e., $p_{ij} = p_j$, or, for the production of identical parts, $p_i = p$ for all i.

Once the part is finished, two policies may apply. In the *no-wait* case, the part must be removed immediately from the machine and transferred to the following machine. In the *unbounded* case (also denoted as *free pick-up*), the part can remain on the machine waiting for the robot.

A classical extension of those two cases is the so-called Hoist Scheduling Problem (HSP); in this case, the processing time is described by an interval, and the no-wait policy applies. This means that the time part j may remain on machine M_i lays in the specified interval. This applies to chemical treatments were the machines correspond to chemical baths. The HSP is NP-hard even for identical parts and very simple (additive, as defined in the next section) configurations of cells (Crama and van de Klundert, 1997b).

[1] Also called *proportionate flowshop*.

Loading and unloading times

During the loading of a part onto a machine from the robot or the unloading of a part from a machine onto the robot, both the machine and the robot are occupied. The time necessary to perform these operation is often assumed to be part- and machine-independant, and denoted by ϵ. Lehoux-Lebacque (2007) showed that in bufferless robotic cells, loading and unloading times can be dismissed without loss of generality.

Note however that the problem of robotic cells with sequence-dependent loading times has been investigated recently (see e.g. Majumder and Laha, 2016, Zarandi et al., 2013).

1.1.2 Travel metric and layout

There are different classical metrics for the travel times of the robot depending on the physical configuration of the cell and on the characteristics of the robot. We denote by $\delta_{i,j}$, the travel time of the robot (empty or loaded) from M_i to M_j. It is natural, and in practice desirable, to assume that the travel times satisfy the classical properties of a metric:

- the travel time from a machine to itself is zero, that is, $\delta_{i,i} = 0$;
- the travel times satisfy the triangle inequality, that is, $\delta_{i,k} + \delta_{k,j} \geq \delta_{i,j}$ for all i, k and j;
- The travel times are symmetric, that is $\delta_{i,j} = \delta_{j,i}$ for all i and j.

Travel times verifying those three assumptions are called *general* (or sometimes "euclidean", e.g. in Dawande et al. (2005b)). Special configurations of cells have been studied in the literature.

For *additive* times, to travel between distant machines, the robot passes through all intermediate machines and its speed is constant. This metric is the most popular since, in practice, it is applicable if the machines are on a circle or on a line and if the cell is dense (the robot does not have time to speed up between distant machines). In this case, one has the triangle equality $\delta_{i,j} = \delta_{i,k} + \delta_{k,j} = \sum_{k=i}^{j-1} \delta_{k,k+1}$ for any $i \leq k < j$. By symmetry, this also defines $\delta_{i,j}$ for $i > j$. Some authors (e.g. Hall et al. (1997), Logendran and Sriskandarajah (1996)) consider an extension of this case having a constant gain γ when travelling between distant machines, i.e., $\delta_{i,j} = \sum_{k=i}^{j-1} \delta_{k,k+1} - (j - i - 1)\gamma$ for $i < j$. We assume $\gamma = 0$.

In the *regular* case, the machines are equidistant and we denote $\delta_{i,i+1} = \delta$. This constraint can be added to the additive constraint (as in the seminal paper by Sethi et al. (1992)).

For *constant* travel times (introduced in Dawande et al. (2002)), δ is the time for the robot to travel between any two distinct machines M_i and M_j: $\delta_{i,j} = \delta$.

Finally, depending on the type of robot used, the machines and the input and output stations can be disposed in several ways. Two main configurations are used: on the one hand, linear or semi-circular layouts (Figure 1.1a), where the input and output buffers are separated and located respectively at each end of the line (Crama and van de Klundert, 1997a), and on the other hand, circular layouts (Figure 1.1b), where the machines are arranged in a circle, with the input and output buffers either occupying the same spot ($M_0 = M_{m+1}$), or very close (Jung et al., 2015, Rajapakshe et al., 2011). In this case, the robot chooses the shortest path along the circle formed by the machines. Then, the travel times verify $\delta_{i,j} = \min(\sum_{k=i}^{j-1} \delta_{k,k+1}, \sum_{k=j}^{m} \delta_{k,k+1} + \sum_{k=0}^{i-1} \delta_{k,k+1})$ for any $i < j$. This can be combined with the additive and regular constraints.

1.1.3 Throughput optimisation and cyclic scheduling

A possible optimization goal is to minimize the makespan of a lot (total completion time of the lot). Makespan optimization in robotic flow-shops is addressed for example by Carlier et al. (2010), Hurink and Knust (2001), Kise (1991), Soukhal and Martineau (2005), Yang et al. (2016).

However, for large-scale production, the maximization of the *throughput rate* of the cell, with infinite horizon, may be considered as the objective. In this context, it seems operationally relevant to prefer a cyclic programmation. This means that the robot repeats indefinitely the same move sequence, each iteration leaving the cell in the same state, with the same machines loaded, the same machines empty and the robot in the same place. Indeed, cyclic programmations are dominant (Dawande et al., 2005a), which means that, for any set of parameters, there always exists an optimal programmation which is cyclic. The elementary sequence is called a cycle.

Robot move sequences (and thus cycles) can be described using the concept of activities, defined by Sethi et al. (1992). Activities are elementary robot moves defined as such: for $i \in \{0 \dots m\}$, activity A_i refers to the following sequence of events:

1. The robot unloads a part from M_i;
2. The robot travels to M_{i+1};
3. The robot loads the part onto M_{i+1}.

In the classical model, a cycle is entirely specified by a sequence of activity. Cycles which produce exactly k parts are called k-cycles (or k unit cycles). Thus, 1-cycles are particular cycles which produce one part exactly: during one iteration,

exactly one part enters the cell at M_0, and one processed part leaves the cell at M_{m+1}.

Maximizing the throughput of the cell equates to minimizing its *cycle time*, or more accurately, its cycle time in regards to the number of part produced by one iteration (k for a k-cycle). The cycle time might not be the same for each repetition of the cycle, therefore we consider the *long run average cycle time*. More detail on the computation of the long run average cycle time and an example can be found in Chapter 2, page 32.

Sethi et al. (1992) establish that in an m-machine cell, the number of 1-cycles is $m!$. Using the concept of activity, all the 1-cycles can be obtained by considering the permutations of the m activities $A_1 \dots A_m$ (Sethi et al., 1992). Formally, 1-cycles are of the form $\pi = A_0, A_{i_1}, A_{i_2} \dots A_{i_m}$ where $(i_1, i_2 ... i_m)$ is a permutation of $\{1, 2...m\}$. More generally, k-cycles are characterized as the sequences of activity such that each activity occurs exactly k times, and between two consecutive occurrences (in a cyclic sense) of an activity A_i with $i \notin 0, m+1$, there is exactly one occurrence of the activities A_{i-1} and A_{i+1} (Crama and van de Klundert, 1999). Among 1-cycles, two common (and intuitive) permutations are the identity (or uphill) permutation $A_0 A_1 A_2 \dots A_m$, and the downhill permutation $A_0 A_m A_{m-1} \dots A_1$. They will be described in more detail in Chapter 2.

One-cycles are easy to describe and enumerate using the concept of activities, as they are exactly the permutations of the m activities (Sethi et al., 1992). They are also easier to implement operationally. As a consequence, it is convenient to restrict the possible move sequences to 1-cycles only (this problem is known as the best 1-cycle problem). But does this allow to find an optimal sequence? Sethi et al. (1992) formulate the 1-cycle conjecture:

***Conjecture* 1.1** (1-cycle conjecture Sethi et al. (1992))**.** *The set of 1-cycles is dominant (for any set of parameters, there always exists a 1-cycle which is optimal).*

This conjecture has been settled in some cases, which we expose later in Section 1.2 and Section 1.3.

For the case where multiple part types are to be produced, the cyclic production of a Minimal Part Set (MPS) is considered. A MPS is defined as a minimum cardinality set such that the relative proportion of each type of part is the same as in the overall demand (Dawande et al., 2007). A MPS-cycle is a cycle during which all the parts in a MPS are produced: all parts in the MPS enter the cell and all parts in the MPS exits the cell. The idea of 1-cycles can be extended to multiple part type production by considering Concatenated Robot Move (CRM) cycles: CRM-cycles are MPS-cycles consisting of the repetition of a same 1-cyclic sequence as many times as there are parts in the MPS.

When considering relations between cycles, we will use these meanings of "optimality" and "dominance":

- A cycle c is *optimal* for a given instance (or set of instances) if it performs at least as well as any possible cycle for this instance (or for each instance of the set).
- A cycle c is *optimal within* a set of cycles S for a given instance or set of instances, if it performs at least as well as any cycle in S on this instance or set of instances (in other words, it is optimal for the problem restricted to the cycles in S).
- A set of cycles S is *dominant* if, for every possible instance, an optimal cycle can be found in S.
- A set of cycles S is *dominant over* a set of cycles T, or S *dominates* T, if for every possible instance, an optimal cycle *within* T can be found in S.
- A cycle c *dominates* a cycle g (or g is dominated by c) for a given set of instances, if c performs at least as well as g for all these instances.
- A cycle c is *non-dominated* by a set of cycle S on a given set of instances if there exist an instance within this set for which c *strictly* outperforms every cycle in S.

Several synthetic efforts provide useful insights about cycling scheduling in robotic cells. Levner et al. (2010) present an overview of the complexity status of cyclic scheduling problem, including robotic cyclic scheduling. Dawande et al. (2007) have written a book on throughput optimization in robotic cells both with single and dual gripper robots. Crama et al. (2000) present a survey on bufferless robotic cells and Brauner (2008) on the cyclic production of identical part in classical robotic cells. Bagchi et al. (2006) present a survey of TSP-based approaches for contemporary flow-shop problems, including the robotic flow-shop model.

In the following, we present an overview of the existing work on robotic cells with no-wait condition (Section 1.2), cells with unbounded waiting times (free-pickup) (Section 1.3), and finally, cells with flexibility (Section 1.4). In this last case, the allocation of operations to machines has to be decided, jointly with the robot move sequence and, if relevant, the part sequencing. We do not develop the more complex case of cells with interval or time window condition (HSP). Remember that in this configuration, the problem is NP-hard even in simple cases (Crama and van de Klundert, 1997b); existing literature focuses on Mixed Integer Programming (MIP) formulation, branch and bound and heuristic algorithms (see e.g. Kats et al., 2008, Phillips and Unger, 1976, Zhou et al., 2012). Unless otherwise

specified, the work presented here deals with cyclic optimization. We also do not develop the case of hybrid robotic flow-shops with parallel machines at each stage ((see e.g Elmi and Topaloglu, 2014, Geismar et al., 2004, Soukhal, 2001, Zabihzadeh and Rezaeian, 2016).

1.2 No-Wait condition

Production constraints may require for the parts to be picked up immediately after the end of their processing (Levner et al. (1997)). This classical scheduling constraint can model, for example, the case where the processing stations are chemical baths which must last a precise amount of time (Song et al. (1993)). It is referred to as *no-wait condition*, *constant processing times* or *zero-width processing time windows*. Note that for a given instance and move cycle, there might not exist a schedule satisfying the no-wait constraint: contrary to the unbounded case, feasibility must be considered. In the case of identical part production, a schedule is completely specified by the input time of the parts from which the robot move sequence can be deduced (Agnetis, 2000).

This section presents major results on no-wait robotic cells. All the models considered have a linear layout (where the input and output buffer are separated, as in Figure 1.1a), although results given for 2- and 3-machine cells can also be applied to circular layout cells.

1.2.1 Solved and open problems for small-dimension cells

Agnetis (2000) considers general production cycles for 2-machine cells producing multiple part-types, and 2- or 3-machine cells producing identical part types. Travel times are assumed to satisfy the triangle inequality. For identical part production, the author notes that the 2-machine no-wait case is similar to the 2-machine case with unbounded waiting times solved by Sethi et al. (1992). For the 3-machine case, Agnetis (2000) shows that the optimal cycle is either a 1-cycle or a 2-cycle and he provides an algorithm to compute it in constant time, using a forbidden region approach which is not easily extensible to a higher number of machines. The author gives an example of a non-dominated 2-cycle thus proving false the 1-cycle conjecture.

For multiple part-types production, Agnetis (2000) shows that the 2-machine case may be reduced to a classical 2-machine no-wait flowshop problem and hence solved in $\mathcal{O}(n \log n)$ using the Gilmore-Gomory algorithm (Gilmore and Gomory, 1964) for special TSP cases, while the NP-hardness of the problem for 3 machines and more ensues of the NP-hardness of the corresponding classical no-wait flowshop problem (Röck, 1984).

Subsequently, Agnetis and Pacciarelli (2000) focus on multiple part-types production on 3-machine cells and consider only repetitions of the same 1-cyclic sequences (defined as CRM cycles by Sriskandarajah et al. (2004)). For each of the six possible cycles, they study both the feasibility and optimization aspects of the part sequencing problem. For two of these six cycles, both problems are polynomially solvable. For two others, even the feasibility problem is shown to be unary NP-complete (this is proven using a reduction from Numerical Matching with Target Sum). For the last two, the feasibility problem equates to a classical 2-machine flowshop problem and can thus be solved by Gilmore-Gomory algorithm, but the optimization problem is a special case of the Traveling Salesman Problem (TSP) which complexity status is still open.

1.2.2 Identical part production

In this section, we present works addressing identical part production in cells of arbitrary dimension, whether restricted to 1-cyclic schedules or more general 2-cyclic or k-cyclic schedules.

1-cycles

Levner et al. (1997) and Kats and Levner (1997) consider identical part production in m-machine cells, with general travel-times, and provide polynomial algorithms for finding the best 1-cycle in the no-wait case. Kats and Levner (1997) provide an algorithm based on a forbidden intervals approach, running in $\mathcal{O}(m^4 \log m)$, which also solves the more general problem of reentrant robotic cells (where the number of operations to be performed is greater than the number of machines, meaning that the part has to reenter the cell to complete its processing). Levner et al. (1997) give an improved algorithm for the classical model, finding the best 1-cycle in $\mathcal{O}(m^3 \log m)$.

As 1-cycles are generally not optimal (as noted e.g. by Agnetis (2000)), subsequent works focus on multi-cyclic schedules.

2-cycles

Che et al. (2003) consider 2-cyclic robot move sequences in conditions similar to Levner et al. (1997). The polynomiality of finding an optimal 2-cycle had been conjectured by Levner et al. (1996). Che et al. (2003) derive a $\mathcal{O}(m^8 \log m)$ algorithm, subsequently improved by Chu (2006) to $\mathcal{O}(m^5 \log m)$. The authors' numerical experiments show that considering 2-cycles leads to a significant improvement of the optimal cycle time compared to considering only 1-cycles, which increases with the number of machines, and decreases when the time necessary to perform the robot

moves increases. With $m = 10$ fixed, the experiments show that the improvement is smaller for very small ($d_{i,i+1} < 2$ or very large $d_{i,i+1} > 7$) travel times. Kats and Levner (2009) extend this results to the non-euclidean case where the travel times don't satisfy the triangle inequality.

k-cycles

Kats et al. (1999) study cycles of arbitrary degree but only instances with integer input data, and integer schedules. They show that the number of integer solutions for a given degree k is finite, and provide a sieve procedure performing an exhaustive search in order to output an optimal k cycle. The algorithm runs in $\mathcal{O}(km^3m^k)$ and is therefore polynomial in m for a given k, but exponential in k.

Agnetis' conjecture

As stated in section 1.2.1, Agnetis (2000) proves that the optimal cycle for a 3-machine cells is either a 1-cycle or a 2-cycle. The author conjectures that this may be extended to an arbitrary number of machines; formally, that an optimal cycle can be found within the set of k-cycles with $k \leq (m-1)$. Mangione (2003) explores the validity of this conjecture for the special case of regular balanced cells and proves it is true for 4-machine cells, by studying possible configurations of the cells state graph. For m-machine cells, Mangione et al. (2003) states a conjecture on the structure of optimal cycles which, if proven, confirms Agnetis' conjecture, and proves it for some values of k and of the processing time p. Pavlov (2013) appears to prove the remaining cases for 5-machine cells[2].

To this day, Agnetis' conjecture is still open for no-wait robotic cells, even in the regular balanced case.

1.2.3 Multi-robot cells

More recent works consider cells with several robots. Robots can evolve on *parallel tracks* or on a *single track*. In the latter case, collision-avoidance constraints are added to the problem, while the former might be reduced to an extension of the single-robot problem coupled with an assignment problem.

Kats and Levner (2002) extend the results from Kats and Levner (1997) to the problem with several robots on parallel tracks, with robot-dependent travel-times, providing a $\mathcal{O}(m^3 \log m)$ algorithm for finding the best 1-cycle.

In the single track case, collision-avoidance constraints change the structure of the problem. Considering identical parts, identical robots and 1-cyclic schedules

[2]Due to a lack of Russian reading skills, this statement relies heavily on blind trust in automated translation tools.

only, Leung and Levner (2006) compute the minimum number of robots necessary to achieve a given cycle time in $\mathcal{O}(m^2)$. Their algorithm operates by defining a partial order on robot moves based on the notion of conflicting moves, then finding the longest chain in the corresponding graph. Based on this, they suggest an algorithm for computing the minimum number of robots necessary to achieve any cycle time in $\mathcal{O}(m^5)$, and note that it is not strictly decreasing with the cycle time. Assuming similar hypothesis, Che and Chu (2008) give a mathematical formulation of the problem and devise an algorithm which finds the best 1-cycle for a given number of robots R in $\mathcal{O}(m^6R)$. Liu and Jiang (2005) and Jiang and Liu (2007) consider a more general problem where the part flow does not necessarily coincide with the order of the machines in the cell, respectively with 2 robots and an arbitrary number of robots, and solve it in polynomial time (respectively $\mathcal{O}(m^4 log m)$ and $\mathcal{O}(m^6R)$, where R is the number of robots). These algorithms are based on the identification of threshold cycle times where the feasibility condition changes. Che et al. (2012) solve the problem in $\mathcal{O}(m^5)$ for any number of robots on the more general model of reentrant cells, although contrary to Jiang and Liu (2007), they do not enforce the upkeep of a safe distance between the robots.

Multi-cyclic production with several robots on parallel tracks are studied by Che and Chu (2005, 2009), respectively with 2 robots and an arbitrary number of robots. The proposed algorithms run in polynomial time in m for a given degree k, but are exponential in k.

The case of several robots on a single track is addressed by Che et al. (2011), who propose an algorithm for finding the best 2-cycle in $\mathcal{O}(m^7)$.

1.3 Unbounded waiting times

In this section we consider the case where the parts' pickup is not submitted to a time-window constraint. This is referred to as *unbounded waiting times* or *free pickup criterion*. Contrary to the no-wait case, once processed, a part might stay on a machine until the robot is free to pick it up. If there are no machine buffers, the machine is blocked until the part has been unloaded and transferred to the next machine. Models including machine buffers, robot buffers or robots with several gripper allow to release this constraint increasing the complexity of the problem. In the case of multiple part production, the problems to solve are the part sequencing and the robot moves scheduling. In the case of identical part production, contrary to the no-wait case, a schedule is not entirely specified by the input times of the parts: the robot moves sequence has to be determined.

1.3.1 Small scale cells

Sethi et al. (1992), Hall et al. (1997) and Hall et al. (1998) consider additive semi-circular 2- and 3-machine cells with linear layout. The identical part production problem on 2-machine cells is solved by Sethi et al. (1992): the authors show that the optimal robot move sequence is one of the two possible 1-cycles $A_0A_1A_2$ or $A_0A_2A_1$. The proof relies on the fact that while the robot is transferring a part between the 2 machines, both machines are empty, and this does not depend on the configuration of the cell. It is therefore also valid for the circular case. For production of multiple parts on 2-machine cells, Sethi et al. (1992) study the part sequencing problem when the robot move sequence is set to either one of the two possible CRM sequence: the repetition of $A_0A_1A_2$, or the repetition of $A_0A_2A_1$, and show it can be solved using the Gilmore and Gomory algorithm (Gilmore and Gomory, 1964). However, Hall et al. (1997) show that CRM cycles are not generally optimal, by giving a counter-example for the 2-machine case. In this case, finding an optimal robot move sequence consist in deciding which part are to be produced under which of the two possible sequence, and how to switch from one to another. The authors give a $\mathcal{O}(m^4)$ algorithm that outputs jointly the robot move schedule and the part sequencing leading to the optimal cycle time, using the Gilmore and Gomory algorithm as a subroutine, improved later on by Aneja and Kamoun (1999), who provide a $\mathcal{O}(m \log m)$ algorithm.

Sethi et al. (1992) define the 6 possible 1-cycles for 3-machine cells producing identical parts, derive their cycle time and provide a decision tree for determining the best one. However, for multiple part-type production, Hall et al. (1998) show that for 2 of the 6 possible CRM-cycle, the part sequencing problem is unary NP-complete, as well as the general part sequencing problem not restricted to any robot cycle (even for the regular case). For these two cycles, Chen et al. (1997) propose a branch-and bound algorithm (extensible to m machines), using the Gilmore and Gomory algorithm to provide a lower bound. Hall et al. (1997) show that for the 4 other CRM-sequences the part sequencing problem is polynomially solvable and deduce special cases for which the best 1-cycle problem is polynomially solvable. Kamoun et al. (1999) propose simple heuristics for some of the other cases. More recently, Zahrouni and Kamoun (2012) propose a constructive heuristic for the three-machine case not restricted to CRM cycles which outperform the latter in most of the tested instances. Kamalabadi et al. (2008) propose a metaheuristic approach using a swarm particle optimization algorithm and a formulation of the problem based on Petri nets.

Steiner and Xue (2005) extend the results of Sethi et al. (1992) and Hall et al. (1997) to the model of reentrant regular cells (as described for example in (Middendorf and Timkovsky, 2002)), showing that CRM cycle are not generally optimal for 2-machine reentrant cells, and that in 3-machine loop-reentrant cell (where the

processing must be finalized on the first machine), the part sequencing problem is strongly NP-hard for two of the possible CRM-cycles, and for the general problem not restricted to any robot move sequence.

1.3.2 Multiple part types

As for cyclic production of multiple part types, Sriskandarajah et al. (1998) extend the complexity results of Hall et al. (1997) to m machine cells. The authors classify the $m!$ possible CRM-cycles into four categories according to the structure and complexity of the associated part scheduling problem: trivial, polynomially solvable modeled as a special case of TSP, unary NP-hard modeled as a TSP, unary NP-hard without TSP structure. They show that only $2m - 2$ of the $m!$ possible cycle fall in the first two categories, while for all the remaining CRM-cycles, the part scheduling problem in NP-hard. Dawande et al. (2007) explains how to extend the heuristic proposed in Kamoun et al. (1999) to the m-machine case.

Makespan optimization Other works consider non-cyclic production, with the *makespan* of a given set of part as the objective function. Soukhal and Martineau (2005) propose a Mixed Integer Linear Programming (MILP) model (for additive travel times); they derive a lower bound and propose a genetic algorithm. Carlier et al. (2010) propose a decomposition of the problem into a flowshop problem (integrating transportation times and blocking problem) and a single machine problem with precedence constraints, both of which are solved using a branch and bound procedure. They also propose a genetic algorithm. Kharbeche et al. (2011) give a MILP formulation for the problem with general (euclidean) travel time, and new lower bounds. They describe a branch and bound algorithm, using a genetic algorithm to provide an initial upper bound.

1.3.3 Identical part production

In this section we present the works addressing identical part production in classical (single-gripper, bufferless) robotic cells.

Brauner et al. (2003) show, by a reduction from a TSP problem, that with general (euclidean) travel time, the problem is strongly NP-hard. Geismar et al. (2005b) provides a 4-approximation. More recently, a metaheuristic approach is presented by Elmi and Topaloglu (2016) for solving the problem with multiple robots.

In the following, we expose the results concerning, on the one hand, cells with constant and linear additive travel time (which are quite similar), and on the other hand, cells with circular additive time (as in Figure 1.1b).

Constant travel times and linear layout with additive travel times

Finding a best 1-cycle is proven to be polynomial in constant and linear additive cells respectively by Dawande et al. (2002) and Crama and van de Klundert (1997a). Crama and van de Klundert (1997a) result relies on the dominance of pyramidal permutation over 1-cycles. A 1-cycle is said to be pyramidal if its sequence of activities, starting with A_0, is strictly increasing, then strictly decreasing. Formally, a 1-cycle $\pi = (A_0, A_{i_1}, A_{i_2} \dots A_{i_m})$ where $(i_1, i_2 ... i_m)$ is a permutation of $\{1, 2...m\}$, is pyramidal if there is a p such that $1 \leq i_1 < \dots < i_p = m$ and $m > i_{p+1} > \dots > i_m \geq 1$. Crama and van de Klundert (1997a) prove their dominance over other permutation and deduce a $\mathcal{O}(m^3)$ algorithm for finding the best 1-cycle.

We have seen that, for the production of multiple part types, CRM-sequences are not dominant even for 2 machines (Hall et al. (1997)). However, in the case of production of identical parts, Hall et al. (1997) show that in 3-machine regular cells, 2-cyclic sequences are dominated by 1-cycles. This is extended to k-cycles by Crama and van de Klundert (1999), proving the 1-cycle conjecture for 3-machine additive cells. Brauner and Finke (1999) propose a shorter and simpler proof of this result. However, the conjecture was proven false for 4-machines by Brauner and Finke (2001), who provide a 3-cycle which performs better than any 1-cycle on a specified instance in a regular additive 4-machine cell. The conjecture formulated by Agnetis (2000) (for no-wait cells), stating that an optimal cycle can be found within the set of k-cycles with $k \leq m - 1$ was also proven false in 4-machine regular additive and constant cells by Brauner and G.Finke (2005). In the case of regular additive balanced cells, however, the 1-cycle conjecture is proven valid until 6 machines by Brauner and Finke (2001); Brauner (2008) claims the proof can be extended for $m \leq 15$.

Thus, 1-cycles are generally not dominant, and the complexity of finding the best k-cycle with $k \geq 2$ is still open, both in the constant and additive cases. However Geismar et al. (2008, 2005b, 2007) provide bounds on the performance of 1-cycles over k-cycles (and thus approximation algorithm with guaranteed performance). Geismar et al. (2005b) expose how to construct in linear time a 1-cycle with a performance ratio of 1.5, both in the constant and regular additive case. Geismar et al. (2008, 2007) improve this ratio respectively to $\frac{10}{7}$ in the regular additive case, and $\frac{9}{7}$ in the constant case, both ratios being tight.

Finally, the complexity status of the best 1-cycle problem in the balanced case (where all processing times are equal) is not settled by the algorithm proposed by Crama and van de Klundert (1997a). Indeed, in this case, the input consists in only 3 or 4 numbers: the number of machines m, the processing times p, the travel time between consecutive machines (for the regular additive case) or between any pair of machines (for the constant case), and possibly the loading/unloading time ϵ.

Thus, the algorithm of Crama and van de Klundert (1997a) is exponential with respect to the input size. Brauner (1999, 2008) address this issue. The author shows that the best k-cycle in the constant balanced case, and the best 1-cycle in the regular additive balanced case can be found in polynomial time. In the constant balanced case, the optimal k-cycle is one of the two simple permutations: the uphill cycle or the downhill cycle; so the 1-cycle conjecture is true in this case, for any number of machine.

Circular layout with additive travel times

Major results for classical linear cells rely on the dominance of pyramidal permutations over 1-cycles. However this *no longer holds* for cells with circular layout: a counter-example can easily be found (see for example Rajapakshe et al. (2011) or Brauner (1999)). Geismar et al. (2005a) note that in this case, the 1-cycle conjecture is still open, and no polynomial algorithm is known for finding the best 1-cycle.

The two machine case is solved, as the proof presented in Sethi et al. (1992) is also valid in the circular case.

Brauner (1999) determines the dominant 1-cycles for 3 and 4 machines cells in the regular additive case. The author introduces the odd-even production cycle, which takes advantage of the cell's circular arrangement by making the robot circle the cell twice, serving even machines on the first round and odd machines on the second, and conjectures its properties for an m-machine regular balanced cells.

For cells with $m \geq 3$, Rajapakshe et al. (2011) consider a circular layout with additive regular travel times and generic processing times. Note that the input and output buffers are not in the same place, but separated by δ. However we believe this difference does not qualitatively impact the results. The authors first prove the NP-hardness of the best 1-cycle problem for this layout, then seek to provide approximations in order to assess the performance of a circular layout as opposed to the linear one. They show the odd-even cycle provides a 2-approximation of the best 1-cycle, and provide a $\frac{5}{3}$ approximation by introducing a class of cycles called loop cycles, formed by alternating uphill and downhill segments[3]. For some cases, depending on the number of short, medium, and long processing times compared to the distance between consecutive machines, a $\frac{3}{2}$ approximation or even an optimal 1-cycle is given. No specific results are derived for the balanced case, in particular, the complexity status of the best 1-cycle problem is still open.

As a follow up to this work, Jung et al. (2015) extends these results to k-unit cycles.

[3]The authors appear to use a lower bound instead of the cycle time for the aforementioned odd-even cycle. In Chapter 2 we derive another expression of the cycle time.

Multi-function robot

Foumani et al. (2014) suggest to replace circular m-machine cells where odd-numbered machines perform similar operation by a $\frac{m-1}{2}$-machine cell served by a multifunction robot able to perform those operations. For 1, 2 and 3 machine cells, the author give the exact parameter for which this model allows to decrease the cycle-time. For m machines, they give a sufficient condition: the cycle time is improved if $p_i \leq \delta$ for all $1 \leq i \leq m$. They also give a TSP-based formulation of the problem.

1.3.4 Relaxing blocking constraints

Although the classical setup is generally studied on linear cells, models relaxing some of the blocking constraints under additive travel time assumption mostly consider circular layouts.

A way to release this constraint is the use of a dual gripper robot: the robot can handle two parts. The robot can then, for example, use one gripper to transport the part, keep one gripper empty to unload the next machine, then rotate its gripper to load it with the part. This kind of set up is perceived as much more complex than the single gripper one, as the number of possible cycles is much greater: Sethi et al. (2001) show there are 52 possible 1-cycles for the two machine case (as opposed to 2 cycles only for the single gripper case), 13 of which are non-dominated. Sriskandarajah et al. (2004) extend the study to the production of multiple part types, showing that for 6 of the 13 undominated cycles, the part sequencing problem is NP-hard. For those cases, Drobouchevitch et al. (2004) provide a heuristic with a $\frac{3}{2}$ performance ratio.

For the production of identical parts in m-machine cells, Sethi et al. (2001) shows that, if the gripper rotating time is smaller than the travel *and* processing times, then a very simple cycle which circles the cell once, stopping at each machine to unload then reload it is optimal within 1-cycles. Geismar et al. (2006) shows a similar result in the case of constant travel time, for which Geismar et al. (2008), provide a linear $\frac{9}{7}$-approximation for the optimal multi-unit cycle.

Another way to release the blocking constraint is to add input or output buffers at each machine. Drobouchevitch et al. (2010) study 1-cyclic production of identical parts in single gripper circular cells with input and output machine buffers. They show that this model allows no improvement compared to a single gripper cell with output machine buffers only. For buffers sufficiently close to the machines, they prove that 1-cycles are dominant.

Drobouchevitch et al. (2006) compare the performance of these two models. The study is limited to 1-cycles, and the authors consider both dual gripper cells with bufferless machines and single gripper cell with outputs buffers. The travel

times are additive and regular. First, they derive all active 1-cycles ("active" meaning that the robot doesn't carry a part without ever unloading it during a full cycle) and construct a polynomial algorithm for finding an optimal 1-cycle in the case where the gripper rotation time is less than the distance between two consecutive machines. Under this assumption, they show that a dual gripper cell performs at least as well as a single gripper cell with output buffers at each machine, and show an example where the improvement is strictly positive. The intuition is that although dual grippers and machine buffers serve the same purpose of temporarily storing parts to free the machines, the dual gripper solution is more time efficient because it potentially reduces the robot travel time.

Note that the equivalence of these two models for an arbitrary number of machines has later been shown by Dawande et al. (2008) for the production of multiple parts type, restricted to CRM cycles (which are formed by the repetition of 1-cycles), by establishing a bijection between the dominant CRM cycles for each model, based on the cycle time.

Jung et al. (2015) study regular additive circular cell with a dual gripper robot. As in Rajapakshe et al. (2011), the input and output stations are not at the same place, but separated by δ. Contrary to previous works, the study is not limited to 1-cycles: the authors seek to provide algorithms with performance guarantee for the problem of the optimal k-cycle. To this end, they introduce a class of 2-unit dual gripper cycles called epicyclic cycles providing a $\frac{3}{2}$-approximation, and an optimal k-cycle for some special cases.

Geismar et al. (2012) introduce a new model, a circular robotic cell served by a dual armed robot. They study the cyclic scheduling for 2 and 3 machine cells and compare the performance with single or dual gripper single armed robot cell.

Jolai et al. (2012) study circularly configured cell where the robot has an additional ability: it can swap the part it holds with the part loaded on the machine. Thus, the robot can load a part on a non-empty machine (and unload the previous part at the same time), which is not possible in the single-gripper classical case. The author derive a formulation of the 1-cycles cycle time depending on the waiting time at the machines, and give a lower bound on the cycle time of any k-cycle.

Gundogdu and Gultekin (2016) consider identical-part production on a 2-machine linear regular additive model with a self-buffered robot (where the machines are bufferless but the robot has a dedicated buffer). For a buffer of capacity one, they find the best 1-cycle; for a buffer of capacity 2, they give a $\frac{5}{4}$-approximation algorithm for a case where the loading time of the buffer is small enough (smaller than the travel time between consecutive machines). For buffer of infinite capacity, they study a particular class of cycles. They compare their model with the dual-gripper and swap model via a computational study.

1.4 Flexibility and cost optimization

In flexible installations, the operations constituting a job can be performed in any order (operational flexibility), and any machine is capable of performing any type of operation (process flexibility). The change of operation type performed for a machine may usually generate a cost. In this model, the allocation of operations to machines must be solved jointly with the robot move sequence, which makes it closest to the open-shop model.

Geismar et al. (2005a) consider a circular layout with input and output buffers at the same position, with regular additive travel times. In this study, the assignment of an operation type to a machine is decided before processing a lot, and cannot be changed until the end of the lot's processing. Therefore, once this assignment is decided, the setup is similar to a classical robotic cell. Note that this type of flexibility has no interest in the balanced case, where the processing times are equal for all operations. The authors provide a bound of the performance gain that can be achieved by changing the operation assignment for 3 and 4 machines, and prove that changing the assignment is never profitable for 2 machines. They note that the bound is the same for 3 and 4 machine cells, but are not sure it could be extended to cells with $m \geq 5$.

In subsequent works, machines are considered to have the ability to switch operations very quickly by changing tools, so that several operations for a single part may be performed on the same machine, if the adequate tools are available.

Akturk et al. (2005) consider the case of identical part production on 2 machines; they show that the optimal cycle is either a 1-cycle or a 2-cycle (disproving the 1-cycle conjecture for this case). Gultekin et al. (2006) extend the study to the case where the available tools are limited (tooling constraint): some operations can only be performed on the first machine, and some only on the second.

Gultekin et al. (2007, 2008, 2009) consider identical part production in linear cells, without tooling constraint. They introduce pure cycles, a class of production cycles which take advantage of the operational and process flexibility: each part is loaded on a single machine where it will be entirely processed. Gultekin et al. (2007) study the 3-machine case and show that pure cycles dominate all classical 1- and 2- unit cycles except one. Gultekin et al. (2008) focus on a particular simple pure cycle where the robots loads every machines from 1 to m, then unloads them in the same order, evaluating its performance compared to classical robot move cycle. They also compare the performance of this cycle in linear and circular 2-machine cells, showing that its cycle time is reduced in the circular layout. Finally, Gultekin et al. (2009) show that pure cycles dominate classical flow shop cycles (thus further publications focus on this type of cycle). They also formulate the problem of finding the best pure cycle in an m machine cell as a TSP where some distances are decision variables. Gultekin et al. (2017) extend the study to dual-

gripper cells: they define a framework to study pure cycles for cells with dual gripper robots; for the 2-machine case, they show that only 5 of them dominate the others. Jolai et al. (2012) and Foumani and Jenab (2013) look into the case of a robot with swap ability. Jolai et al. (2012) shows that in this case too, pure cycles dominate over classical robot move cycles, while Foumani and Jenab (2013) introduce "improved pure cycles", where some machines are left idle, showing that in some instances they perform better than standard pure cycles.

Most studies focus on identical part production. Multiple part type production is addressed by Batur et al. (2012) in 2-machine cells. Like Gultekin et al. (2009) they model the problem as a generalized form of TSP, and propose a heuristic.

The problem of minimizing jointly the cycle time and the manufacturing costs is studied by Gultekin et al. (2008, 2010) and Yildiz et al. (2012). The flexibility of the machines allow to vary their parameters (such as their speed) to adjust the processing times, with associated costs: thus processing times may be considered as decision variables. Gultekin et al. (2008) study this problem in 2 and 3-machine linear cells, determining sets of non-dominated cycles. Gultekin et al. (2010) provides a Mixed Integer Non-Linear Model (MINLP) for the 2-machine case. Yildiz et al. (2012) consider a circular layout with regular additive time. The authors also assume that the total processing time is machine-independent. For an arbitrary number of machines, they give a lower bound on the cycle time of pure cycles, and study the dominance regions and performance limitations of 2 specific pure cycles. The 3-machine case is completely solved. They perform a similar analysis with controllable processing times and similar results: in this case too the 3-machine case is completely solved.

1.5 Scope of this work and notations

As exposed in Section 1.3.3, the classical robotic flowshop problems with additive time in a linear layout (Figure 1.1a) and in a circular layout (Figure 1.1b), though very similar in formulation, appear very different in structure and complexity. For the production of identical parts, with unbounded waiting times, finding the best 1-cycle of production has long been known[4] to be of polynomial complexity in linear cells (Crama and van de Klundert, 1997a), while the exact same problem in circular cell (Rajapakshe et al., 2011) was recently proven NP-hard. However, the existing proof (a reduction from 2-Partition) is rather technical, so that it does not make clear the intuition of the difference between the two models. The 1-cycle conjecture, closed for linear layouts (true for $m \leq 3$ and false for $m > 4$) is still open for the circular layout. Though dual-gripper cells have been historically

[4]To anyone who suddenly felt not so young anymore reading this sentence, my apologies.

studied in circular layouts, few work have considered this layout in the classical, single gripper setup. The authors who consider this case, appear to make a simplifying assumption on the cycle time of a crucial robot move cycle. Aside for the theoretical matter, this makes the comparative evaluation of different models in order to decide on the more appropriate structure and equipment for a cell more complicated.

This work aims to be a step towards a better understanding of cyclic optimization in robotic cells with circular layouts. For this purpose, we mostly focus on the special *balanced* case (where all processing times are equal) which is well understood in linear layouts (Brauner, 1999, 2008), but largely open for $m \geq 3$ in circular layouts. In particular, the complexity of finding the best 1-cycle is still open in circular regular balanced cells, as well as the 1-cycle conjecture (which is proven true for $m \leq 15$ for the linear balanced configuration)

In the following, we consider cyclic production of *identical parts* in a circular robotic cell (as in Figure 1.1b), with a single gripper, no machine buffer, unbounded waiting times, and regular additive travel times. We use the following notations:

- m the number of machines,
- $M_1 \dots M_m$ the m machines,
- M_0 the input buffer (IN), and M_{m+1} the output buffer (OUT),
- δ the travel time between any two consecutive machines M_i and M_{i+1} (both ways) with $i \in \{0, \dots, m\}$. As the input and output buffers are in the same place, the travel time between M_0 and M_{m+1} is 0.
- p_i for $i \in \{1, \dots, m\}$ the processing time on machine M_i (in the non balanced case), and p the processing time on any machine in the balanced case.

Consistently with Lehoux-Lebacque (2007), and to simplify calculations, we assume the loading and unloading times to be zero. Thus, in the balanced case, an instance of the problem is specified by (m, δ, p), and in the non-balanced case by $(m, \delta, p_1, \dots, p_m)$.

Our objective is the minimization of the long run average cycle length. Cycles are specified by their sequence of activities, denoted A_i with $i \in \{0, \dots, m\}$. We use the following notation for cycles:

- π (for permutation) is used to denote a 1-cycle,
- c is used to denote a generic cycle (a k-cycle where k is not necessarily 1),
- For any cycle c, T_c is the (long run average) cycle time of c. When useful, the cycle time may be expressed as a function of p and denoted $T_c(p)$. If c

is a k-cycle, its cycle length is given by $\frac{T_c}{k}$ (obviously, the cycle length of a 1-cycle π is simply T_π).

Other useful notations will be introduced as need be; they are recapitulated page 125.

As the cell is circular, with regular additive travel times, the travel time between any two machines M_i and M_j is given by

$$\delta_{i,j} = \min(|i-j|, m+1-|i-j|)\delta \tag{1.1}$$

Chapter 2 presents tools that will be used throughout this work to analyze production cycles in circular layouts; its reading is recommended before undertaking Chapter 3 and Chapter 4. Chapter 3 adresses the best 1-cycle problem and the 1-cycle conjecture in small cells. By deriving properties of interesting production cycles, we settle the best 1-cycle problem for $m \leq 8$ and provide a counter-example to the 1-cycle conjecture for $m = 6$. In Chapter 4, we consider the best 1-cycle problem in large cells (any $m > 8$). We identify, define and study a new interesting family of cycles, which lead us to conjecture its dominance over 1-cycles (and thus the polynomiality of the best 1-cycle problem in circular regular balanced cells), proven for $m \leq 11$. Chapter 5 is an opening toward a more practical problem, also combining scheduling and transportation.[5]

[5] Although not exactly with the same exact meaning.

Chapter 2

Tools for analysis of production cycles in circular layouts

> Notre tête est ronde pour permettre à la pensée de changer de direction.
>
> Francis Picabia

This chapter presents tools that are used throughout this document to bound, visualize and compute cycle times. Section 2.1 expose classical lower bounds for the cycle time. Section 2.2 presents relevant cycles together with practical graphical representations. One of this cycle is specific to circular layouts and so far its cycle time was not fully understood: in Section 2.3, we give and prove a formulation of its cycle time. Finally, Section 2.4 presents a method and tool to numerically compute cycle times.

2.1 Lower bounds

In this section, we first present two classical lower bounds on the cycle time of any k-cycle, valid both for linear and circular layout. For each one, we give the formulation adapted to the regular case, and the formulation adapted to the regular balanced case.

2.1.1 Classical bounds

The following bound (LB_M) considers the cell from the machine's point of view. It conveys the minimum time between two loadings of the same machine.

Proposition 2.1. $\mathbf{LB_M}$ *(Crama and van de Klundert, 1997a)*
In a regular cell, any k-cycle c satisfies

$$T_c \geq k(\max_i p_i + 4\delta) \tag{2.1}$$

In the balanced case, this simplifies to

$$T_c \geq k(p + 4\delta) \tag{2.2}$$

The following bound (LB_R) considers the cell from the robot's point of view. It counts the minimum travel time of the robots refined by the respective position of consecutive activities.

Proposition 2.2. $\mathbf{LB_R}$ *(Dawande et al., 2002)*
In a regular cell, any k-cycle c satisfies

$$T_c \geq k((m+1)\delta + \sum_{i\in\{1,\ldots,m\}} \min(p_i, \delta)) \tag{2.3}$$

In the balanced case, this simplifies to

$$T_c \geq k((m+1)\delta + m\min(p, \delta)) \tag{2.4}$$

Each one of the $(m+1)$ activities must be performed exactly k times: the loaded robot moves add up to a travel time of $k(m+1)\delta$. Additionally, any activity A_i is either immediately followed by the subsequent activity A_{i+1}, in which case the robot must wait p time units at machine M_{i+1} for the part to be processed, or by any other activity, in which case the robot performs an empty move immediately after A_i, adding at least δ to the travel time.

2.1.2 Minimum waiting time

We now introduce the following notations, for every cycle c:

- Δ_c is the total travel time of the robot,
- $d_{i,l}(c)$ is the travel time of the robot between the l-th loading of machine M_i and its subsequent unloading (in a cyclic sense). For a 1-cycle, as each machine is loaded (and unloaded) exactly one time, the second index is unnecessary and we simply use the notation $d_i(\pi)$.
- $d_{min}(c) = \min_{i,l}(d_{i,l}(c))$

For example, consider the cycle $c = A_0A_3A_2A_4A_1$, represented later on Figure 2.1. One can verify that $\Delta_c = 13\delta$, $d_1 = d_3 = 9\delta$, $d_2 = 8\delta$ $d_4 = 4\delta$, thus $d_{min}(c) = 4\delta$.

The following lower bound is actually a lower bound on the cycle time of a given cycle, depending on the values Δ and d_{min}.

Proposition 2.3. *For any cycle c, the cycle time verifies*

$$T(c) \geq \Delta(c) + \max(0, p - d_{min}(c)) \tag{2.5}$$

Proof. It is easy to see that $T(c) \geq \Delta(c)$.

Let $(i_0, l_0) = \arg\min_{i,l}(d_{i,l}(c))$. Between the l_0-th loading of M_{i_0} and its subsequent unloading, the robot travels d_{min}, but there must be at least p units of time for the part to be processed. So, if $p \geq d_{min}$, somewhere between the loading and the unloading, the robot must wait (additionally to its travel time) at least $p - d_{min}$ units of time. Hence $T(c) \geq \Delta(c) + \max(0, p - d_{min}(c))$ □

2.2 Classical Cycles and Representation

In this section we introduce three relevant cycles in circular cells (Section 2.2.2). To describe them, we rely on the two types of graphical representations of cycles that are exposed in Section 2.2.1.

2.2.1 Graphical representation

The first representation is classical and describes the movement of the robot within the cell during the unfolding of a cycle. The second one is new and specific to the balanced case: it allows to compare cycle times of different cycles while varying the cell's parameters.

Representation of the robot moves

The unfolding of a cycle in space and time can be graphically represented on a chronogram. The vertical axis represents the cell, each graduation standing for a machine, from the input buffer to the output buffer. Note that on a circular cell where the output and input buffer occupy the same location, the first and last graduation are virtually equivalent. The horizontal axis represents time.

On this graph, loaded robot moves are represented by a solid line and empty robot moves are represented by a dashed line. When needed, we also represent parts being processed by a red solid line. Figure 2.1 presents an example. Without

processing times (or with $p = 0$), this representation can also be used to visualize the general structure of the cycle.

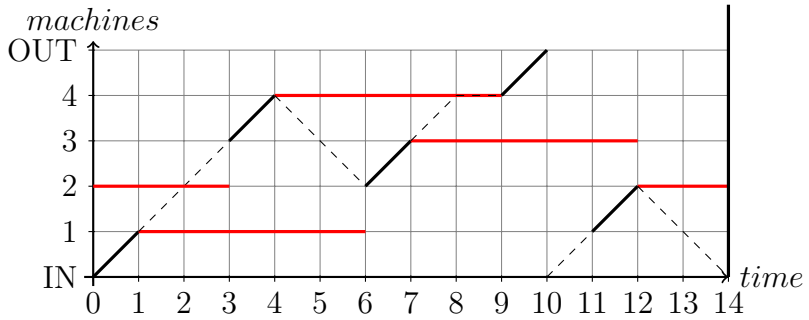


Figure 2.1: One iteration of cycle $(A_0A_3A_2A_4A_1)$ on instance $(m = 4, p = 5, \delta = 1)$.

Representation of the cycle time in the balanced case

In the balanced case, a cell is characterized by three parameters, the number of machines m, the processing p and the travel time δ. Without loss of generality, we can consider that δ is one time unit (by dividing p and δ by δ). For a given m, and by setting δ to 1, we can look at cycle times and lower bounds as functions of p. Seen that way, the cycle time of any cycle is an increasing piecewise linear function.

In order to highlight the performances and optimality regions of cycles, the cycle time functions can be represented on a graph where the horizontal axis represents p and the vertical axis represents time. For example, Figure 2.2 shows lower bounds LB_M (Equation (2.2)) and LB_R (Equation (2.4)) on such a graph.

2.2.2 Some classical cycles

In this section, we present three classical 1-cycles. The first two, the *identity cycle* and *downhill cycle* were formalized and studied by Sethi et al. (1992). We give their cycle time formulation adapted to circular regular cells. The third one, the *odd-even cycle* is only of interest in circular cells. It has for example been considered by Geismar et al. (2005a), Rajapakshe et al. (2011), who uses in their papers a lower bound as its cycle time: they only consider one iteration of the cycle, while deriving the precise cycle time requires to consider several consecutive iterations. This makes the cycle time derivation more complex than for the two former cycles: we give a formulation and proof in Section 2.3.

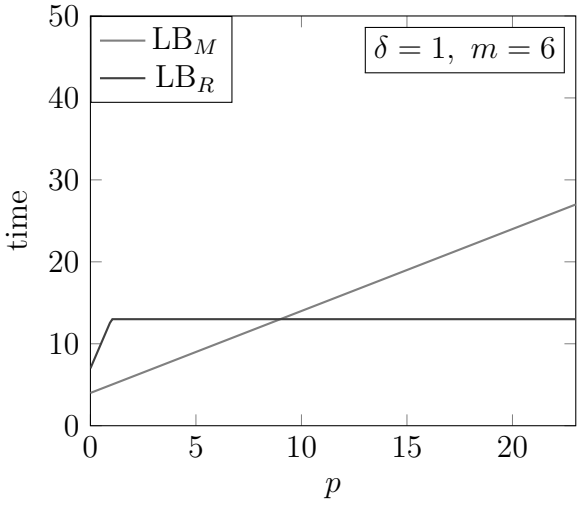


Figure 2.2: For $m = 6$, $\delta = 1$, lower bounds LB_M (Equation (2.2)) and LB_R (Equation (2.4)) on the cycle time of 1-cycles, as functions of p.

For each of these three cycles, we give an execution example on instance I : $(m = 4, p = 3, \delta = 1)$ (see respectively Figures 2.3, 2.4 and 2.6).

Identity cycle

We call *identity cycle* (also named *uphill permutation* or *forward cycle* in the literature) the cycle $\pi_{id} = A_0 A_1 ... A_m$, represented on Figure 2.3. In this cycle, the robot circles the cell once, doesn't perform any empty move and waits at each machine during the full length of the processing. We have $\Delta_{\pi_{id}} = (m+1)\delta$ and $d_i(\pi_{id}) = 0$ for all $i \in \{1, \ldots, m\}$, thus $d_{min}(\pi_{id}) = 0$. The corresponding cycle time, respectively in the non-balanced and balanced case, is:

$$T_{\pi_{id}} = (m+1)\delta + \sum_{i \in \{1,\ldots,m\}} p_i \tag{2.6}$$

$$T_{\pi_{id}} = (m+1)\delta + mp \tag{2.7}$$

Intuitively, this cycle is interesting for instances for which p is much smaller than δ. From the lower bound LB_R (Proposition 2.2), one can derive the following classical result:

Proposition 2.4. *In a regular balanced cell, if $p \leq \delta$, then the identity cycle π_{id} is optimal.*

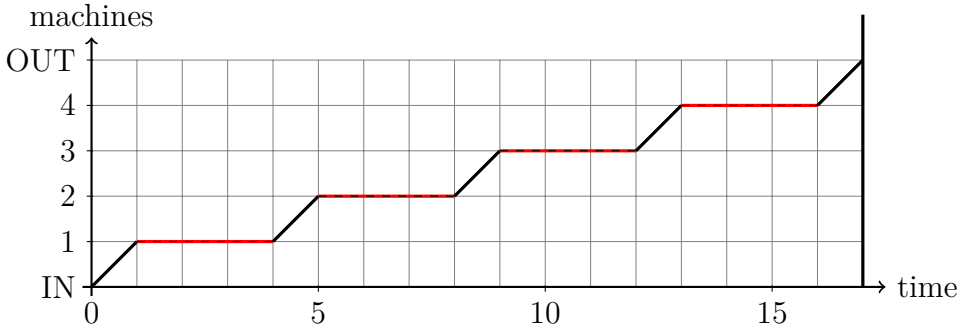


Figure 2.3: One iteration of π_{id} on instance $I = (m = 4, p = 3, \delta = 1)$

Downhill cycle

We call *downhill* cycle (also named *reverse cycle*) the cycle $\pi_{dh} = A_0 A_m A_{m-1} ... A_1$, represented on Figure 2.4. We have $\Delta_{\pi_{dh}} = 3(m+1)\delta$ and $d_i(\pi_{dh}) = (3m-1)\delta$ for all $i \in \{1, \ldots, m\}$, thus $d_{min}(\pi_{dh}) = (3m-1)\delta$. Its cycle time, respectively in the non-balanced and balanced case, is:

$$T_{\pi_{dh}} = 3(m+1)\delta + \max_{i \in \{1,\ldots,m\}} (0, p_i - (3m-1)\delta) \tag{2.8}$$

$$T_{\pi_{dh}} = 3(m+1)\delta + \max(0, p - (3m-1)\delta) \tag{2.9}$$

In this cycle, each spot is visited by the robot three times. Intuitively, this cycle is interesting for instances for which p is much larger than δ. From the lower bound LB_M (Proposition 2.1), one can derive the following classical result:

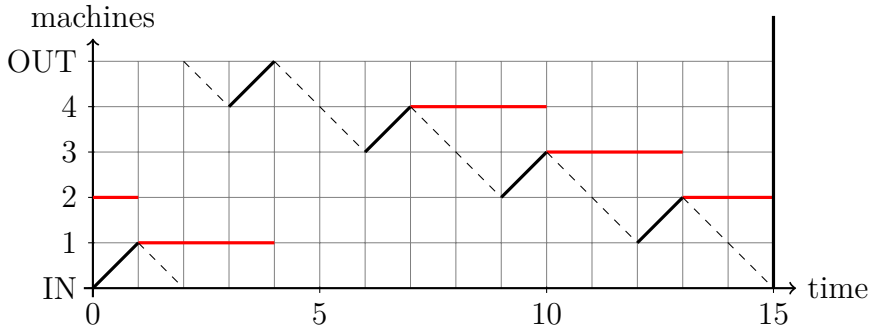


Figure 2.4: One iteration of π_{dh} on instance $I = (m = 4, p = 3, \delta = 1)$

Proposition 2.5. *If $p \geq (3m-1)\delta$, then the downhill cycle π_{id} is optimal.*

Figure 2.5 shows the cycle times of π_{id} and π_{dh} as function of p, as well as lower bounds from LB_M and LB_R, highlighting their regions of optimality.

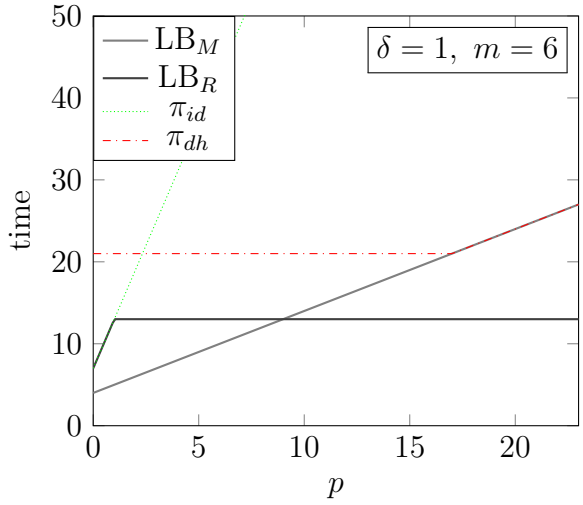


Figure 2.5: For $m = 6$, $\delta = 1$, lower bounds LB_M and LB_R, (respectively Equations 2.2 and 2.4), and cycle times of π_{id} and π_{dh} as functions of p.

Odd-Even cycle

In circular cells, a third 1-cycle of particular interest is the odd-even cycle, defined as such:
For m even,

$$\pi_{oe} = A_0 A_2 A_4 \dots A_m A_1 A_3 A_5 \dots A_{m-1}$$

And for m odd,

$$\pi_{oe} = A_0 A_2 A_4 \dots A_{m-1} A_1 A_3 A_5 \dots A_m.$$

In this cycle, the robot circles around the cell twice: the first time performing even activities (thus loading odd machines), the second time performing odd activities (thus loading even machines). Is overall travel time is $\Delta_{\pi_{oe}} = 2(m+1)\delta$ and we have $d_i(\pi_{oe}) = (m+1)\delta$ for all $i \in \{1, \dots, m\}$, thus $d_{min}(\pi_{oe}) = (m+1)\delta$.

Note that contrary to π_{id} and π_{dh}, this cycle is not a pyramidal permutation as defined in (Crama and van de Klundert, 1997a), therefore it is dominated in linear cells.

2.3 Cycle time of the odd-even Cycle

In order to understand the cycle time of π_{oe}, it is necessary to consider several iterations. In this section, we give and prove the formulation of its cycle time in the balanced case as well as lower and upper bounds for the non-balanced case.

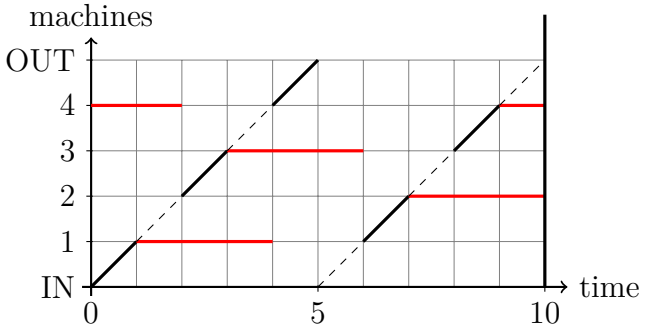


Figure 2.6: One iteration of π_{oe} on instance $I = (m = 4, p = 3, \delta = 1)$

2.3.1 Balanced case

Proposition 2.6. *The cycle time for the odd-even cycle is*

$$T_{\pi_{oe}} = 2(m+1)\delta + \frac{2\alpha - 1}{\alpha} \max(0, p - (m+1)\delta) \qquad \text{with } m = \begin{cases} 2\alpha & \text{if } m \text{ even} \\ 2\alpha - 1 & \text{if } m \text{ odd} \end{cases}$$

Proof. Consider α consecutive iterations of π_{oe}. The robot travels $2(m+1)\alpha\delta$, circling the cell 2α times. Let us follow the path of one same part in the cell. Figure 2.7 presents an example in a 4-machine cell.

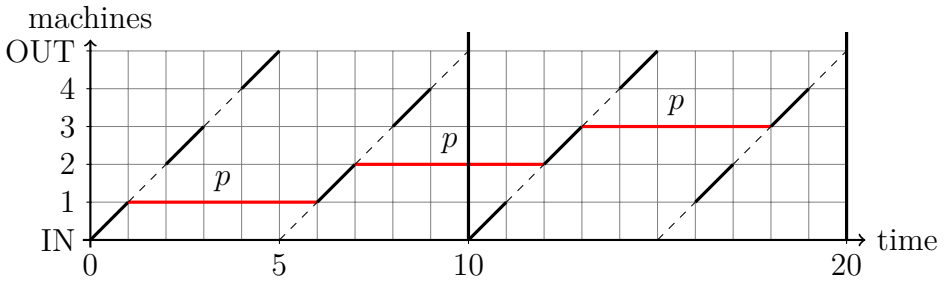


Figure 2.7: Two consecutive iterations of cycle π_{oe} in a 4-machine cell.

Each iteration is constituted by two loops around the cell. During the first loop of the first iteration, machine M_1 is loaded with A_0. It is then unloaded during the second loop with A_1. Between the loading and the unloading of M_1, the robot travels $(m+1)\delta$ and the part must remain on M_1 at least p time units, so the robot must wait $\max(0, p - (m+1)\delta)$. Similarly, during any loop, the part is loaded on a machine, then unloaded and taken to the next machine during the next loop, and the robot must wait $\max(0, p - (m+1)\delta)$. If m is odd, the part exits the cell on the last loop. If m is even, it is loaded on M_m during the last loop.

Eventually, for this one part, the robot must wait at least $(2\alpha - 1)\max(0, p - (m+1)\delta)$ over α iterations of the cycle. Therefore,

$$T_{\pi_{oe}} \geq 2(m+1)\delta + \frac{2\alpha - 1}{\alpha}\max(0, p - (m+1)\delta) \tag{2.10}$$

Now, let us call w_i^j the waiting time of the robot at machine M_j during the i-th iteration of cycle π_{oe}, and $W^i = (w_1^i, \ldots, w_m^i)$. Let $a = p - (m+1)\delta$.

Case 1: m odd $\pi_{oe} = (A_0 A_2 A_4 \ldots A_{m-1} A_1 A_3 A_5 \ldots A_m)$ The waiting times at iteration i are:

$$\begin{cases} w_{2j}^i &= \max(0, a - \sum_{k=1+j}^{\alpha} w_{2k-1}^{i-1} - \sum_{k=1}^{j-1} w_{2k}^i) \\ w_{2j-1}^i &= \max(0, a - \sum_{k=j}^{\alpha-1} w_{2k}^i - \sum_{k=1}^{j-1} w_{2k-1}^i) \end{cases}$$

If $a \leq 0$ then all waiting times are zero, and the cycle time is $2(m+1)\delta$.

If not, we can easily check that the vector $W_0 = (\frac{a}{\alpha}, \ldots \frac{a}{\alpha})$ is a fixed point of W. The corresponding cycle time is $2(m+1)\delta + \frac{2\alpha-1}{\alpha}a$.

Case 2: m even $\pi_{oe} = (A_0 A_2 A_4 \ldots A_m A_1 A_3 A_5 \ldots A_{m-1})$

Similarly, the waiting times at iteration i are

$$\begin{cases} w_{2j}^i &= \max(0, a - \sum_{k=1+j}^{\alpha} w_{2k-1}^{i-1} - \sum_{k=1}^{j-1} w_{2k}^i) \\ w_{2j-1}^i &= \max(0, a - \sum_{k=j}^{\alpha} w_{2k}^i - \sum_{k=1}^{j-1} w_{2k-1}^i) \end{cases}$$

If $a \leq 0$ then $W = 0$, and the cycle time is $2(m+1)\delta$.

If not, we can check that the vector $W_0 = (0, \frac{a}{\alpha}, \ldots \frac{a}{\alpha})$ is a fixed point of W. The corresponding cycle time is $2(m+1)\delta + \frac{2\alpha-1}{\alpha}a$

In both cases the lower bound (2.10) is tight, so the cycle time is:

$$T_{\pi_{oe}} = 2(m+1)\delta + \frac{2\alpha - 1}{\alpha}\max(0, p - (m+1)\delta) \tag{2.11}$$

□

Figure 2.8 completes Figure 2.5 with the cycle time of π_{oe}.

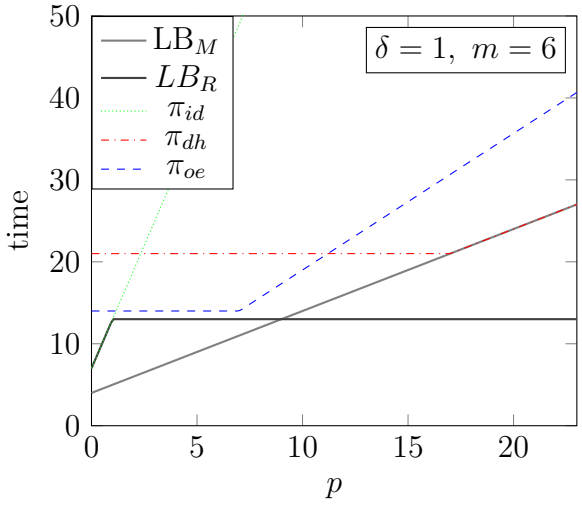


Figure 2.8: For $m = 6$, $\delta = 1$, lower bounds LB_M and LB_R, (respectively Equations 2.2 and 2.4), and cycle times of π_{id}, π_{dh}, and π_{oe} as functions of p.

2.3.2 Non balanced case

In the non-balanced case, one can derive in a similar way a lower and an upper bound for π_{oe}'s cycle time, depending on $p_{min} = \min_{i\in\{1,\dots,m\}} p_i$ and $p_{max} = \max_{i\in\{1,\dots,m\}} p_i$

Proposition 2.7.

(i) $T_{\pi_{oe}} \geq 2(m+1)\delta + \max\left(0, \frac{2\alpha-1}{\alpha}(p_{min} - (m+1)\delta)\right)$

(ii) $T_{\pi_{oe}} \leq 2(m+1)\delta + \max\left(0, \frac{2\alpha-1}{\alpha}(p_{max} - (m+1)\delta)\right)$

Proof.

(i) The proof is similar to the balanced case, considering that the necessary waiting time between the loading and unloading of a part on any machine is at least $p_{min} - (m+1)\delta$.

(ii) The waiting time needed between the loading and unloading of a part on any machine is at most $p_{max} - (m+1)\delta)$, which is ensured by the waiting times vector $W_{max} = \left(\frac{p_{max}-(m+1)\delta}{\alpha}, \dots \frac{p_{max}-(m+1)\delta}{\alpha}\right)$ for m odd or $W_{max} = \left(0, \frac{p_{max}-(m+1)\delta}{\alpha}, \dots \frac{p_{max}-(m+1)\delta}{\alpha}\right)$ for m even. □

2.4 Cycle time computation

(Brauner, 1999) proved that for any cycle, independently of the initial state, the cell reaches a periodic steady state, after a finite number of cycle iterations. This

means that there exists a number of iterations (a period) that leaves the cell in exactly the same state, as defined above. Note that if the minimum period is strictly greater than one, cycle time may differ between consecutive iterations. Therefore, to compute the long run average cycle time, it is necessary to consider a whole period.

For example, Figure 2.9 shows 4 consecutive iterations of π_{oe} on instance $J : (m = 4,\ p = 6,\ \delta = 1)$ once reached a periodic steady state. Iterations 3 and 4 are identical to iterations 1 and 2, which shows the execution is 2-periodic. Looking at the first two iterations, we can see that they unfold differently: during the first one, the robot waits one time unit at machine M_2 and M_3, while during the second one, waiting is only needed at M_4. Consequently, the first iteration lasts 12 time units, while the second one lasts 11 time units. Looking at the duration of a single iteration leads to an over- or under-estimation of the long run cycle time: to get an accurate measure, it is necessary to consider the average execution time over 1 period, here 2 iterations, which gives the correct cycle time of 11.5.[1]

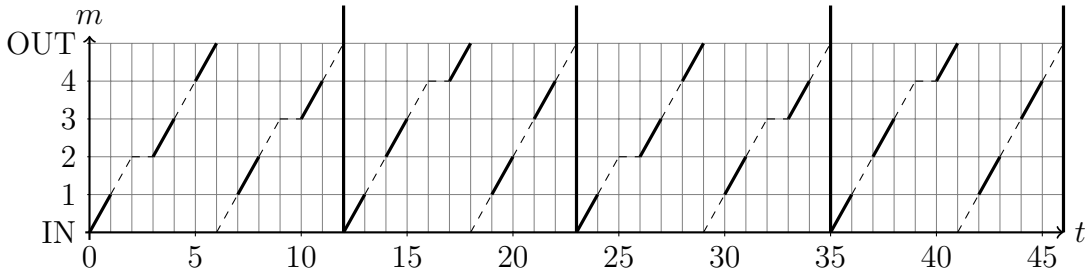


Figure 2.9: Two periods (four iterations) of π_{oe} operating in periodic steady state on instance $J : (m = 4, p = 6, \delta = 1)$

We developed a tool in Java allowing to simulate the behavior of the cell and therefore experimentally compute cycle times. The input consists in the size of the cell, travel times and processing times, either integer or fractional.

The state of the cell is represented by the position of the robot, and a vector of fractional values representing the state of the machines. If the value for a machine is non-negative, it represents the remaining processing time on this machine. If not, it means the machine is free. Given a cycle, a feasible initial state is automatically computed by loading the machines which need to be loaded with a remaining processing time of 0. To compute the cycle time, the program first performs several

[1] *Mutatis mutandis*, this example also works for linear cells (consider $p = 9$, for example). However, note that π_{oe} is not a pyramidal permutation. Brauner (1999) conjectures that pyramidal permutations (which dominate 1-cycles in linear cells, but not in circular cells) are *stable*, meaning that no matter the initial state, the steady state of these cycles has a period of 1. The author proves this property for linear cells with 2, 3 and 4 machines.

initialization iterations to ensure the cell is running in a periodic steady state (the number of initial iterations can be set by the user). The period is then calculated by monitoring the cell's state after each subsequent iteration and comparing it to the state it was in immediately after the initialization. The programs then outputs the average value of the cycle time over one period.

2.5 Conclusion

In this chapter, we presented tools that serves as a basis for studying cyclic production in circular cells, including lower bounds on the cycle time, graphical representations and three particular cycles.

We showed that for cycle which exploits the circular structure of the cell by circling it twice, as the odd-even cycle π_{oe}, derivation of the cycle time formulation is not trivial as the cycle has to be considered over several iterations. This cycle structure is especially important in Chapter 4, where we study families of cycles formed by altering the odd-even cycle.

Chapter 3

Properties of 1-cycles and small cells analysis

The objective of this chapter is to settle properties for 1-cycles in order to find interesting production cycles. We first study the region of optimality within 1-cycles of the three classical 1-cycles introduced in Chapter 2 (Section 3.1). Then, we establish necessary properties for the best 1-cycles (Section 3.2), which allow to conclude for small cells ($m \leq 8$) in Section 3.3. The best 1-cycle problem for larger cells will be studied in Chapter 4. As the 2-machine case is solved (Sethi et al., 1992), we consider $m \geq 3$.

In regular additive linear balanced cells, the 1-cycle conjecture is valid for $m < 16$ (Brauner, 2008). In regular additive *circular* balanced cell, for $m = 6$ we exhibit a 2-cycle that is strictly better than any 1-cycle for a given instance, thus disproving the 1-cycle conjecture for circular balanced cells (Section 3.4).

To make the proofs more readable, we use this notation for activities:

$$A_i = A_{i \bmod (m+1)}$$

3.1 Regions of optimality for the classical cycles

In this section, within the set of 1-cycle, we seek for the best ones. Here "optimality" is in the sense of optimality within 1-cycles. We consider the three classical cycles $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ and describe characteristics of instances for which they are optimal. We use the lower bounds on the cycle time LB_M and LB_R described in Chapter 2 (page 23), to which we add a new lower bound (LB_D) specific to 1-cycles, by considering the direction of the robot:

Proposition 3.1. ($\mathbf{LB_D}$) *If $p \geq \delta$, and π is a 1-cycle with $\pi \neq \pi_{id}$, then*

$$T_\pi(p) \geq 2(m+1)\delta \tag{3.1}$$

Proof. Let π be a 1-cycle different from the identity cycle π_{id}.

Case 1: If π contains a sequence A_iA_j with $j \neq i+2$ and $j \neq i+1$, then, following a reasoning similar to the one behind Proposition 2.2 page 24, we have

$$T_\pi(p) \geq (m+1)\delta + (m-1)\min(p,\delta) + 2\delta \geq 2(m+1)\delta$$

It takes the robot $(m+1)\delta$ time units to perform all activities, then at least $\min(p,\delta)$ between any pair of consecutive activities except for A_iA_j, and at least 2δ to travel from M_{i+1} to M_j while performing A_iA_j.

Case 2: If not, then π contains at least a sequence A_iA_{i+2} (as $\pi \neq \pi_{id}$), and every other 2-element sub-sequence in π can be written A_jA_{j+1} or A_jA_{j+2}. Then, we know that:

- The robot always travels in the same direction, forward: with $m \geq 3$, the shortest circular path from a machine M_j to M_{j+1} never requires to go backward.
- The robot travels at least twice between M_{i+1} and M_{i+2}. Once loaded, while performing activity A_{i+1} and once empty, while performing the sequence A_iA_{i+2}.

Hence the robot circles the cell at least twice:

$$T_\pi(p) \geq 2(m+1)\delta$$

□

The following proposition specifies the regions of optimality of $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$.

Proposition 3.2.

(i) if $p \leq \frac{m+1}{m}\delta$, then the identity permutation π_{id} dominates 1-cycles.

(ii) if $\frac{(m+1)}{m}\delta \leq p \leq (m+1)\delta$, then the odd-even cycle π_{oe} dominates 1-cycles.

(iii) if $p \geq (3m-1)\delta$, the downhill permutation π_{dh} dominates 1-cycles (and even all k-cycles).

Proof. Recall that the the cycle times of π_{id}, π_{oe} and π_{dh} are given by

$$T_{\pi_{id}}(p) = (m+1)\delta + mp$$
$$T_{\pi_{oe}}(p) = 2(m+1)\delta + \frac{2\alpha-1}{\alpha}\max(0, p-(m+1)\delta) \quad \text{where } \alpha = \left\lfloor \frac{m+1}{2} \right\rfloor$$
$$T_{\pi_{dh}}(p) = 3(m+1)\delta + \max(0, p-(3m-1)\delta)$$

Claim *(iii)* is a classical result that follows directly from Proposition 2.1. For $p \geq (3m-1)\delta$, $T_{\pi_{dh}}(p) = p + 4\delta$, hence by Proposition 2.1 (LB_M), π_{dh} dominates all k-cycles thus proving *(iii)*.

For $p \leq \delta$, then from Proposition 2.2 (LB_R), for any k-cycle c, one has $T_c(p) \geq k(m+1)\delta + mp = kT_{\pi_{id}}(p)$, hence π_{id} is optimal.

For $\delta \leq p \leq \frac{m+1}{m}\delta$, we have

$$T_{\pi_{id}}(p) = (m+1)\delta + mp \leq 2(m+1)\delta$$

and, from Proposition 3.1 (LB_D), for any 1-cycle $\pi \neq \pi_{id}$:

$$T_\pi(p) \geq 2(m+1)\delta \geq T_{\pi_{id}}(p) \tag{3.3}$$

which implies *(i)*.

Finally, for $\frac{(m+1)}{m}\delta \leq p \leq (m+1)\delta$, we have

$$\begin{aligned} T_{\pi_{oe}}(p) &= 2(m+1)\delta \\ T_{\pi_{id}}(p) &= (m+1)\delta + mp \geq 2(m+1)\delta \end{aligned}$$

and for any 1-cycle $\pi \neq \pi_{id}$, from Proposition 3.1 (LB_D):

$$T_\pi(p) \geq 2(m+1)\delta \geq T_{\pi_{oe}}(p)$$

which implies *(ii)*. □

Proposition 3.2 settles the case of $p \leq (m+1)\delta$ and $p \geq (3m-1)\delta$, as summarized by Figure 3.1. In this graphic, we can see that in their regions of optimality the corresponding cycles fit with a lower bound. Now remain the instances with $(m+1)\delta < p < (3m-1)\delta$. This is the issue addressed in the next section.

3.2 Necessary properties of optimal 1-cycles

In this section, we assume that $(m+1)\delta < p < (3m-1)\delta$ since other cases are settled in Proposition 3.2. We establish some properties (Propositions 3.3 to 3.6) that a 1-cycle must satisfy in order to do strictly better than both the odd-even cycle and the downhill permutation. We will use these properties later on in Section 3.3 to determine the best 1-cycles for $m \leq 8$.

Let π^* be such a 1-cycle, thus verifying for some $(m+1)\delta \leq p \leq (3m-1)\delta$,

$$\begin{cases} T_{\pi^*}(p) & < & T_{\pi_{oe}}(p) \\ T_{\pi^*}(p) & < & T_{\pi_{dh}}(p) = 3(m+1)\delta \end{cases} \tag{3.4}$$

Unless otherwise specified, the following properties are valid for $m \geq 3$. Propositions 3.3 to 3.4 are valid for $m \geq 5$.

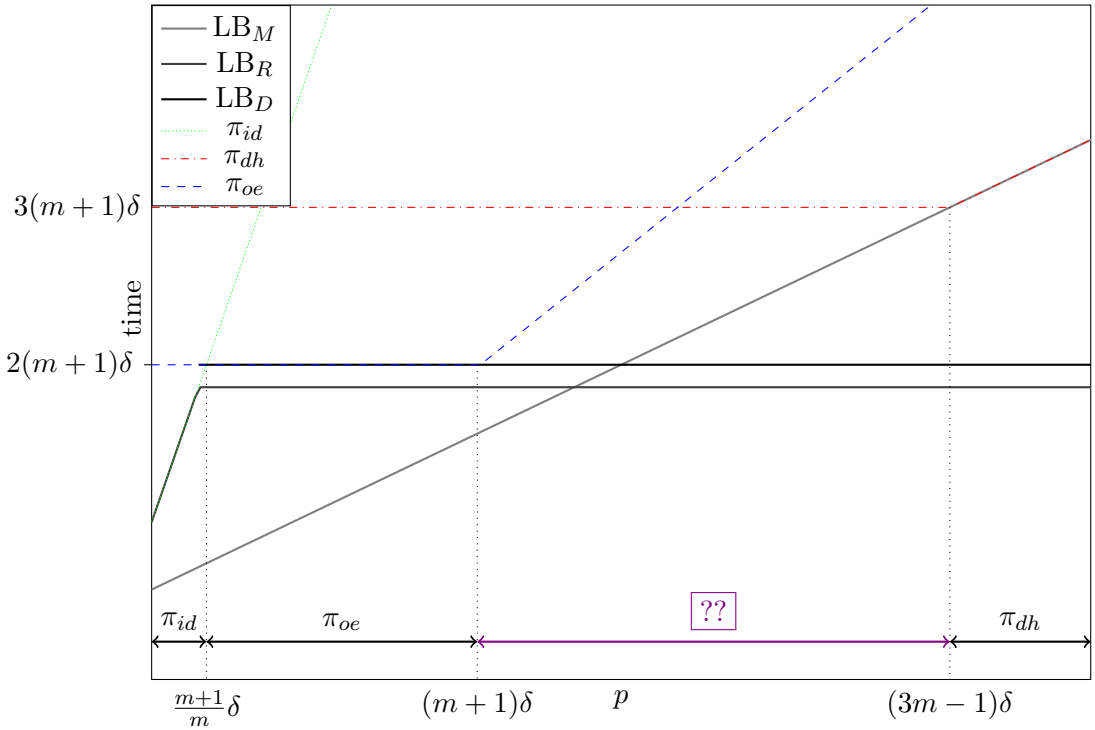


Figure 3.1: Lower bounds and cycle times of π_{id}, π_{oe} and π_{dh}, showing their regions of optimality

Proposition 3.3.
If $m \geq 5$, the cycle π^* contains no sequence of the form A_iA_{i+1} with $i \neq m$.

Proof. First, π^* contains at most one sub-sequence of two consecutive activities (except from the sequence A_mA_0). Otherwise, as the robot waits p time units at each machine M_i such that π^* contains A_iA_{i+1} and travels at least $(m+1)\delta$ (performing all activities), we have $T_{\pi^*}(p) \geq (m+1)\delta + 2p \geq 3(m+1)\delta = T_{\pi_{dh}}(p)$, which contradicts Equation (3.4).

Let us assume now that there exists exactly one $i \neq m$ so that A_iA_{i+1} is a sub-sequence of π^*. The robot waits $(m+1)\delta$ at machine M_{i+1}. It travels at least $(m+1)\delta$ while loaded (performing the $m+1$ activities), and $(m-1)\delta$ while empty (after performing each one of the activities $A_j, j \notin i, m$, the robot travels at least δ to get to the next machine to unload). Thus, we already have

$$T^*_\pi(p) \geq 2(m+1)\delta + (m-1)\delta \tag{3.5}$$

We show that the robot travels at least an additional 2δ, leading to $T^*_\pi(p) \geq 3(m+1)\delta$.

Case 1: π^* is of the form $A_0A_1 \dots A_m$.

In this case, for $m \geq 5$, either π^* contains the subsequence $A_{m-1}A_2$, or there exist indices $j, k \notin \{m, 0, 1\}$ so that $A_{m-1}A_j$ and A_kA_2 are two distinct sub-sequences of π^*. In both cases, the robot travels at least an additional 2δ compared to Equation (3.5).

Case 2: π^* is of the form $A_0 \dots A_{m-1}A_m$.

In this case, for $m \geq 5$, either π^* contains the subsequence $A_{m-2}A_1$, or there exist indices $j, k \notin \{m-1, m, 0\}$ so that $A_{m-2}A_j$ and A_kA_1 are two distinct sub-sequences of π^*. Again, in both cases, the robot travels at least an additional 2δ compared to Equation (3.5).

Case 3: π^* is of the form $A_0 \dots A_iA_{i+1} \dots A_m$, with $1 \leq i \leq m-2$.

In this case, either π^* contains the subsequence $A_{i-1}A_{i+2}$, or there exist indices $j, k \notin \{i, i+1\}$ so that $A_{i-1}A_j$ and A_kA_{i+2} are sub-sequences of π^*. In the second case, the robot travels an additional 2δ. In the first case, π^* can be written in one of the following ways:

$$\pi^* = A_0 \dots A_iA_{i+1}SA_{i-1}A_{i+2} \dots A_m \tag{3.6}$$

$$\pi^* = A_0 \dots A_{i-1}A_{i+2}SA_iA_{i+1} \dots A_m \tag{3.7}$$

Where S is an activity sequence, including the empty sequence.

The robot already travels an additional δ while performing $A_{i-1}A_{i+2}$. In order to have no additional travel time, the sequence $A_{i+1}SA_{i-1}$ in (3.6) or $A_{i+2}SA_i$ in (3.7) must be one of the following sequences (in the cyclic sense):

$$\begin{cases} A_0A_2A_4 \dots A_{m-1} \text{ or } A_1A_3A_5 \dots A_m & \text{if } m \text{ is odd} \\ A_0A_2A_4 \dots A_mA_1A_3A_5 \dots A_{m-1} & \text{if } m \text{ is even} \end{cases}$$

which is impossible as S cannot contain A_0 or A_m. So, the robot travels at least an additionnal 2δ compared to Equation (3.5)

Case 4: π^* is of the form $\dots A_iA_{i+1} \dots$ and π^* does not contain A_mA_0.

In this case, there exist indices $j, k \notin \{i, i+1\}$ and $l \neq 0$ so that $A_{i-1}A_j$, A_kA_{i+2} and A_mA_l are subsequences of π^*, with at least 2 of these subsequences distinct. In both cases, the robot travels at least an additional 2δ.

In all cases, we have $T_{\pi^*}(p) \geq (m+1)\delta + (m+1)\delta + (m-1)\delta + 2\delta = 3(m+1)\delta$, thus any cycle containing a sequence of the form A_iA_{i+1} with $i \neq m$ is dominated by π_{dh}. □

Proposition 3.4.
If $m \geq 5$, the travel time Δ_{π^*} associated with π^* verifies $\Delta_{\pi^*} > 2(m+1)\delta$.

Proof. First, we show that $\Delta_{\pi^*} \geq 2(m+1)\delta$. This is a consequence of Proposition 3.3. As π^* contains no sequence of the form A_iA_{i+1} with $i \neq m$, there are two possibilities:

Case 1: There exists a sub-sequence A_iA_j with $j \neq i+2$ in π^*. Then the travel time of the robot is at least $(m+1)\delta + m\delta + \delta = 2(m+1)\delta$

Case 2: π^* contains only subsequences of the form A_iA_{i+2} and maybe A_mA_0. Similarly to the proof of Proposition 3.1, we can say that the robot always travels forward, and circles the cell twice: $\Delta_{\pi^*} \geq 2(m+1)\delta$.

Now, let us show that the only cycle such that $\Delta_{\pi^*} = 2(m+1)\delta$ is the odd-even cycle π_{oe}. We already know that π^* can have no sub-sequences A_iA_{i+1} except for A_mA_0.

Case 1: π^* does not contain the sequence A_mA_0. Then all sub-sequences have to be of the form A_iA_{i+2}: the only possible cycle is of the form $A_0A_2A_4 \dots A_mA_1A_3 \dots A_{m-1}$, which is π_{oe} if m is even, and not possible if m is odd.

Case 2: π^* contains the sequence A_mA_0. The cycle π^* contains only one sequence of the form A_iA_{i+3} or A_iA_{i-1}. The others are of the form A_iA_{i+2}. The cycle π^* contains a sequence $A_{m-1}A_j$ with $j \neq 0$, and a sequence A_lA_1 with $l \neq m$. Thus, π^* must contain the sequence $A_{m-1}A_1$. The only possible cycle is of the form $A_0A_2A_4 \dots A_{m-1}A_1A_3 \dots A_m$, which is π_{oe} if m is odd, and not possible if m is even.

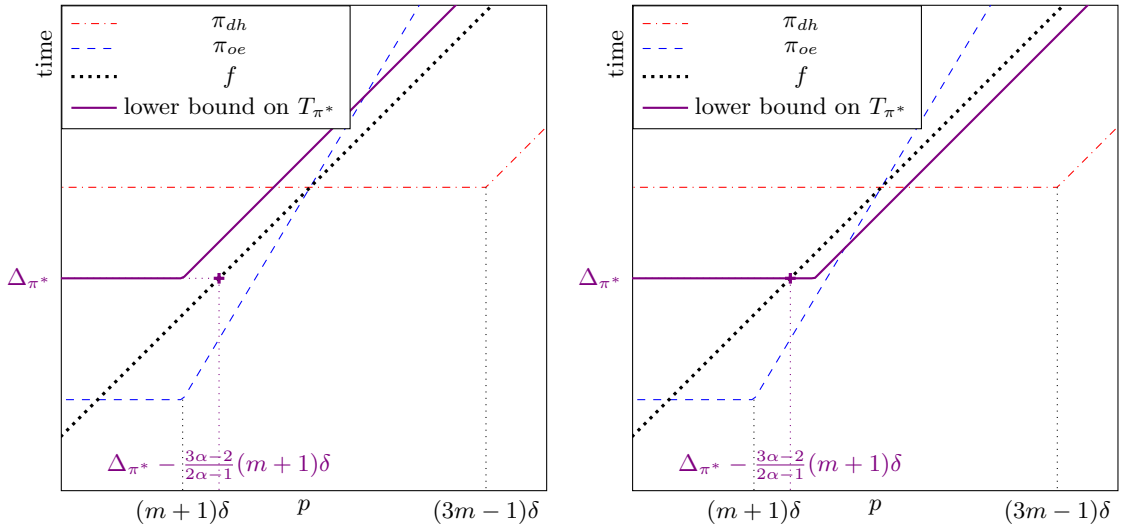
□

Proposition 3.5. *The travel time Δ_{π^*} associated with π^* verifies $\Delta_{\pi^*} < 3(m+1)\delta$*

Proof. Otherwise, we would have $T_{\pi^*}(p) \geq 3(m+1)\delta = T_{\pi_{dh}}(p)$ □

The following proposition links the total travel time of π^*, Δ_π, with the minimum travel time between the loading and unloading of a machine $d_{min}(\pi^*)$. Recall from Proposition 2.3 page 25 that the cycle time of any cycle c is at least $\Delta_c + \max(0, p - d_{min}(c))$. Note that this lower bound is a piece-wise linear function of p whose slope changes for $p = d_{min}$ (the slope is 0 for $p \leq d_{min}$ and 1 otherwise).

Proposition 3.6. $d_{min}(\pi^*) > \Delta_\pi - \frac{3\alpha-2}{2\alpha-1}(m+1)\delta$, *where* $\alpha = \left\lfloor \frac{m+1}{2} \right\rfloor$.



(a) $d_{min}(\pi^*) < \Delta_{\pi^*} - \frac{3\alpha-2}{2\alpha-1}(m+1)\delta$ (b) $d_{min}(\pi^*) > \Delta_{\pi^*} - \frac{3\alpha-2}{2\alpha-1}(m+1)\delta$

Figure 3.2: Illustration of the proof of Proposition 3.6, representing cycle times of π_{oe}, π_{dh}, f (Equation (3.8)) and the lower bound on the cycle time T_{π^*} given by Proposition 2.3.

Proof. Assume δ is fixed, and call $T_c(p)$ the cycle time of any cycle c depending on p. We assume $p \leq (3m-1)\delta$.

Let $p^* = \frac{3\alpha-1}{2\alpha-1}(m+1)\delta$ be the value of parameter p for which π_{dh} and π_{oe} have the same cycle time.

$$T_{\pi_{oe}}(p^*) = T_{\pi_{dh}}(p^*) = 3(m+1)\delta$$

Let f be the linear function with coefficient 1 verifying $f(p^*) = T_{\pi_{oe}}(p^*) = T_{\pi_{dh}}(p^*) = 3(m+1)\delta$ (see Figure 3.2 for a graphical representation). The expression of f is

$$f(p) = p + \frac{3\alpha-2}{2\alpha-1}(m+1)\delta \tag{3.8}$$

For $(m+1)\delta \leq p \leq p^*$,

$$T_{\pi_{dh}}(p) \geq f(p) \geq T_{\pi_{oe}}(p)$$

For $p^* \leq p \leq (3m-1)\delta$,

$$T_{\pi_{oe}}(p) \geq f(p) \geq T_{\pi_{dh}}(p)$$

Suppose that $d_{min}(\pi^*) \leq \Delta_{\pi^*} - \frac{3\alpha-2}{2\alpha-1}(m+1)\delta$. This is the situation schematized

on Figure 3.2a. By Proposition 2.3,

$$T_{\pi^*}(p) \geq \Delta_{\pi^*} + max(0, p - d_{min}(\pi^*)) \tag{3.9}$$

$$\geq \Delta_{\pi^*} + max(0, p - \Delta_{\pi^*} + \frac{3\alpha - 2}{2\alpha - 1}(m+1)\delta) \tag{3.10}$$

For $p \leq \Delta_{\pi^*} - \frac{3\alpha-2}{2\alpha-1}(m+1)\delta$, we get

$$T_{\pi^*}(p) \geq \Delta_{\pi^*} \geq p + \frac{3\alpha - 2}{2\alpha - 1}(m+1)\delta = f(p)$$

and for $p > \Delta_{\pi^*} - \frac{3\alpha-2}{2\alpha-1}(m+1)\delta$,

$$T_{\pi^*}(p) \geq \Delta_{\pi^*} + p - \Delta_{\pi^*} + \frac{3\alpha - 2}{2\alpha - 1}(m+1)\delta = p + \frac{3\alpha - 2}{2\alpha - 1}(m+1)\delta = f(p)$$

$T_{\pi^*} \geq f$ and thus π^* is dominated by $\{\pi_{oe}, \pi_{dh}\}$ for $(m+1)\delta < p < (3m-1)\delta$

So necessarily, $d_{min}(\pi^*) > \Delta_{\pi^*} - \frac{3\alpha-2}{2\alpha-1}(m+1)\delta$ (this is the situation represented on Figure 3.2b).

□

3.3 Best 1-cycles for $m \leq 8$

We know that 1-cycles are permutations of activities. Given a 1-cycle, it is easy to check if it complies with the conditions exposed in the previous section. Of course this can only be done for small numbers of machines as the number of 1-cycles is exponential ($m!$). In this section, we use these properties to prove the following theorem, establishing dominant sets over 1-cycles for $m \leq 8$.

Theorem 3.1. *Dominance within 1-cycle for $m \leq 8$*

For regular balanced cells with $m = 3$ or $6 \leq m \leq 8$ the set $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ dominates all 1-cycles.

For regular balanced cells with $m = 4$, the set $\{\pi_{id}, \pi_{oe}, \pi_4^*, \pi_{dh}\}$, where $\pi_4^* = A_0A_3A_2A_1A_4$ dominates all 1-cycles.

For regular balanced cells with $m = 5$, the set $\{\pi_{id}, \pi_{oe}, \pi_5^*, \pi_{dh}\}$, where $\pi_5^* = A_0A_3A_2A_5A_1A_4$ dominates all 1-cycles.

3.3.1 $m \leq 4$

For $m = 3$ and $m = 4$, we use Proposition 3.5 and Proposition 3.6 to filter out cycles. Among the six possible cycles for $m = 3$, only π_{oe} and π_{id} satisfy both conditions, thus $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ dominates 1-cycles. This confirms the result presented in Brauner (1999).

Thus the best 1-cycles for $m = 3$ are given by:

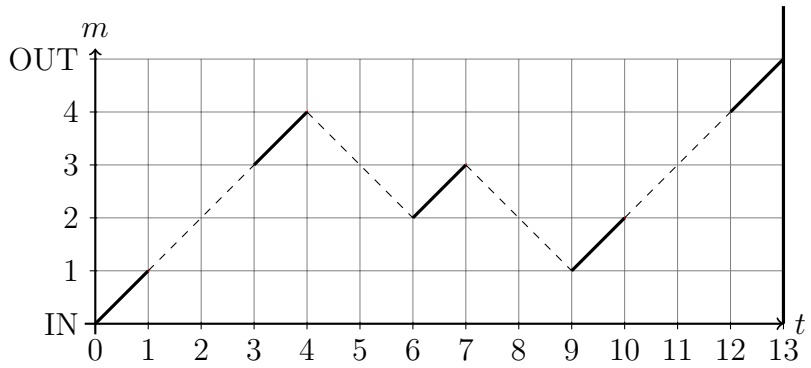


Figure 3.3: Cycle $\pi_4^* = A_0A_3A_2A_1A_4$

- for $p \leq \frac{4}{3}\delta$, π_{id} is optimal within 1-cycle;
- for $\frac{4}{3}\delta < p \leq \frac{20}{3}\delta$, π_{oe} is optimal within 1-cycles
- for $p > \frac{20}{3}\delta$, π_{dh} is optimal within 1-cycles.

Among the 24 possible 1-cycles for $m = 4$, π_{oe}, π_{dh} as well as another permutation $\pi_4^* = A_0A_3A_2A_1A_4$, represented on Figure 3.3 satisfy both conditions. The best 1-cycle for $m = 4$ can be determined by comparing the cycle times of $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ and π_4^*.

Proposition 3.7. *Best 1-cycles for m=4*
The best 1-cycles for $m = 4$ are given by:

- *for $p \leq \frac{5}{4}\delta$, π_{id} is optimal within 1-cycle;*
- *for $\frac{5}{4}\delta < p \leq 7\delta$, π_{oe} is optimal within 1-cycles*
- *for $7\delta < p \leq 10\delta$, π_4^* is optimal within 1-cycles*
- *for $p \geq 10\delta$, π_{dh} is optimal within 1-cycles.*

Proof. For $p \leq 5\delta$, and $p \geq 11\delta$, this is a direct consequence of Proposition 3.2.

The characteristics of π_4^* are as follows: $\Delta_{\pi_4^*} = 13\delta$, $d_1(\pi_4^*) = d_4(\pi_4^*) = d_{min}(\pi_4^*) = 8\delta$ and $d_2(\pi_4^*) = d_3(\pi_4^*) = 9\delta$.

For $5\delta < p < 11\delta$, the cycle time of π_{oe} and π_{dh} are given by

$$T_{\pi_{oe}}(p) = \frac{3}{2}p + \frac{5}{2}\delta \tag{3.11a}$$

$$T_{\pi_{dh}}(p) = 15\delta \tag{3.11b}$$

By Proposition 2.3, we have

$$T_{\pi_4^*}(p) \geq 13\delta + max(0, p - 8\delta) \tag{3.12}$$

For $p \geq 10$, from Equation (3.11b) and Equation (3.12), π_4^* is dominated by π_{dh}. We can show that for $p \leq 10$, the bound given by Equation (3.12) is tight. For $p \leq 8\delta$, no waiting time is necessary. For $8\delta < p \leq 9\delta$ the bound is reached by waiting exactly $p - 8\delta$ at machine M_1; for $9\delta < p \leq 10\delta$, it is reached by waiting exactly $p - 9\delta$ at machine M_2 and 1δ at M_1. We can easily verify that no other waiting time is necessary. Together with Equation (3.11a), we get the final result. □

3.3.2 $m \geq 5$

For $m \geq 5$, we can use Proposition 3.4, Proposition 3.5 and Proposition 3.6 to filter out cycles. Note that Proposition 3.4 and Proposition 3.6 imply Proposition 3.3: if $\Delta_\pi > 2(m+1)\delta$, then the condition in Proposition 3.6 implies that if π is non-dominated by $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$, $d_{min}(\pi) > 0$. Thus no loading of a machine is immediately followed by its unloading, which means that π contains no sequence $A_i A_{i+1}$ with $i \neq m$.

Thus, we seek cycles satisfying:

$$2(m+1)\delta < \Delta_{\pi^*} < 3(m+1)\delta \tag{3.13}$$

$$d_{min}(\pi^*) > \Delta_{\pi^*} - \frac{3\alpha - 2}{2\alpha - 1}(m+1)\delta \tag{3.14}$$

For $m = 5$, only the cycle $\pi_5^* = A_0 A_3 A_2 A_5 A_1 A_4$, represented on Figure 3.4 satisfies these conditions, as can be verified by simple enumeration of the 120 1-cycles.

Proposition 3.8. *Best 1-cycles for m=5*

The best 1-cycles for $m = 5$ are given by:

- *for $p \leq \frac{6}{5}\delta$, π_{id} is optimal within 1-cycles;*
- *for $\frac{6}{5}\delta < p \leq \frac{42}{5}\delta$, π_{oe} is optimal within 1-cycles;*
- *for $\frac{42}{5}\delta < p \leq 12\delta$, π_5^* is optimal within 1-cycles;*
- *for $p \geq 12\delta$, π_{dh} is optimal within 1-cycles.*

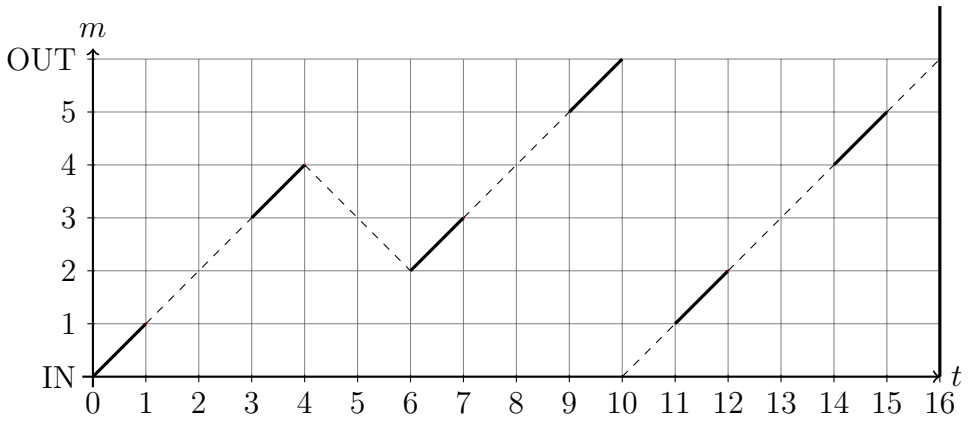


Figure 3.4: Cycle $\pi_5^* = A_0A_3A_2A_5A_1A_4$

Proof. For $p \leq 6\delta$, and $p \geq 14\delta$, this is a direct consequence of Proposition 3.2.

The characteristics of π_5^* are as follows: $\Delta_{\pi_5^*} = 16\delta$, $d_1(\pi_5^*) = d_2(\pi_5^*) = d_4(\pi_5^*) = d_5(\pi_5^*) = d_{min}(\pi_5^*) = 10\delta$ and $d_3(\pi_5^*) = 12\delta$. For $6\delta < p < 14\delta$, the cycle times of π_{oe} and π_{dh} are given by

$$T_{\pi_{oe}}(p) = \frac{5}{3}p + 2\delta \tag{3.15a}$$

$$T_{\pi_{dh}}(p) = 18\delta \tag{3.15b}$$

By Proposition 2.3, we have

$$T_{\pi_5^*}(p) \geq 16\delta + max(0, p - 10\delta) \tag{3.16}$$

For $p \geq 12\delta$, from Equation (3.15b) and Equation (3.16), π_5^* is dominated by π_{dh}.

We can show that for $p \leq 12\delta$, the bound given by Equation (3.16) is tight: for $p \leq 10\delta$, no waiting time is necessary; for $10\delta < p \leq 12\delta$, the bound is reached by waiting exactly $p - 10\delta$ at machine M_2. We can easily verify that no other waiting time is necessary. From this and Equation (3.15a), we get the final result. □

For $6 \leq m \leq 8$, no 1-cycle satisfies the conditions (this can be verified by enumerating the $m!$ possible 1-cycles), thus $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ dominates 1-cycles. This completes the proof of Theorem 3.1.

For $m > 8$, this filtering does not seem sufficient to rule out all dominated 1-cycles. This case will be the topic of Chapter 4.

3.4 A counter-example to the 1-cycle conjecture

In (Brauner, 2008), the author claims that in linear balanced cells, the 1-cycle conjecture is valid at least for $m < 16$. We state, by the following theorem, that

in circular balanced cells it is false for $m = 6$. For $m \leq 5$, it is still open but using the simulation tool presented in Chapter 2, we did not find any 2-cycle performing strictly better than the best 1-cycles.

Theorem 3.2. *In a regular balanced unbounded 6-machine cell with circular lay-out, the 2-cycle*

$$\hat{C} = (A_0A_2A_5A_4A_1A_6A_0A_3A_2A_5A_1A_4A_3A_6)$$

dominates all 1-cycles for the following instance:

$$\delta = 1 \quad \epsilon = 0 \quad p = 11$$

.

Proof. Let us consider a 6-machine circular cell, with $p = 11$, $\delta = 1$, $\epsilon = 0$. From Theorem 3.1, we know that the set $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ dominates all 1-cycles.

For these parameters, one has

$$T_{\pi_{id}} = 7 + 66 = 73 \tag{3.17}$$

$$T_{\pi_{oe}} = 14 + \frac{5}{3}4 = 20 + \frac{2}{3} \tag{3.18}$$

$$T_{\pi_{dh}} = 21 \tag{3.19}$$

We just need to show that $\frac{T(\hat{C})}{k} < T_{\pi_{oe}} = 20 + \frac{2}{3}$. An iteration of $\hat{C}$ is represented on Figure 3.5.

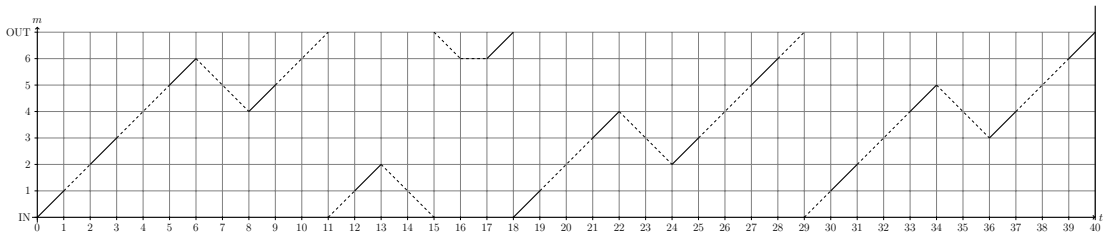


Figure 3.5: An iteration of C, for $p = 11$ and $\delta = 1$

During one iteration of $\hat{C}$, the robot travels $\Delta_{\hat{C}} = 39$.

We have $d_{6,1} = d_{2,1} = 10$, and for all other (i, k), $d_{i,k} \geq 11$. As the first unloading of M_6 takes place between the first loading of M_2 and its subsequent unloading, the robot only needs to wait one unit of time, before unloading M_6 (see Figure 3.5).

Thus, we have $T_{\hat{C}} = 39 + 1 = 40$

$$\frac{T_{\hat{C}}}{2} = 20 < T(\pi_{oe})$$

□

In conclusion, $\hat{C}$ is strictly better than all 1-cycles on this instance. We proved Theorem 3.2, thus showing that the 1-cycle conjecture is false for 6 machines in circular balanced cells. The problem is still open for $m \leq 5$ (although we think it is valid).

3.5 Conclusion

In this chapter, we focused on 1-cycles. Using the lower bounds and the cycle time of the odd-even cycle π_{oe} derived in Chapter 2, we settled values of parameters p and δ for which one of the classical cycles $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ is optimal. Together with necessary properties of optimal 1-cycles we were able to solve the best 1-cycle problem for cells with up to 8 machines. However, we showed that the 1-cycle conjecture is false for circular balanced cells, by exhibiting a 2-cycle which performs strictly better than any 1-cycle for a given instance of 6 machines. The best k-cycle even for $k \leq 2$ is still open for circular regular balanced cells with $m \geq 3$.

The properties established in this chapter are not sufficient to conclude on the best 1-cycle problem for $m \geq 9$. In the next chapter, we look into the structure of optimal 1-cycles for large cells.

Chapter 4

Dominant structures for large cells

> Life was a lot less complex *before* you tried to explain.
>
> Jasper Fforde, *Shades of Grey*

4.1 Introduction

In this chapter, we study the best 1-cycle problem in regular balanced cells with any number of machine m. One-cycles are not necessarily optimal among general cycles, even for cells with circular layout: an instance for which a 2-cycle is better than any 1-cycle is described in the previous chapter. Still, finding the best 1-cycle remains a problem of interest due to their operational convenience. Interestingly, the layout of the cell has an impact on this problem's complexity. While it has been shown polynomial in linear additive cells (Crama and van de Klundert, 1997a), this result, based on the dominance of pyramidal permutations, does not stand for circular layouts: Rajapakshe et al. (2011) showed that finding the best 1-cycle turns to be NP-hard in a circular additive cell, even assuming regularity (regularly spaced machines). Yet for the simpler model of circular regular *balanced* cell (where processing times are equals for all machines), the complexity of finding the best 1-cycle is still unknown. Studying structural properties of good cycles in this simpler case might lead to progress on this issue and to a better understanding of the difference of complexity between both layouts in the more general regular case.

For small-scale regular balanced cells ($m \leq 8$), the best-1-cycle problem has been solved in the previous chapter. We showed in particular that for $6 \leq m \leq 8$,

the three classical cycles introduced in Chapter 2, $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$, dominate 1-cycles (meaning that for any instance, one of these three cycles performs at least as well as any 1-cycle). However, the proof cannot be extended to larger cells. In this chapter, we are interested in cells with an arbitrarily large number of machine m, and thus assume $m > 8$.

Section 4.2 recalls the optimality regions of the classical cycles $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$. In Section 4.3, we present observations resulting from numerical computation of cycle times for $9 \leq m \leq 14$. Section 4.4 investigates general structural properties of cycles likely to dominate $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$. In Section 4.5, we focus on a family of cycles generalizing observations in Section 4.3 and derive their cycle times. Section 4.6 compares this family to more general cycles obeying the dominant structure presented in Section 4.4. Based on this, in Section 4.7, we conjecture that the family presented in Section 4.5 dominates 1-cycles. This conjecture is proven valid for $m \leq 11$ and holds up with numerical computation for $m \leq 14$. If proven valid for any m, it would imply that the best 1-cycle problem in circular regular balanced cells is polynomial.

4.2 Region of optimality of the classical cycles

The three classical 1-cycles, namely the identity cycle $\pi_{id} = A_0A_1 \dots A_m$, the downhill cycle $\pi_{dh} = A_0A_mA_{m-1} \dots A_1$ and the odd-even cycle $\pi_{oe} = A_0A_2A_4 \dots A_1A_3A_5 \dots$ have been introduced in Chapter 2, page 26. Some regions of optimality of these cycle have been established in Chapter 2, page 28 and Chapter 3, page 36:

(i) if $p \leq \frac{m+1}{m}\delta$, then the identity permutation π_{id} dominates 1-cycles;

(ii) if $\frac{(m+1)}{m}\delta \leq p \leq (m+1)\delta$, then the odd-even cycle π_{oe} dominates 1-cycles;

(iii) if $p \geq 3(m-1)\delta$, then the downhill permutation π_d dominates 1-cycles.

Thus, in this chapter, we only consider the remaining case, where $(m+1)\delta < p < (3m-1)\delta$. Recall that for these values of p, cycle π_{id} is dominated by π_{oe} and need not be considered, while the cycle times of π_{dh} and π_{oe} are given by:

$$T_{\pi_{dh}}(p) = 3(m+1)\delta \tag{4.1}$$

$$T_{\pi_{oe}}(p) = 2(m+1)\delta + \frac{2\alpha-1}{\alpha}(p-(m+1)\delta) \text{ with } \alpha = \left\lfloor \frac{m+1}{2} \right\rfloor \tag{4.2}$$

Recall also that p^* denotes the value of the processing time such that $T_{\pi_{oe}}(p^*) = T_{\pi_{dh}}(p^*)$. Its value is $p^* = \frac{3\alpha-1}{2\alpha-1}(m+1)\delta$. Over the considered interval, p^* is

the smallest value for which $\min(T_{\pi_{oe}}, T_{\pi_{dh}})$ reaches its maximum (see Figure 4.1 to visualize this property). Potential best 1-cycles must be non-dominated by $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$, meaning that there exists at least a value of p for which their cycle time is smaller than the cycle time of any member of $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$. The following proposition further restricts the interval to consider to find such a value of p and to show that a cycle is non-dominated by $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$, using this property of p^*.

Proposition 4.1. *If a cycle c is non-dominated by $\{\pi_{oe}, \pi_{dh}\}$ over $](m+1)\delta,\ (3m-1)\delta[$ then it is non-dominated by $\{\pi_{oe}, \pi_{dh}\}$ over $](m+1)\delta,\ p^*]$.*

Proof. This is a direct consequence of the cycle time being an increasing function of the processing time and $\min(T_{\pi_{oe}}(p), T_{\pi_{dh}}(p))$ reaching its maximum in p^*. Consider $p \in](m+1)\delta,\ (3m-1)\delta[$ such that $T_c(p) < \min\{T_{\pi_{oe}}(p), T_{\pi_{dh}}(p)\}$. If $p > p^*$, then

$$T_c(p^*) \leq T_c(p) < \min(T_{\pi_{oe}}(p), T_{\pi_{dh}}(p)) \leq 3(m+1)\delta = \min\left(T_{\pi_{oe}}(p^*), T_{\pi_{dh}}(p^*)\right)$$

and hence $T_c(p^*) < \min\left(T_{\pi_{oe}}(p^*), T_{\pi_{dh}}(p^*)\right)$. □

Figure 4.1 summarizes the results presented in this section. This diagram shows the cycle times of $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ and the lower bounds presented in Chapter 2 (LB_M, LB_R) and Chapter 3 (LB_D), outlining parameters regions for which the optimal cycle is known, and the remaining region $p \in](m+1)\delta,\ (3m-1)\delta[$ which we focus on in this chapter. We are looking for potential optimal cycles other than $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$, thus dominating $\{\pi_{oe}, \pi_{dh}\}$ for some value of p: Proposition 4.1 indicates that the cycle times of such cycles necessarily take some values in the hatched area shown on Figure 4.1.

4.3 Experiments

Propositions 3.3 to 3.6 (pages 38 to 40) state that any cycle π dominating $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ for some p in the considered region satisfies the following three structural properties:

$$\forall i \neq m,\ A_i A_{i+1} \notin \pi \tag{4.3}$$

$$2(m+1)\delta < \Delta_\pi < 3(m+1)\delta \tag{4.4}$$

$$d_{min}(\pi) > \Delta_\pi - \frac{3\alpha - 2}{2\alpha - 1}(m+1)\delta \quad \text{where } \alpha = \left\lfloor \frac{m+1}{2} \right\rfloor \tag{4.5}$$

where Δ_π is the total travel time during one iteration of the cycle, and $d_{min}(\pi)$ is the minimum travel time between the loading and unloading of a same machine ($d_{min} = \min_{i \in \{1,\dots,m\}} d_i$).

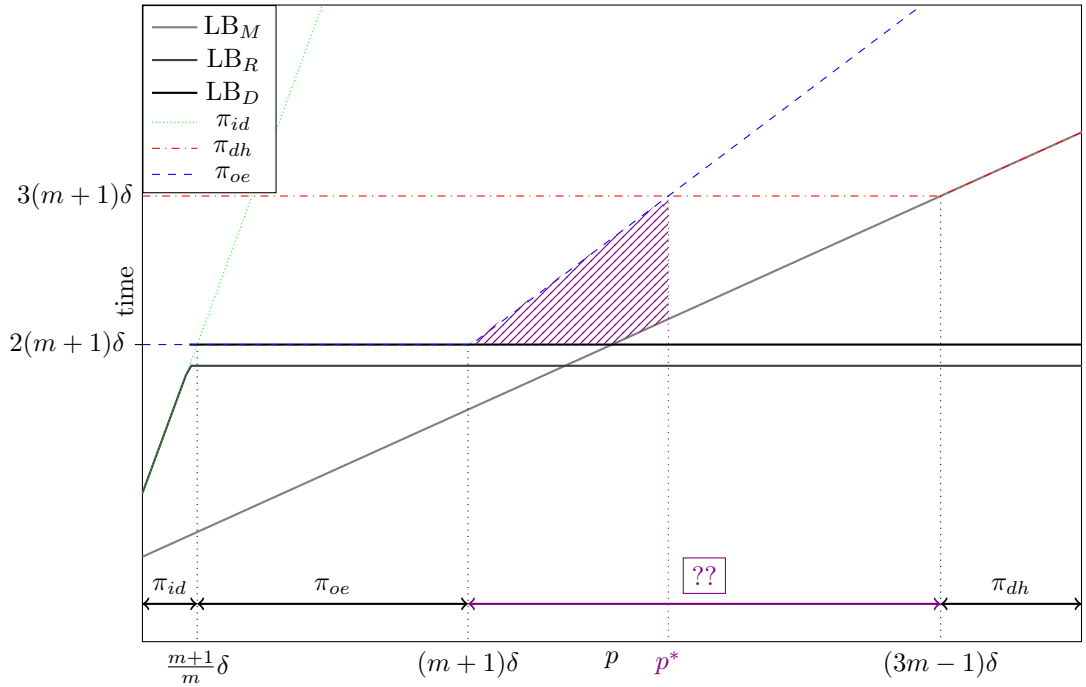


Figure 4.1: Cycle times and known regions of optimality of $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$.

For $6 \leq m \leq 8$, no 1-cycle satisfies these conditions; however for $m > 8$, they do not rule out all 1-cycles. We call $filtered(m)$ the set of 1-cycles satisfying the conditions; Table 4.1 shows the number of such cycles for $9 \leq m \leq 14$. For $m \geq 15$, the total number of 1-cycles is too high for an exhaustive filtering.

m	9	10	11	12	13	14
$\#filtered(m)$	26	23	32	41	63	63

Table 4.1: number of cycles in $filtered(m)$ for $9 \leq m \leq 14$

Using the simulation tools presented in Section 2.4, page 32, we numerically computed the cycle times of these cycles while varying the processing time p. This section presents an overview of the resulting observations.

Although ergodicity of the cycle time is often assumed when studying robotic cells, it has not been formally proven in general. Thus, as the periodic steady state reached by the simulated cell depends on its initial state, so *might* the resulting mean cycle time, and these numerically computed values are only upper bounds on the cycle time. The following numerical observations should therefore be taken with caution: they only serve as a motivation for the subsequent formal study.

4.3.1 Performance

For each $9 \leq m \leq 14$, some cycles of $filtered(m)$ appear to slightly dominate $\{\pi_{oe}, \pi_{dh}\}$ in the neighborhood of p^*, as can be seen for example in Figure 4.2 for 9 and 10 machines.

Table 4.2 shows the computed performance ratio of $\{\pi_{oe}, \pi_{dh}\}$ for $p = p^*$, based on the computed cycle times of members of $filtered(m)$, at p^* where it appears to be the worst. Note the higher ratio for $m = 13$.

m	9	10	11	12	13	14
$\max_{\pi \in filtered(m)} \frac{T_{\pi_{dh}}(p^*)}{T_{\pi}(p^*)}$	1.015	1.028	1.005	1.026	1.050	1.005

Table 4.2: Performance ratio of $\{\pi_{oe}, \pi_{dh}\}$ for $\delta = 1$, $p = p^*$, using the computed cycle times of cycles in $filtered(m)$.

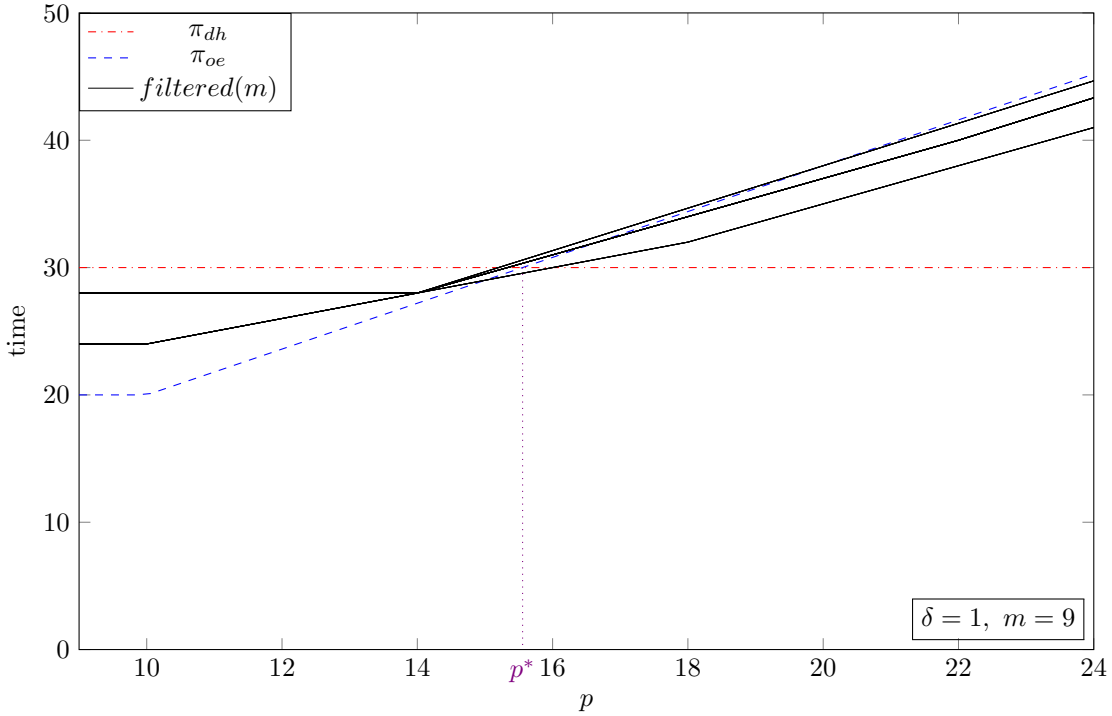
4.3.2 Set of dominant 1-cycles

Many of the cycles in $filtered(m)$ are equivalent or strictly dominated by others within the considered area. Looking closer, it appears that few cycles (1 or 2) need to be added to the set $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ to form a dominant set within 1-cycles. This is apparent on Figure 4.3, which shows computed cycle times for members of $filtered(m)$ in p^* neighborhood.

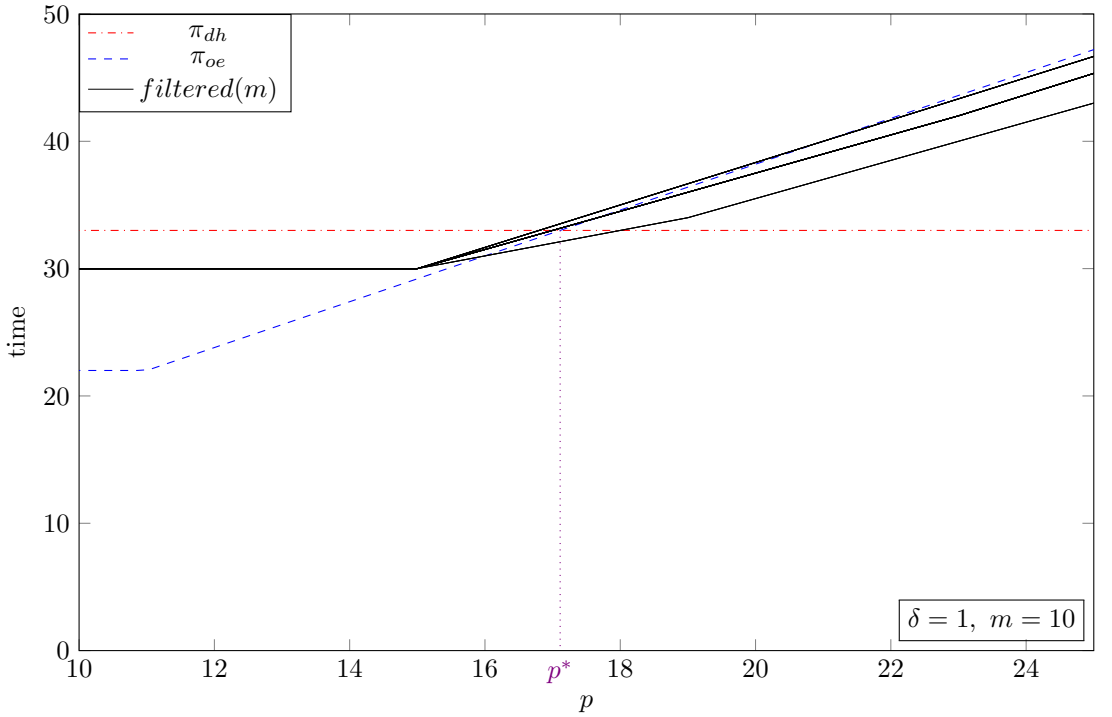
Table 4.3 shows, for $9 \leq m \leq 14$, a possible dominant set and the numerical optimality regions, computed using a step of 0.01 for p. It seems that only one or two cycles need do be added to $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ to form a dominant set. Note that the set is not unique, except for $m = 13$.

4.3.3 Structure of dominant 1-cycles

One-cycles which outperform $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ are similar in structure. Like π_{oe}, one iteration consists of 2 turns of the cell, but with additional backward sequences (two for $9 \leq m \leq 11$ and $m = 14$, two or three for $m = 12$, three for $m = 13$). Examples are presented in Figure 4.4. These sequences add to the total travel time but also increases the time travelled between the loading and unloading of any machine (increasing d_{min} to $d_{min} = (m + 5)\delta$ for $9 \leq m \leq 11$ and $m = 14$; $d_{min} = (m+5)\delta$ or $d_{min} = (m+9)\delta$ for $m = 12$; $d_{min} = (m+9)\delta$ for $m = 13$). When several cycles appear to be equivalent, the relative placement of the backward sequences is the same.



(a) m=9



(b) m=10

Figure 4.2: π_{dh}, π_{oe} and computed cycle times of all members of $filtered(m)$ as functions of p. The step used for p is 0.1. Notice that many cycles have the same cycle times.

$m+1 \leq p \leq 15.00$	π_{oe}
$15.00 \leq p \leq 16.00$	$A_0A_3A_2A_5A_8A_1A_4A_7A_6A_9$
$16.00 \leq p \leq 3m-1$	π_{dh}

(a) $m = 9$ ($p^* \approx 15.56$)

$m+1\delta \leq p \leq 16.00$	π_{oe}
$16.00 \leq p \leq 18.00$	$A_0A_3A_2A_5A_8A_{10}A_1A_4A_7A_6A_9$
$18.00 \leq p \leq 3m-1$	π_{dh}

(b) $m = 10$ ($p^* \approx 17.11$)

$m+1\delta \leq p \leq 18.00$	π_{oe}
$18.00 \leq p \leq 18.66$	$A_0A_3A_2A_5A_8A_{10}A_1A_4A_7A_6A_9A_{11}$
$18.67 \leq p \leq 3m-1$	π_{dh}

(c) $m = 11$ ($p^* \approx 18.55$)

$m+1 \leq p \leq 19.00$	π_{oe}
$19.00 \leq p \leq 19.66$	$A_0A_3A_2A_5A_8A_{10}A_{12}A_1A_4A_7A_6A_9A_{11}$
$19.67 \leq p \leq 21.66$	$A_0A_3A_2A_5A_8A_{11}A_{10}A_1A_4A_7A_6A_9A_{12}$
$21.67 \leq p \leq 3m-1$	π_{dh}

(d) $m = 12$ ($p^* \approx 20.09$)

$m+1 \leq p \leq 20.46$	π_{oe}
$20.47 \leq p \leq 23.33$	$A_0A_3A_2A_5A_8A_{11}A_{10}A_{13}A_1A_4A_7A_6A_9A_{12}$*
$23.34 \leq p \leq 3m-1$	π_{dh}

(e) $m = 13$ ($p^* \approx 21.54$) (*unique)

$m+1 \leq p \leq 22.0$	π_{oe}
$22.0 \leq p \leq 23.20$	$A_0A_3A_2A_5A_8A_{10}A_{12}A_{14}A_1A_4A_7A_6A_9A_{11}A_{13}$
$23.20 \leq p \leq 3m-1$	π_{dh}

(f) $m = 14$ ($p^* \approx 23.08$)

Table 4.3: Best 1-cycle depending on p, for $9 \leq m \leq 14$ and $\delta = 1$.

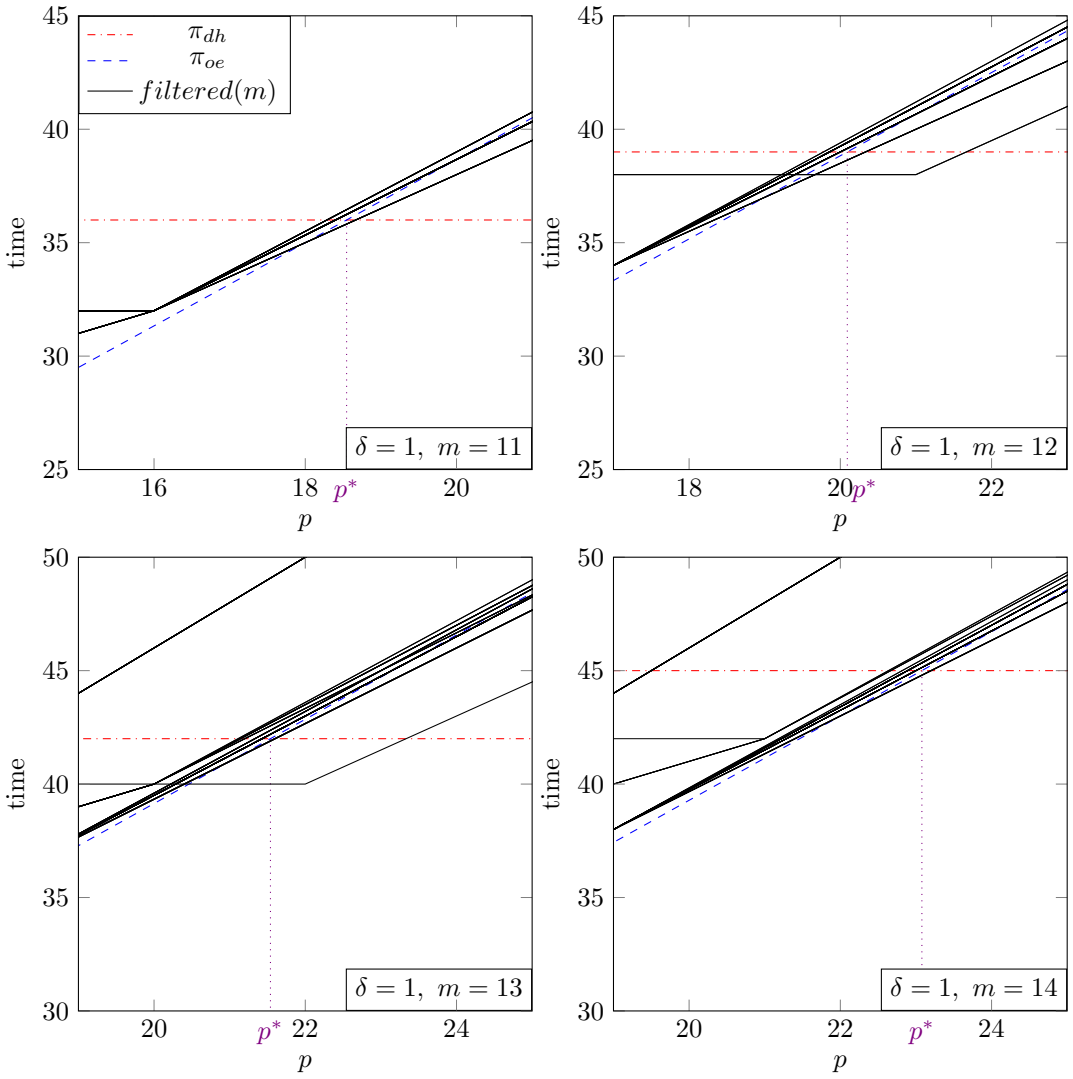


Figure 4.3: π_{oe}, π_{dh} and computed cycle times of all members of $filtered(m)$ as functions of p, zoomed around p^*. The step used for p is 0.1.

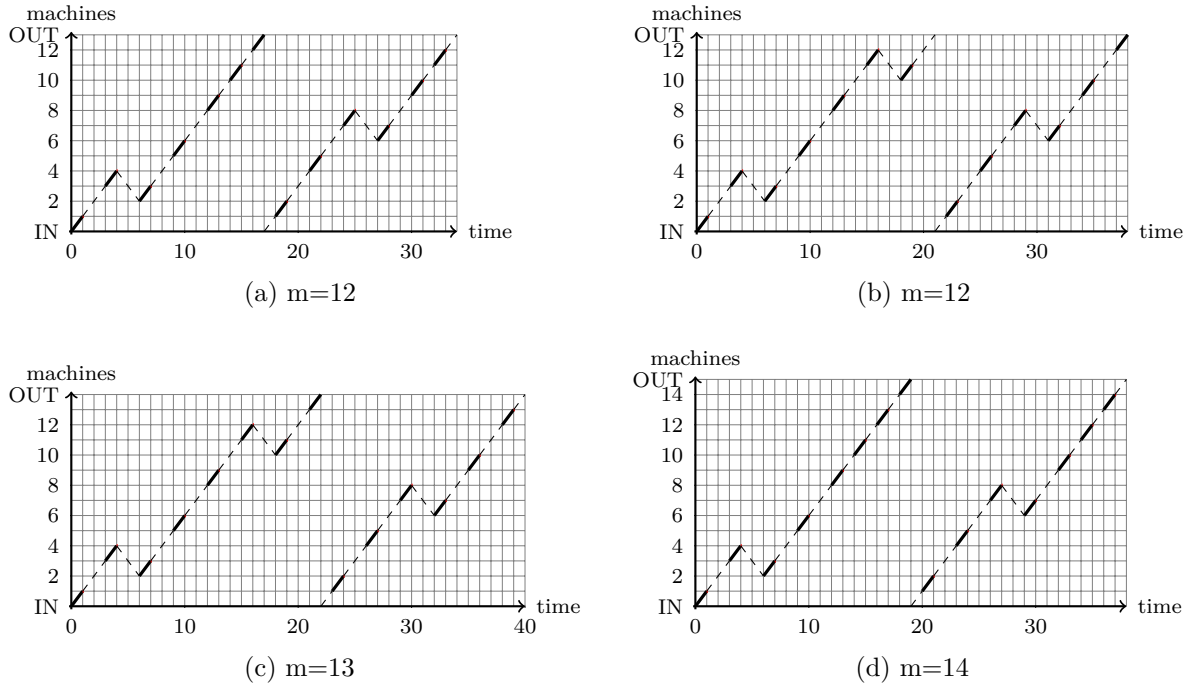


Figure 4.4: Some 1-cycles which outperform $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ in the neighborhood of p^*. For 12 machines, note that the two cycle presented are not equivalent: one has two backward sequences and the other has three.

4.4 General structure of performant cycles

The experimental results obtained in Section 4.3 suggest that any cycle dominating $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ consists in two forward turns around the cell, similarly to the odd-even cycle, altered by a number of backward sequences $A_i A_{i-h}$. We will call such alterations "waves", referring to their aspects on the cycle graphical representation. In this section, we define formally such cycles and derive some of their properties.

4.4.1 Two turns...

In the following, unless otherwise specified, $m > 8$ and $(m+1)\delta < p < (3m-1)\delta$ and π refers to a 1-cycle dominating $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ for some $(m+1)\delta < p < (3m-1)\delta$. In this interval, cycle time values for π_{dh} and π_{oe} are given respectively by Equation (4.1) and Equation (4.2). Note that the cycle time of π_{oe} satisfies:

$$T_{\pi_{oe}}(p) < 2p \tag{4.6}$$

Remember (inequality (4.4)) that the total travel time (denoted by Δ_π) over one iteration of cycle π must be comprised between two and three times the time necessary to circle the cell once:

$$2(m+1)\delta < \Delta_\pi < 3(m+1)\delta \tag{4.7}$$

This relation alone is not sufficient to conclude that the robot effectively circles the cell twice. This section aims at showing this is indeed the case; formally, that there is at least one spot where the robot passes *exactly* twice, both times while traveling in the *forward* direction.

We use the following notations: for any $i \in \{0, \dots, m\}$, $m_\pi(i)$ denotes the number of times the robot travels between M_i and M_{i+1} in either direction (forward or backward) during one iteration of cycle π. Similarly, $m^+_\pi(i)$ (respectively $m^-_\pi(i)$) denotes the number of times the robot travels forward (respectively backward) from M_i and M_{i+1}. When there is no ambiguity as for the referenced cycle, we omit it and simply use $m(i)$, $m^+(i)$ and $m^-(i)$.

Note that $\Delta_\pi = \delta \sum_{i \in \{0,\dots,m\}} m_\pi(i)$.

Proposition 4.2. *Let π be a 1-cycle such that $T_\pi(p) < \min(T_{\pi_{oe}}(p), T_{\pi_{dh}}(p))$ for some $p \in](m+1)\delta, 3(m+1)\delta[$. Then:*

$$\forall i \in \{0, \dots, m\} \quad m_\pi(i) \geq 2$$

and

$$\exists j \in \{0, \dots, m\} \quad m_\pi(j) = 2$$

Proof. First, we show that if $m(i) = 1$ for some i, then π is dominated by π_{oe} or π_{dh}.

Case 1. $i \notin \{0, m\}$. Looking at π's sequence starting with A_i, we consider cases depending on the respective positions of A_{i+1} and A_{i-1}.

Case 1.1. If A_{i+1} precedes A_{i-1}, then $T_\pi(p) \geq 2p > T_{\pi_{oe}}$: processing on machines M_{i+1} and M_i do not overlap, and both require p time units (see Figure 4.5a).

Case 1.2. If A_{i-1} precedes A_{i+1}, then (see Figure 4.5b):

- Between the beginning of A_i and the end of A_{i-1}, the robot cannot pass a second time between M_i and M_{i+1} (otherwise $m_i \geq 2$), hence it has to travel forward between M_{i+1} and M_i, thus circling the whole cell and traveling at least $(m+1)\delta$.
- Likewise, between the end of A_{i-1} and the beginning of A_{i+1}, the robot circles the cell backward, traveling at least $m\delta$.
- Likewise, between the beginning of A_{i+1} and the beginning of the next iteration, the robot circles the cell forward, traveling at least $m\delta$.

Moreover as the fastest path from M_i to M_{i+1} is forward, and the robots must take the fastest path to join subsequent activities, at least one activity is performed between A_{i-1} and A_{i+1}, adding at least 2δ travel time compared to a straight backward route.

Finally, $\Delta_\pi \geq (m+1)\delta + 2m\delta + 2\delta = 3(m+1)\delta$, and π is dominated by π_{dh}.

Case 2. $i = 0$. Looking at π's sequence starting with A_0, we consider cases depending on the respective positions of A_1 and A_m.

Case 2.1. A_m precedes A_1. This case is similar to Case 1.2; see Figure 4.5c.

Case 2.2. A_1 precedes A_m. Let A_k be the highest activity (largest k) performed between A_0 and A_1.

Case 2.2.1. $k < m - 2$. Activities higher than A_k occur after A_1 and therefore must be performed in decreasing order, otherwise $T_\pi(p) \geq 2p \geq T_{\pi_{oe}}(p)$ (see Figure 4.5d). We have $m(0) = 1$, $m(j) \geq 3$ for $0 < j \leq k+1$, $m(j) \geq 4$ for $k+1 < j < m$ and $m(m) \geq 3$ (see Figure 4.5e). Thus $\Delta_\pi \geq 3(m+1)\delta$, and π is dominated by π_{dh}.

Case 2.2.2. $k = m-2$. Likewise, A_{m-1} and A_m must be performed in decreasing order. $m(0) = 1$ and $m(j) \geq 3$ for $0 < j \leq m$, which leads to $\Delta_\pi \geq (3m+1)\delta$. Additionally, as the fastest path from M_{m-2} to M_1 is forward ($m - 2 - 1 > m + 1 - (m - 2 - 1)$), at least one activity is performed between A_{m-2} and A_1, adding at least 2δ travel time, leading to $\Delta_\pi \geq 3(m+1)\delta$ (see Figure 4.5f). Thus π is dominated by π_{dh}.

Case 2.2.3. $k = m - 1$. We have $m(0) = m(m) = 1$ and $m(j) \geq 3$ for $0 < j < m$, thus $\Delta_\pi \geq (3m-1)\delta$. If A_2 is after A_{m-1} (Figure 4.5g), then $m(2) \geq 4$. Additionally, at least one activity is performed between A_{m-1} and A_2, since otherwise the robot would have gone forward from M_m to M_2. This adds an additional 2δ, leading to $\Delta_\pi \geq 3(m+1)\delta$. On the other hand, if A_2 takes place before A_{m-1}, then, the robot travels at least $2m\delta$ between the beginning of A_2 and the end of A_1, and travels or waits at least p between A_1 and the next unloading of M_1 (see Figure 4.5h). The cycle time (calculated from the beginning of A_2) $T_\pi(p)$ is at least $2m\delta + p$. If $p \geq (m+3)\delta$, then $T_\pi(p) \geq 3(m+1)\delta$ and π is dominated by π_{dh}. Otherwise, $(m+1)\delta < p < (m+3)\delta$: then, $T_\pi(p) \geq 2p$ and π is dominated by π_{dh}.

Case 3. $i = m$. This case is symmetrical to the case where $i = 0$.

Finally, if π dominates $\{\pi_{oe}, \pi_{dh}\}$ for some $p \in](m+1)\delta, (3m-1)\delta[$, then for all $i \in \{0, \ldots, m\}$, $m_\pi(i) \geq 2$. Moreover, inequality (4.7) implies that there exist $j \in \{0, \ldots, m\}$ such that $m_\pi(j) = 2$ (otherwise, $\Delta_\pi \geq 3(m+1)\delta$). □

Proposition 4.3.
For $i \in \{0, \ldots, m\}$, if $m_\pi(i) = 2$, then $m^+_\pi(i) = 2$ and $m^-_\pi(i) = 0$.

Proof. For the sake in simplicity, we give the proof for $i = 0$, but it can be adapted to any i in $\{0, \ldots, m\}$. Let us assume that $m^+(0) = 1$ and $m^-(0) = 1$. This means that cycle does not take advantage of the circular layout of the cell. We consider the cycle sequence starting with A_0. It can be decomposed into two parts: the part between A_0 and A_m, and the rest of the sequence following A_m. For any j in $\{1, \ldots, m\}$, $m(j) = 2$ only if the 3 following conditions are satisfied:

(i) A_j is performed between A_0 and A_m

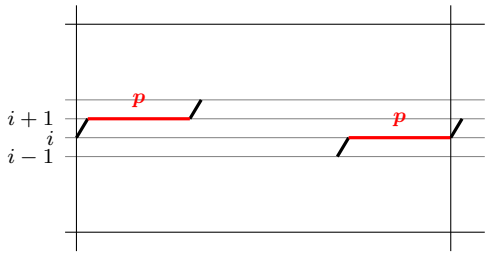
(ii) No activity A_k with $k < j$ is performed between A_j and A_m

(iii) There is no pair of activities A_k, A_l such that $k < j < l$ and A_k and A_l both take place after A_m *in that order.*

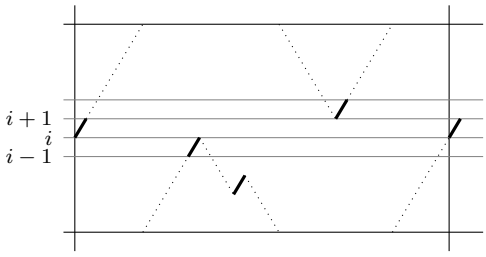
In all other cases, $m(j) \geq 4$:

- If condition (i) is not satisfied, then the robot travels at least twice forward between M_j and M_{j+1} (once between A_0 and A_m and once to perform A_j, and twice backwards (once between A_m and A_j (to get to M_j or a previous machine), and once after A_j (to get back to the input station).

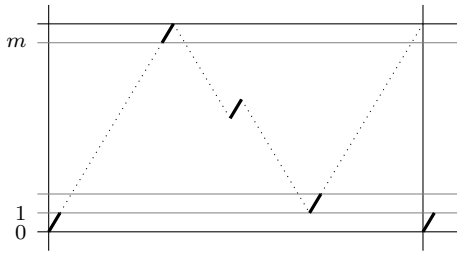
- Otherwise, if (ii) is not satisfied, the robot travels between M_j and M_{j+1} at least once forward performing A_j, once backward between A_j and A_k, once forward between A_k and A_m, and once backward after A_m.



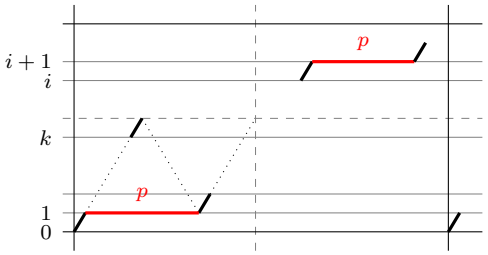
(a) Case 1.1.



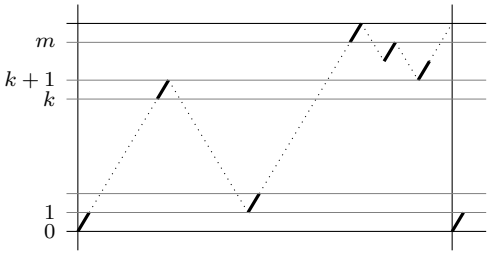
(b) Case 1.2.



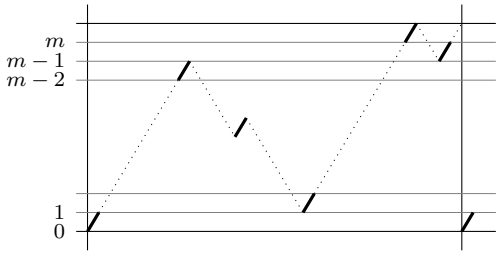
(c) Case 2.1.



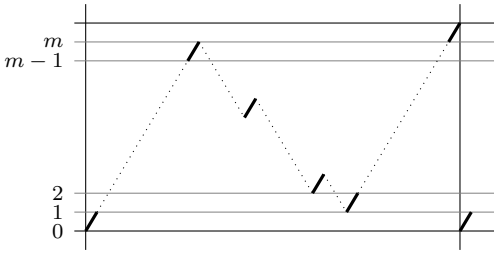
(d) Case 2.2.1. (non-decreasing order after k)



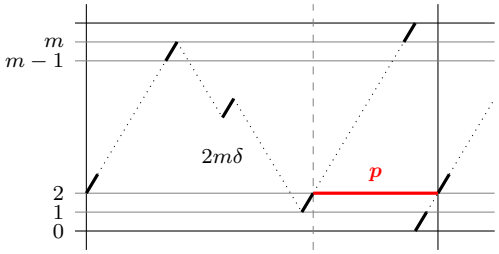
(e) Case 2.2.1. (decreasing order after k)



(f) Case 2.2.2.



(g) Case 2.2.3. (A_2 after A_{m-1})



(h) Case 2.2.4.(A_2 before A_{m-1})

Figure 4.5: For the proof of Proposition 4.2.

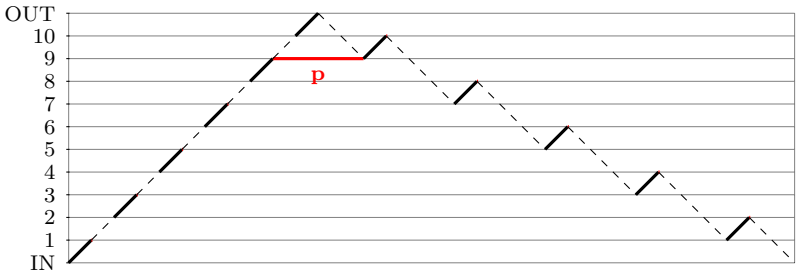


Figure 4.6: For the proof of Proposition 4.3: cycle $A_0A_2 \dots A_{m-2}A_mA_{m-1}A_{m-3} \dots A_1$ (here for $m = 10$)

- Otherwise, if (iii) is not satisfied, the robot travels between M_j and M_{j+1} at least once forward performing A_j, once backward between A_m and A_k, once forward between A_k and A_l, and once backward after A_l.

As π does not contain any subsequence of consecutive activities, at most $\lceil \frac{m+1}{2} \rceil$ activities satisfy these conditions. If at most $\lfloor \frac{m+1}{2} \rfloor$ do so (in particular, this is the case if m is odd), then $T_\pi(p) \geq 2\lfloor \frac{m+1}{2} \rfloor \delta + 4\lceil \frac{m+1}{2} \rceil \delta \geq 3(m+1)\delta$ and π is dominated by π_{dh}.

Otherwise, the only possible cycle is $\pi = A_0A_2 \dots A_{m-2}A_mA_{m-1}A_{m-3} \dots A_1$ (see Figure 4.6). Then we have $T_\pi \geq 2\left(\lceil \frac{m+1}{2} \rceil - 1\right)\delta + p + 4\left(\lfloor \frac{m+1}{2} \rfloor - 1\right)\delta \geq 3(m+1)\delta$ (as $p \geq (m+1)\delta$)), thus π is dominated by π_{dh}.

□

Definition 4.1. Consider $i_0 \in \{0, \dots, m\}$ such that $m^+_{i_0}(\pi) = 2$. The robot travels twice between M_{i_0} and M_{i_0+1}: once loaded while performing A_{i_0}, and once empty. Starting an iteration of the cycle with A_{i_0}, we define all robot moves from A_{i_0} included, to the empty travel between M_{i_0} and M_{i_0+1} excluded as constituting the *first turn* of the cycle. All subsequent moves form the *second turn*. Each turn entails at least $(m+1)\delta$ units of travel time.

The following Proposition extends Equation (4.3):

Proposition 4.4. *Two activities A_i and A_{i+1} cannot be performed during the same turn of π in increasing order.*

Proof. Let $i \in \{0, m-1\}$ such that A_i and A_{i+1} take place during a same turn in increasing order. Then, $T_\pi(p) \geq 2(m+1)\delta + p \geq 3(m+1)\delta$ as $p \geq (m+1)\delta$. Thus π is dominated by π_{dh}. □

4.4.2 ...Waves...

In the next two sections, we restrict ourselves to cycles satisfying Proposition 4.3, meaning that there is at least one spot in the cell which the robot crosses exactly twice, both times forward. Graphically, these cycles can be seen as two forward climbs (representing the two turns) altered by "waves" formed by backwards sequences of activities of type A_iA_{i-h}. We call h the size of the wave: for example A_6A_5 is a wave of size 1, and A_7A_4 a wave of size 3. Intuitively, these sequences, at the cost of an additional travel time, increase the amount of travel between the loading of some machines and their subsequent unloading, thus reducing the waiting time.

Figure 4.7 shows how one wave of size h affects the travel time and waiting times.

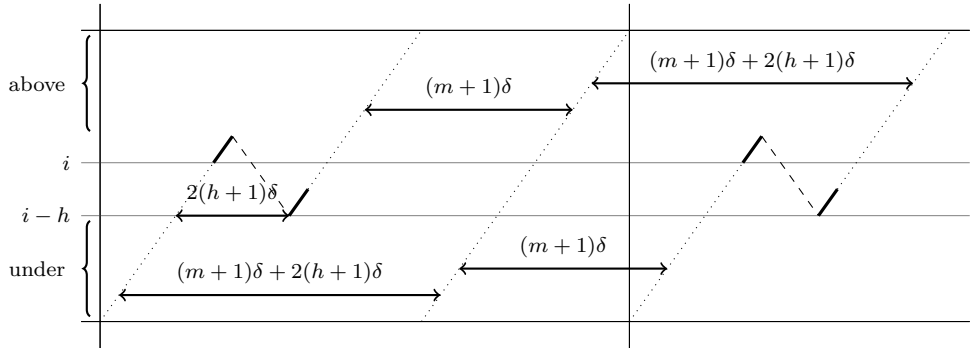


Figure 4.7: One wave

Compared to π_{oe}, $2(h+1)\delta$ is added to the total travel time by one wave of size h. Recall that d_j is the time traveled by the robot between the loading and unloading of machine M_j, $j \in \{1, \ldots, m\}$. These values are also affected depending on when machine M_j is loaded relatively to the sequence A_iA_{i-h} ("under" or "above" in Figure 4.7):

- if M_j is loaded before performing A_iA_{i-h}, during the same turn ("under"), d_j is increased by $2(h+1)\delta$.
- if M_j is loaded after performing A_iA_{i-h}, during the same turn ("above"), d_j doesn't change.
- if M_j is loaded "under" the sequence A_iA_{i-h}, during the other turn, d_j doesn't change.
- if M_j is loaded "above" the sequence A_iA_{i-h}, during the other turn, d_j is increased by $2(h+1)\delta$.

Number

Note that in order to increase d_j by at least $2(h+1)\delta$ for all $j \in \{1, \dots, m\}$, it is necessary to add two waves, one on each turn, or one at each extremity of a turn.

More generally, for m arbitrarily large, $2n$ waves of size h are required to increase d_j by $n(2(h+1))\delta$ for all $j \in \{1, \dots, m\}$, adding $4n(h+1)\delta$ to the total travel time.

Placement

Let $i_1 < i_2 \in \{0, \dots, m\}$. In order to increase each d_j value by at least $2(h+1)\delta$ with only the sequences $A_{i_1}A_{i_1-h}$ and $A_{i_2}A_{i_2-h}$, the indices i_1 and i_2 must verify:

$$i_2 - i_1 \leq h + 3 \tag{4.8}$$

Otherwise, d_{i_1+3} is unchanged (see Figure 4.8): if A_{i_1} takes place on the first turn, then by Proposition 4.4, A_{i_1+2} takes place on the first turn, after the first wave, and A_{i_1+3} takes place on the second turn before the second wave. Thus the travel between the loading and unloading of M_{i_1+3} doesn't contain any of the two waves and is not impacted.

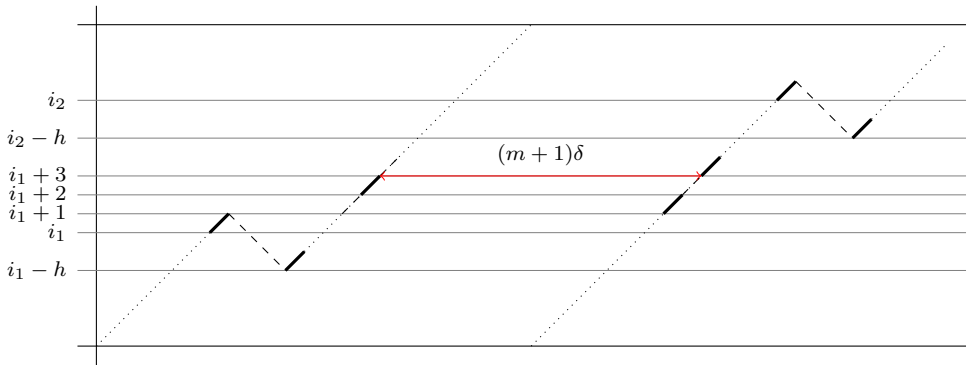


Figure 4.8: Placement.

More generally, to increase every d_j value by at least $n(2(h+1))\delta$ using the sequences $A_{i_1}A_{i_1-h}, \dots, A_{i_{2n}}A_{i_{2n}-h}$ with $i_1 < i_2 \cdots < i_{2n}$, Equation (4.8) must be verified by each pair (i_k, i_{k+1}) for $1 \leq k \leq 2n-1$.

Notation

Here, we introduce notations to describe in a compact fashion cycles formed with waves of uniform size h.

For $h = 1$, the number and position of the waves is sufficient to specify the full activity sequence of the cycle, as by Proposition 4.4 no consecutive activities

may take place on the same turn in increasing order, so the only consecutive activities taking place on the same turn are the wave sequences, all the others are alternated. We denote $v(i_1, i_2, \ldots, i_n)$, the cycle formed with the waves $A_{i_1}A_{i_1-1}$, $A_{i_2}A_{i_2-1} \ldots A_{i_n}A_{i_n-1}$. To build the full expression of the cycle, on can proceed as follows: starting with two empty turns, take the activities in increasing order starting with A_0. Each activity A_i is inserted sequentially into one of the two turns following these rules: if $i \in \{i_1, \ldots i_n\}$ then A_i is inserted just before A_{i-1}, on the same turn, otherwise A_i is appended to the other turn. For example, on a 10-machine cell, $v(3,7) = A_0A_3A_2A_5A_8A_{10}A_1A_4A_7A_6A_9$. This cycle, represented in Figure 4.12, is studied with more detail in Section 4.5.

For $h > 1$, the specification of the waves is not sufficient to deduce the complete sequence: we denote by $V_h(i_1, i_2, \ldots, i_n)$ the set of cycles containing waves $A_{i_1}A_{i_1-h}, A_{i_2}A_{i_2-h}, \ldots, A_{i_n}A_{i_n-h}$.

4.4.3 ... And still waters

Graphically, the unfolding in the cell of the cycles considered in this chapter is made of "turbulence zones", where the robot moves are altered by the waves' backward sequences, and "still waters", where the robot behaves as when executing π_{oe}, moving forward and serving one machine in two.

Such a decomposition is represented on Figure 4.9, with two *still water* areas, surrounding one *turbulence zone*. For machines located in the *still water* areas, the distance traveled by the robot between the loading and unloading depends solely on the size and number of waves in the *turbulence zone*. More precisely, for machines in the lower still area loaded during the *first* turn, and machines in the upper still area loaded during the *second* turn, this value (d_a, represented in blue) is identical, and determined by the waves carried on the first turn. Likewise, for machines in the lower still area loaded during the *second* turn and machines in the upper still area loaded during the *first* turn, this value (d_b, represented in green) is identical, and determined by the waves carried on the second turn.

If both turns carry the same number and size of waves, then $d_a = d_b$, making the cycle structure in these still water areas similar to the one of π_{oe}. In this section, we exploit this similarity to give a lower bound on the cycle time, more precisely on the waiting time.

From now on, for any cycle c, we use the notation $W_c(p)$ to denote the long run average total waiting time of the robot for the cycle c, depending on p. Recall that the cycle time of c is the sum of its travel time and waiting time, thus $T_c(p) = \Delta_c(p) + W_c(p)$.

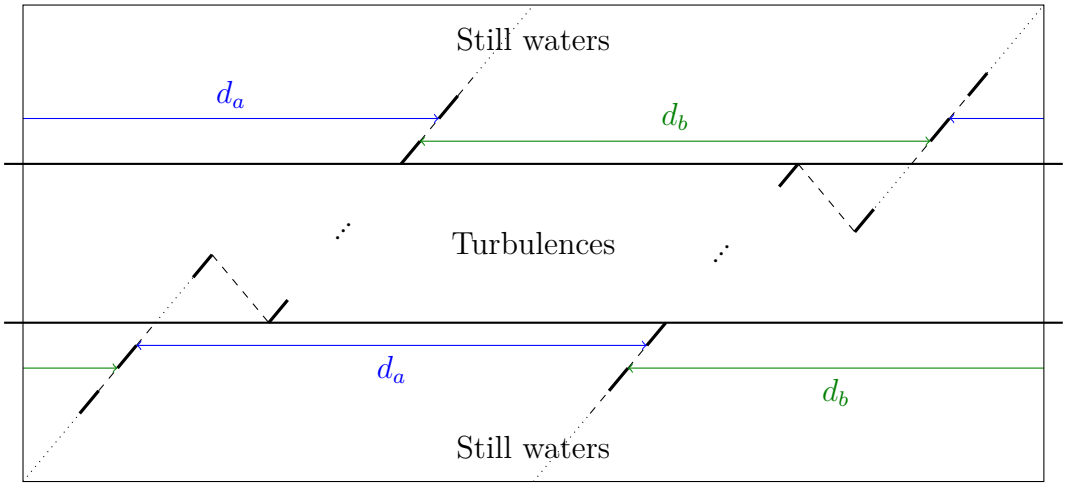


Figure 4.9: Decomposition of a cycle in still waters and turbulences

Definition 4.2. For any $n > 0,\ p \geq 0,\ d \geq 0$, we define:

$$\hat{W}(n,d,p) = \frac{2\alpha(n)-1}{\alpha(n)} \max(0, p-d), \text{ where } \alpha(n) = \left\lfloor \frac{n+1}{2} \right\rfloor$$

and for $n = 0$, $\hat{W}(n,d,p) = 0$.

In particular, $\hat{W}\big(m,(m+1)\delta,p\big) = W_{\pi_{oe}}(p)$. More generally, $\hat{W}(n,d,p)$ would be the waiting time of a cycle similar to π_{oe} on n machines, with $d_i = d$ for all $i \in \{0,\dots,m+1\}$. Note that for a given d and p, the function $\hat{W}(n,d,p)$ is increasing in n while for a given n and p, $\hat{W}(n,d,p)$ is decreasing in d.

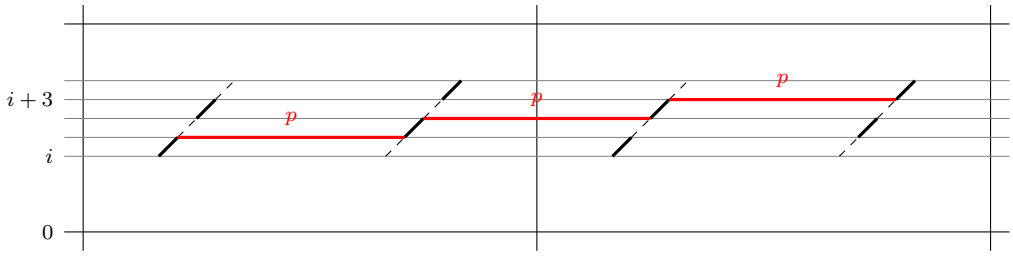
The following Proposition uses $\hat{W}$ to bound the cycle time of a cycle based on the part that is similar to the odd-even cycle. For convenience, we denote $\mathrm{SE}_{i,j}$ (respectively $\mathrm{SO}_{i,j}$) the ordered sequence of all activities A_l with l even (respectively odd), $i \leq l \leq j$. For instance, $\mathrm{SE}_{3,8} = A_4A_6A_8$.

Proposition 4.5. *Let π be a 1-cycle verifying Proposition 4.3.*

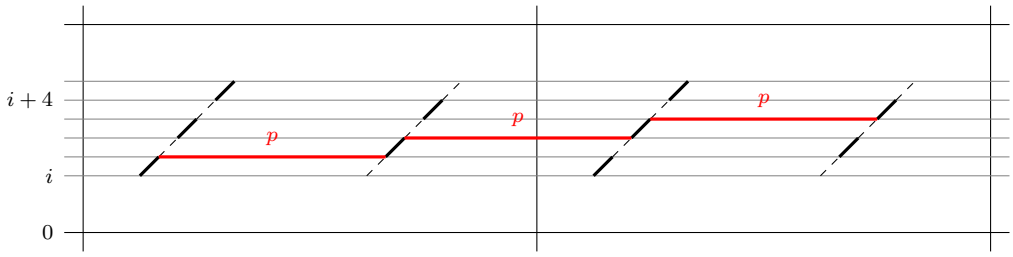
(i) If there exists i, l, d such that

- *π contains both sequences $\mathrm{SE}_{i,i+l}$ and $\mathrm{SO}_{i,i+l}$,*
- *$\forall j \in \{i+1,\dots i+l\},\ d_j = d$*

Then $W_\pi(p) \geq \hat{W}(l,d,p)$.



(a) l=3



(b) l=4

Figure 4.10: Proposition 4.5 (i): example for $l = 3$ and $l = 4$. Two iterations are represented. During each of the three intervals represented in red, at most d unit of travel time occur, and p time units must have gone by in order for the part to be available. Thus at least $\max(0, p - d)$ units of waiting time are necessary. On average, $\frac{3}{2}\max(0, p - d)$ units of waiting time by iteration are necessary.

(ii) More generally, if there exists d, i_1, i_2, l_1, l_2, *with* $i_2 > i_1 + l_1$, *such that*

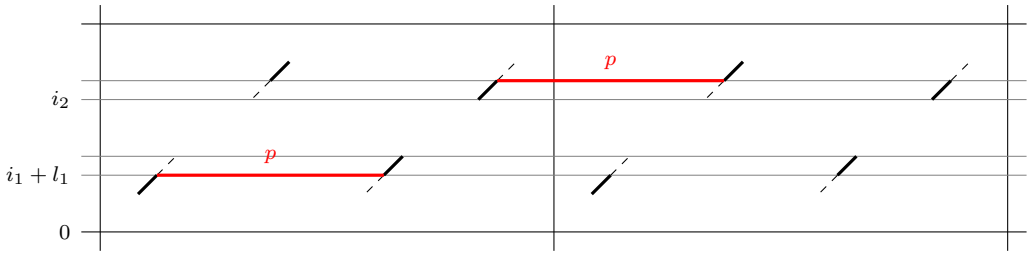
- π *contains both sequences* $\mathrm{SE}_{i_1,i_1+l_1}$ *and* $\mathrm{SO}_{i_1,i_1+l_1}$
- π *contains both sequences* $\mathrm{SE}_{i_2,i_2+l_2}$ *and* $\mathrm{SO}_{i_2,i_2+l_2}$
- $\forall j \in \{i_1 + 1, \dots i_1 + l_1\} \cup \{i_2 + 1, \dots i_2 + l_1\},\ d_j = d$

Then, $W_\pi(p) \geq \hat{W}(l_1 + l_2 - 1, d, p)$

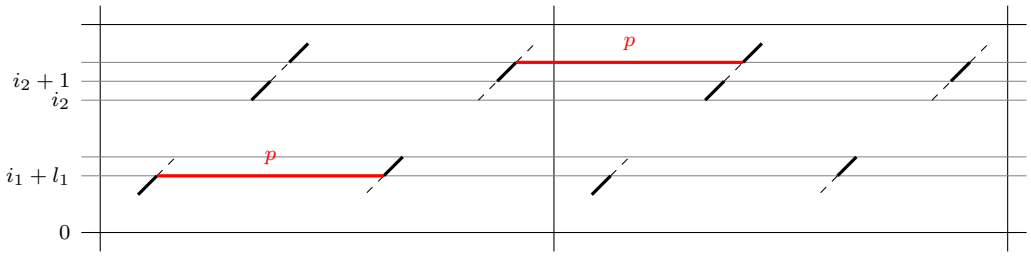
Proof.

(i) Similarly to the proof of π_{oe}'s cycle time, we consider $\alpha(l)$ iterations of the cycle, and follow the path of one part in the cell, from its loading on M_{i+1} (A_i) during the first iteration up to its unloading from M_{i+l} with A_{i+l} (if l odd), or from M_{i+l-1} (if l even). Figure 4.10 shows an example, respectively for $l = 3$ (Figure 4.10a) and $l = 4$ (Figure 4.10b). Between each consecutive loading and unloading of the same machine, the waiting time is at least $\max(0, p - d)$, leading to a total waiting time of at least $(2\alpha(l) - 1)\max(0, p - d)$ over $\alpha(l)$ iterations.

(ii) A_{i_2} and A_{i_2+1} are not performed during the same turn, so one of them is performed during the same turn as $A_{i_1+l_1}$. If it is A_{i_2}, the proof is similar to case (i)



(a) $A_{i_1+l_1}$ and A_{i_2} on the same turn



(b) $A_{i_1+l_1}$ and A_{i_2+1} on the same turn

Figure 4.11: Proposition 4.5 (ii)

with $l = l_1 + l_2$, following the path of one part from its loading on M_{i_1+1} to its unloading from $M_{i_1+l_1}$, then from the loading of M_{i_2+1} to its unloading from $M_{i_2+l_2}$ (see Figure 4.11a). If not, the proof is similar to case (i) with $l = l_1+l_2-1$, following the path of one part from its loading on M_{i_1+1} to its unloading from $M_{i_1+l_1}$, then from the loading of M_{i_2+2} to its unloading from $M_{i_2+l_2}$(see Figure 4.11b).

□

If the number/size of waves carried on each turn is different, the cycle might contain π_{oe}-like sequences with alternating d_i values (see Figure 4.9, with $d_a \neq d_b$). Proposition 4.5(i) can be extended to this case:

Proposition 4.6.

Let π be a 1-cycle verifying Proposition 4.3. If the following conditions are verified:

- *π contains both sequences* $\mathrm{SE}_{i,i+l}$ *and* $\mathrm{SO}_{i,i+l}$,
- $\forall j \in \{i+1, \dots i+l\}$ *with* $j - i = 1 \mod 2$, $d_j = d_a$
- $\forall j \in \{i+1, \dots i+l\}$ *with* $j - i = 0 \mod 2$, $d_j = d_b$

Then $W_\pi(p) \geq \max\left(0,\ p - d_a,\ p - d_b,\ p - d_a + \frac{\alpha(l)-1}{\alpha(l)}(p - d_b)\right)$.

Proof. The proof is similar to the proof of Proposition 4.5(i). We follow the path of one part in the cell between A_i and A_{i+l} (if l odd) or A_{i+l-1} (if l even), over $\alpha(l)$ iterations of the cycle. The part is processed on $\alpha(l)$ machines for which $d_i = d_a$ and $\alpha(l) - 1$ machines for which $d_i = d_b$. If $p \geq \max(d_a, d_b)$, this leads to a total waiting time of at least $\alpha(l)(p - d_a) + (\alpha(l) - 1)(p - d_b)$. □

4.5 Wavelets ($h = 1$)

The observations reported in Section 4.3 on the simulations performed for $9 \leq m \leq 14$ highlighted two types of cycles which outperform $\{\pi_{oe}, \pi_{dh}\}$, both containing only size-1 waves. The first type contains two waves, and outperforms $\{\pi_{oe}, \pi_{dh}\}$ for every tested value of m. The second type contains 3 waves and appeared to be interesting only for $m = 12$ and $m = 13$. In this section, we formalize and generalize these experimental results by defining a new family of cycles, containing only evenly spaced size-1 waves. We study the properties of this family and its dominance over the classical cycles set $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$. Section 4.5.1 presents a direct generalization of the 2-waves type for m machines, and a proof of its cycle time and dominance over $\{\pi_{oe}, \pi_{dh}\}$. In Section 4.5.2 this form is generalized to an arbitrary number of waves, either even or odd. In section Section 4.5.3, we compare these cycles.

4.5.1 Cycle π_{2w}

For any $m \geq 9$, let us consider the cycle $\pi_{2w} = v(3, 7)$. More precisely, π_{2w} is defined as follows:

For m even,

$$\pi_{2w} = A_0 A_3 A_2 A_5 A_8 \dots A_{2i} \dots A_m A_1 A_4 A_7 A_6 A_9 \dots A_{2i+1} \dots A_{m-1}$$

and for m odd,

$$\pi_{2w} = A_0 A_3 A_2 A_5 A_8 A_{10} \dots A_{2i} \dots A_{m-1} A_1 A_4 A_7 A_6 A_9 A_{11} \dots A_{2i+1} \dots A_m$$

Similarly to the odd-even cycle, this sequence consists of the robot circling the cell twice, loading one machine in two, but each of the two turns is altered by a backward sequence $A_i A_{i+1}$. Figure 4.12 shows an example on a 10-machine cell. We call first turn the part of the cycle from A_0 to A_m (respectively A_{m-1}) for m even (respectively odd), and second turn the rest of the cycle.

Compared to π_{oe}, each wave adds 4δ to the total travel time, but for all $i \in \{0, \dots, m+1\}$, d_i is increased by at least 4δ, hence we have

$$\Delta_{\pi_{2w}} = 2(m+1)\delta + 8\delta = 2(m+5)\delta$$

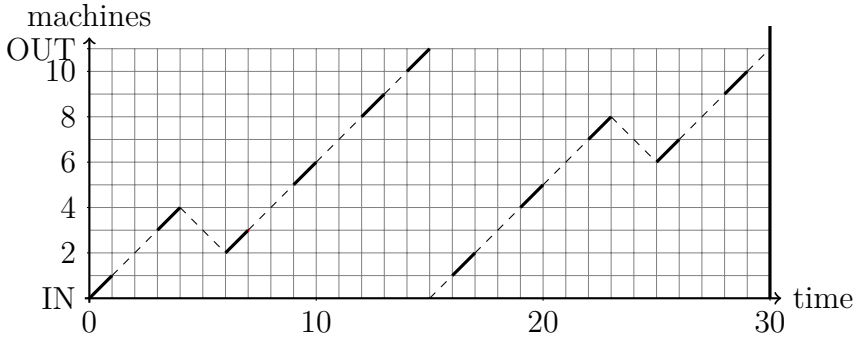


Figure 4.12: cycle π_{2w} on a 10-machine cell.

$$d_{min}(\pi_{2w}) = (m+5)\delta$$

The simulations described in Section 4.3 showed that for $10 \leq m \leq 14$, cycle π_{2w} appears to dominate $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ for some parameter values; by algebraically establishing its cycle time, we prove that this is indeed valid for any $m \geq 9$.

Proposition 4.7. *Cycle time of* π_{2w}
For $(m+1)\delta \leq p \leq (m+5)\delta + 4\left\lfloor \frac{m+7}{2} \right\rfloor \delta$, *the cycle time of* π_{2w} *is:*

$$T_{\pi_{2w}}(p) = 2(m+5)\delta + \hat{W}(m-8, (m+5)\delta, p)$$

In particular, this is true for p^*, *since* $p^* < (m+5)\delta + 4\left\lfloor \frac{m+7}{2} \right\rfloor \delta$
As the cycle time is increasing in p, for $p > (m+5)\delta + 4\left\lfloor \frac{m+7}{2} \right\rfloor \delta$*:*

$$T_{\pi_{2w}}(p) \geq 2(m+5)\delta + \hat{W}(m-8, (m+5)\delta, p)$$

Proof.

The total travel time of π_{2w} is $2(m+5)\delta$; we show that in the first case its waiting time is exactly $\hat{W}(m-8, (m+5)\delta, p)$.

- For $j = 1$ and $j \in \{9, \ldots, m\}$, $d_j(\pi_{2w}) = d_{min}(\pi_{2w}) = (m+5)\delta$, and π_{2w} contains the four sequences $\mathrm{SE}_{0,1}$; $\mathrm{SO}_{0,1}$; $\mathrm{SE}_{8,m}$ and $\mathrm{SO}_{8,m}$. As a consequence of Proposition 4.5:

$$W_{\pi_{2w}}(p) \geq \hat{W}(m-8, (m+5)\delta, p)$$

- To show that $W_{\pi_{2w}}(p) \leq \hat{W}(m-8, (m+5)\delta, p)$, we provide a repartition of waiting times between machines so that the sum of the waiting times is $\hat{W}(m-8, (m+5)\delta, p)$, and no additional waiting time is necessary.

  If $p \leq (m+5)\delta$, then no waiting time is necessary.

Let us consider $p > (m+5)\delta$. Between machines M_{10} and M_1 (in a cyclic sense), the cycle behaves in a similar way to π_{oe} on $(m-8)$ machines with $d_{min} = (m+5)\delta$. We place waiting times on these machines only, similarly to the waiting times of the corresponding odd-even cycle.

More precisely, waiting time are distributed this way for m even (w_i being the waiting time at machine M_i):

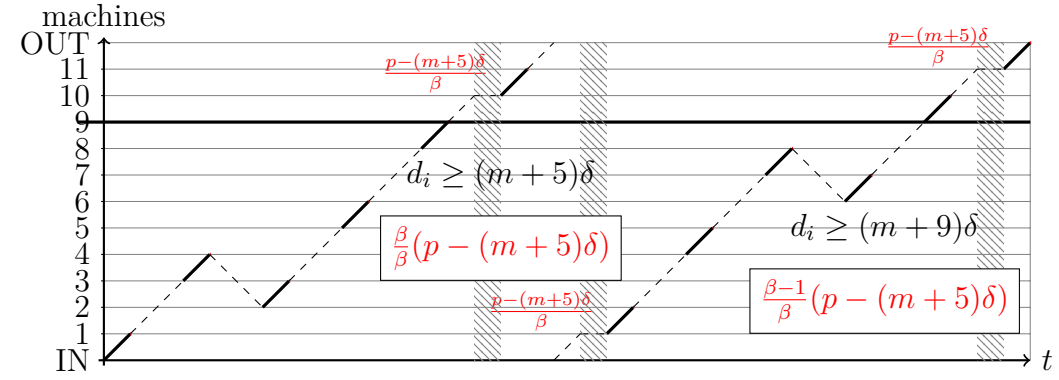
$$w_i = \begin{cases} 0 & \text{for } 1 \leq i \leq 9 \\ \frac{p-(m+5)\delta}{\alpha(m-8)} & \text{for } 10 \leq i \leq m \end{cases}$$

and for m odd:

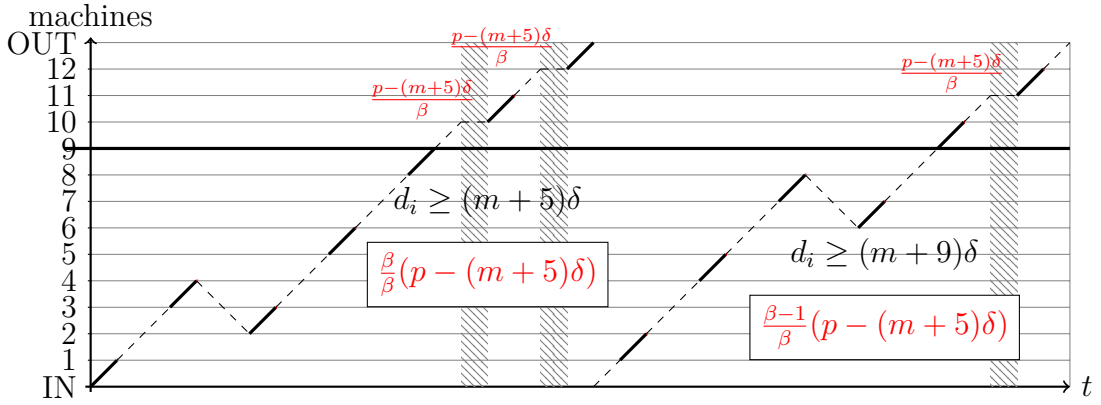
$$w_i = \begin{cases} 0 & \text{for } 2 \leq i \leq 9 \\ \frac{p-(m+5)\delta}{\alpha(m-8)} & \text{for } i = 1 \text{ and } 10 \leq i \leq m \end{cases}$$

Figures 4.13a and 4.13b show an example respectively for $m = 11$ and $m = 12$.

- The sum of all waiting times is:
$$\frac{2\alpha(m-8)-1}{\alpha(m-8)}(p-(m+5)\delta) = \hat{W}(m-8,(m+5)\delta,p).$$
- For $i = 1$ or $10 \leq i \leq m$: $d_i = (m+5)\delta$, and between the loading and unloading of machine M_i the robot waits $\alpha(m-8)$ times $\frac{1}{\alpha(m-8)}(p-(m+5)\delta)$, so no additional waiting time is necessary.
- For $2 \leq i \leq 9$, if M_i is loaded during the first turn: $d_i \geq (m+5)\delta$, and between the loading and unloading of machine M_i the robot waits $\alpha(m-8)$ times $\frac{1}{\alpha(m-8)}(p-(m+5)\delta)$, so no additional waiting time is necessary.
- For $2 \leq i \leq 9$, if M_i is loaded during the second turn: $d_i = (m+9)\delta$, and between the loading and unloading of machine M_i the robot waits $\alpha(m-8)-1$ times $\frac{1}{\alpha(m-8)}(p-(m+5)\delta)$.
  For this waiting time to be sufficient, the following must be satisfied:
$$(m+9)\delta + \left(1 - \frac{1}{\alpha(m-8)}\right)(p-(m+5)\delta) \geq p$$
  which is equivalent to:
$$p \leq (m+5)\delta + 4\alpha(m-8)\delta$$
$$p \leq (m+5)\delta + 4\left\lfloor \frac{m-7}{2} \right\rfloor \delta.$$



(a) 11 machines ($\beta = 2$)



(b) 12 machines ($\beta = 2$)

Figure 4.13: Waiting times during one iteration of π_{2w}. The value $\alpha(m-8)$ is denoted β.

In other words:

$$\begin{aligned} &p \leq 3m - 9 \text{ for } m \text{ odd} \\ &p \leq 3m - 11 \text{ for } m \text{ even} \end{aligned} \tag{4.9}$$

Note that for $m > 8$, condition (4.9) is verified in particular for any $p \leq p^*$.

Finally, for $(m+1)\delta \leq p \leq (m+5)\delta + 4\left\lfloor \frac{m+7}{2} \right\rfloor$,

$$T_{\pi_{2w}}(p) = 2(m+5)\delta + \hat{W}(m-8, (m+5)\delta, p). \tag{4.10}$$

□

In particular one can verify that $T_{\pi_{2w}}(p^*) < T_{\pi_{oe}}(p^*)$. Hence π_{2w} is not strictly dominated by $\{\pi_{oe}, \pi_{dh}\}$. In the next section, we introduce a more general family of cycle, including among others π_{oe} and π_{2w}, and detail their dominance relationships depending on the values of p.

4.5.2 Generalization: $(\pi_{nw})_n$

In this section, we define and study π_{nw}, a generalization of cycle π_{2w} formed by placing n waves alternately on each turn, spaced adequately to uniformly increase travel time between subsequent loading and unloading operations. Cycles π_{nw} for various value of n and of the number of machines m are compared in Section 4.5.3, basically showing that cycles π_{nw} with the highest possible values of n perform better.

Formally, for $n \geq 0$ and $m \geq 4n - 1$, we define:

$$\pi_{nw} = v(3, 7, \ldots, 4i - 1, \ldots, 4n - 1)$$

For example, $\pi_{4w} = v(3, 7, 11, 15)$. For 18 machines, its full expression is: $\pi_{4w} = A_0A_3A_2A_5A_8A_{11}A_{10}A_{13}A_{16}A_{18}A_1A_4A_7A_6A_9A_{12}A_{15}A_{14}A_{17}$. In particular, $\pi_{0w} = \pi_{oe}$.

A representation of π_{nw} for m even (respectively m odd) is shown on Figure 4.14 (respectively Figure 4.15). We call first turn the part of the cycle from A_0 until the robot reaches the output station and second turn the rest of the cycle. The total travel time of π_{nw} is:

$$\Delta_{\pi_{nw}} = 2(m + 2n + 1)\delta$$

In particular, for $m = 4n-1$ one has $\Delta_{\pi_{nw}} = 3(m+1)\delta$. Thus π_{nw} is dominated by π_{dh} for $m = 4n - 1$. Consequently, in the following we only consider $m \geq 4n$. As properties of π_{nw} depend on the parity of n, we consider separately both cases; for n even we give an expression of the cycle time; for n odd, a lower bound on the cycle time, and an expression of the cycle time for the special cases $m = 4n$ and $m = 4n + 1$.

For n even

For n even, π_{nw} satisfies:

$$d_{min}(\pi_{nw}) = (m + 2n + 1)\delta$$

Proposition 4.8 gives an expression for π_{nw} cycle time, generalizing Proposition 4.7.

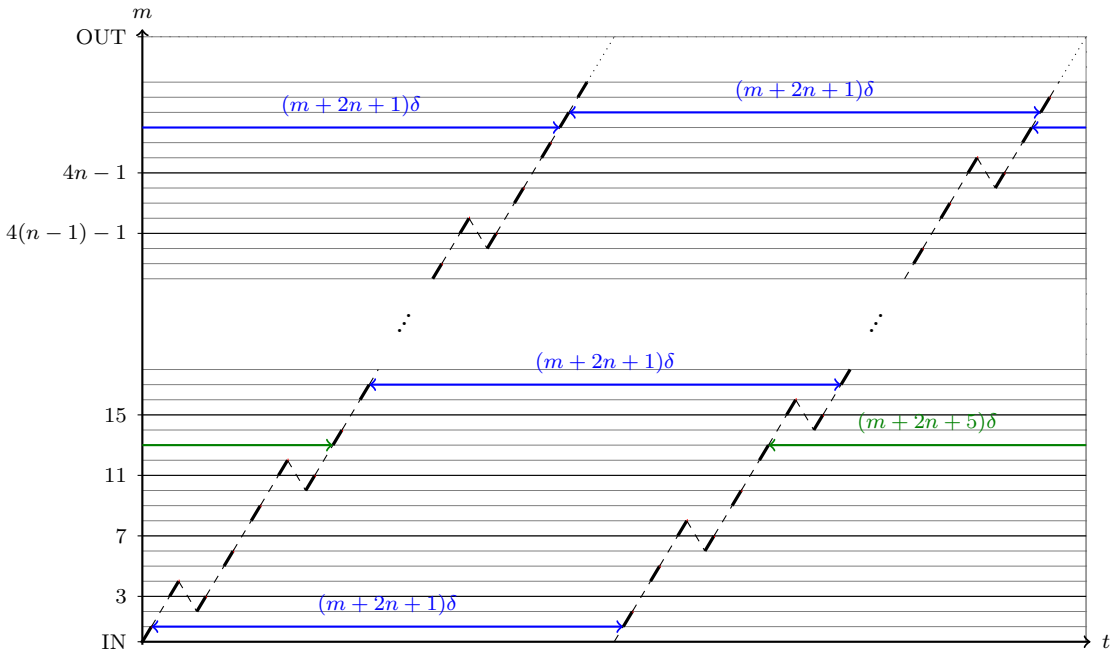


Figure 4.14: Representation of π_{nw}, with n even

Proposition 4.8. *Cycle time of π_{nw} with $m \geq 4n$ and n even*

For $(m+1)\delta \leq p \leq (m+2n+1)\delta + 4\left\lfloor \frac{m+4n-1}{2} \right\rfloor \delta$, the cycle time of π_{nw} is:

$$T_{\pi_{nw}}(p) = 2(m+2n+1)\delta + \hat{W}\Big(\max(1, m-4n), (m+2n+1)\delta, p\Big) \tag{4.11}$$

As the cycle time is increasing in p, for $p > (m+2n+1)\delta + 4\left\lfloor \frac{m+4n-1}{2} \right\rfloor \delta$:

$$T_{\pi_{nw}}(p) \geq 2(m+2n+1)\delta + \hat{W}\Big(m-4n, (m+2n+1)\delta, p\Big) \tag{4.12}$$

Proof. For $p \leq d_{min}(\pi_{nw}) = (m+2n+1)\delta$, there is no waiting time and the cycle time is $T_{\pi_{nw}}(p) = \Delta_{\pi_{nw}} = 2(m+2n+1)\delta$.

For $p > (m+2n+1)\delta$, the proof is similar to the one of Proposition 4.7. We use Proposition 4.5 to bound the waiting time, then we provide an adequate waiting time vector to show equality.

For $m = 4n$, by Proposition 4.5(*i*) with $i = 0$, $l = 1$, $d = (m+2n+1)\delta$: $W_{\pi_{nw}}(p) \geq \hat{W}(1, (m+2n+1)\delta, p)$.

For $m \geq 4n+1$, by Proposition 4.5(*ii*) with $i_1 = 0$, $i_2 = 4n$, $l_1 = 1$, $l_2 = m-4n$, $d = (m+2n+1)\delta$: $W_{\pi_{nw}}(p) \geq \hat{W}(m-4n, (m+2n+1)\delta, p)$.

To prove the equality, we consider the following waiting time vectors:

$$w_i = \begin{cases} 0 & \text{for } 2 \leq i \leq 4n \\ p - (m + 2n + 1)\delta & \text{for } i = 1 \end{cases} \qquad \text{for } m = 4n$$

$$w_i = \begin{cases} 0 & \text{for } 1 \leq i \leq 4n + 1 \\ \frac{p-(m+2n+1)\delta}{\alpha(m-4n)} & \text{for } 4n + 2 \leq i \leq m \end{cases} \qquad \text{for } m \geq 4n + 2, m \text{ even}$$

$$w_i = \begin{cases} 0 & \text{for } 2 \leq i \leq 4n + 1 \\ \frac{p-(m+2n+1)\delta}{\alpha(m-4n)} & \text{for } i = 1 \text{ or } 4n + 2 \leq i \leq m \end{cases} \qquad \text{for } m \geq 4n + 1, m \text{ odd}$$

In each case, the sum of all waiting times is exactly:

$$\hat{W}\Big(\max(1, m - 4n), (m + 2n + 1)\delta, p\Big)$$

.

Noticing that:

- for machines M_i, $2 \leq i \leq 4n + 1$, loaded during the first turn: $d_i = (m + 2n + 1)\delta$,
- for machines M_i, $2 \leq i \leq 4n + 1$, loaded during the second turn: $d_i = (m + 2n + 5)\delta$,

And using the same arguments as in Proposition 4.7, one can verify that these waiting times are sufficient for
$p \leq (m + 2n + 1)\delta + 4\left\lfloor \frac{m-4n+1}{2} \right\rfloor \delta$.

□

As $(m + 2n + 1)\delta + 4\left\lfloor \frac{m-4n+1}{2} \right\rfloor \delta \leq (3m - 1)\delta$, Proposition 4.8 does not give the exact cycle time over the entire considered interval $](m + 1)\delta, (3m - 1)\delta[$. However, once the cycle time reaches $3(m + 1)\delta$, the corresponding cycle is dominated by π_{dh} and need not be considered anymore. Let us define the value of p for which the lower bound given by Proposition 4.8 reaches $3(m + 1)\delta$:

Definition 4.3. For n even, we call p_n^* the value of p such that

$$2(m + 2n + 1)\delta + \hat{W}\Big(\max(1, m - 4n), (m + 2n + 1)\delta, p_n^*\Big) = T_{\pi_{dh}}(p_n^*) = 3(m + 1)\delta.$$

For $m \geq 4n + 1$, its value is:

$$p_n^* = (m + 2n + 1)\delta + \frac{\alpha(m - 4n)}{2\alpha(m - 4n) - 1}(m + 1 - 4n)\delta$$

The following corollary states that, for $m \geq 4n+1$, this value p_n^* falls within the interval for which the exact cycle time of π_{nw} is given by Equation (4.11). This means that the exact cycle time is really known for the values of p for which it is worth knowing; it will be useful to compare π_{nw} to other cycles.

Corollary 4.1.
For $m \geq 4n+1$ and n even, the cycle time of π_{nw} satisfies:
For $(m+1)\delta < p \leq p_n^*$,

$$\begin{aligned} T_{\pi_{nw}}(p) &= 2(m+2n+1)\delta + \hat{W}\Big(\max(1, m-4n), (m+2n+1)\delta, p\Big) \\ &\leq 3(m+1)\delta \end{aligned}$$

For $p \geq p_n^*$, $T_{\pi_{nw}}(p) \geq 3(m+1)\delta$.

.

Proof. By definition of p_n^*, we know that for $p \geq p_n^*$, $T_{\pi_{nw}}(p) \geq 3(m+1)\delta$. To prove that for $(m+1)\delta < p \leq p_n^*$, the cycle time of π_{nw} is given by Equation (4.11), we show that $p_n^* \leq (m+2n+1)\delta + 4\left\lfloor \frac{m+4n-1}{2} \right\rfloor \delta$. Noticing that:

$$\frac{\alpha(m-4n)}{2\alpha(m-4n)-1}(m+1-4n)\delta \leq (m+1-4n)\delta \leq 4\left\lfloor \frac{m+4n-1}{2} \right\rfloor \delta \qquad (4.13)$$

we get:

$$\begin{aligned} p_n^* &= (m+2n+1)\delta + \frac{\alpha(m-4n)}{2\alpha(m-4n)-1}(m+1-4n)\delta \\ &\leq (m+2n+1)\delta + 4\left\lfloor \frac{m+4n-1}{2} \right\rfloor \delta \end{aligned}$$

hence the cycle time of π_{nw} over $](m+1)\delta, p_n^*[$ is given by Equation (4.11). □

For n odd

Consider π_{nw} with n odd. In this case, one more wave is carried on the first turn ($\frac{n-1}{2}+1$ waves) than on the second one ($\frac{n-1}{2}$ waves). As a consequence, machines in the upper still waters area whose processing takes place between the first and second turn have a smaller d_i value than those active between the second and the first turn (see Figure 4.15). If m is large enough so that the cell contains both types of machine ($m \geq 4n+2$), then we have $d_{min}(\pi_{nw}) = (m+2(n-1)+1)\delta$, which is the same d_{min} value as $\pi_{(n-1)w}$, but with more total travel time. However, if m is small enough that there is no machine to load on the first turn after the last wave ($m \leq 4n+1$), then $d_{min}(\pi) = m+2(n+1)+1$.

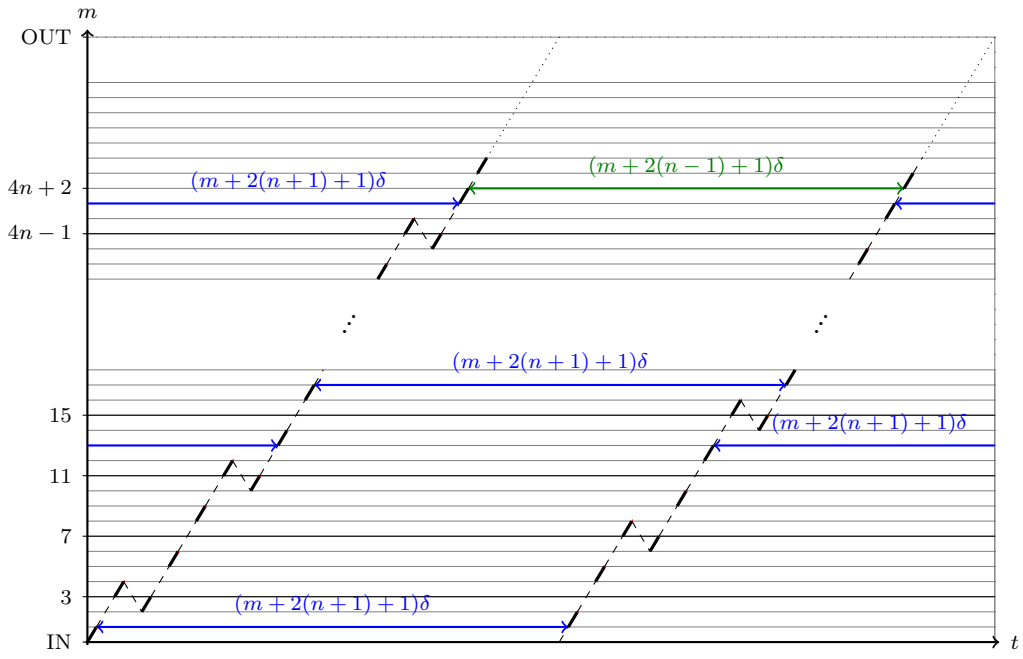


Figure 4.15: Representation of π_{nw}, with n odd.

Remember from simulations presented in Section 4.3 that contrary to π_{2w} which appears to consistently dominate $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ for some p for $9 \leq m \leq 14$, the cycle π_{3w} appears to be interesting for $m = 12$ and $m = 13$ but not $m = 14$, suggesting that π_{nw} with n odd may be dominated when $m \geq 4n + 2$. The following proposition gives a lower bound of the cycle time for $m \geq 4n + 2$, which will be used in Section 4.5.3 to show that it is indeed the case. For the case where $m \leq 4n+1$, an exact expression of the cycle time is given later by Proposition 4.10

Proposition 4.9. *Lower bound for the cycle time of π_{nw} with n odd and $m \geq 4n+2$*
For $m \geq 4n + 2$ and n odd, the cycle time of π_{nw} verifies:

$$\begin{aligned} T_{\pi_{nw}}(p) \geq & 2(m + 2n + 1)\delta \\ & + \max\Big(0,\ p - (m + 2n - 1)\delta, \\ & p - (m + 2n - 1)\delta + \frac{\alpha(m - 4n - 1) - 1}{\alpha(m - 4n - 1)}\Big(p - (m + 2n + 3)\delta\Big)\Big) \end{aligned} \tag{4.14}$$

Proof. For $m \geq 4n + 2$, we have $d_{min}(\pi_{nw}) = (m + 2(n - 1) + 1)\delta$. More precisely, for $i > 4n + 1$ and i even, $d_i = (m + 2(n - 1) + 1)$, while for $i > 4n + 1$ and i odd, $d_i = (m + 2(n + 1) + 1)$ (see Figure 4.15).

Using Proposition 4.6 with $i = 4n+1$, $l = (m-4n-1)$, $d_a = (m+2n-1)\delta$, $d_b = (m+2n+3)\delta$, we get:

$$W_{\pi_{nw}}(p) \geq \max\Big(0,\ p-(m+2n-1)\delta,\\ p-(m+2n-1)\delta+\frac{\alpha(m-4n-1)-1}{\alpha(m-4n-1)}\Big(p-(m+2n+3)\delta\Big)\Big)$$

□

Using this bound on the cycle time, the following corollary precise a sufficient condition on p under which π_{nw} is dominated by π_{dh}:

Corollary 4.2.
For $m \geq 4n+2$ and n odd, if

$$p \geq m+2n+1+\frac{\alpha(m-4n-1)}{2\alpha(m-4n-1)-1}(m+1-4n)-\frac{2}{2\alpha(m-4n-1)-1}$$

then $T_{\pi_{nw}}(p) \geq 3(m+1)\delta$ (and therefore π_{nw} is dominated by π_{dh}).

Proof. Direct consequence of Equation (4.14). □

Recall that for $m \leq 4n$, π_{nw} is dominated as its total travel time is greater than $3(m+1)\delta$. For $m = 4n$ and $m = 4n+1$, the following proposition gives an expression of the cycle time.

Proposition 4.10. *Cycle time of π_{nw} with n odd and $m \leq 4n+1$*
For $m = 4n$ and for $m = 4n+1$, for $(m+1)\delta \leq p \leq (3m-1)\delta$, the cycle time of π_{nw} is given by[1]*:*

$$T_{\pi_{nw}}(p) = 2(m+2n+1)\delta+\max\Big(0,\frac{2n}{n+1}(p-(m+2n+3)\delta)\Big) \tag{4.15}$$

Proof. For $m = 4n$ and $m = 4n+1$, as no machine is loaded after performing $A_{4n-1}A_{4n-2}$ during the same turn, $d_{min}(\pi_{nw}) = (m+2(n+1)+1)\delta$. More precisely, $d_i = (3m-2)\delta$ (if $m = 4n$) or $d_i = (3m-3)\delta$ (if $m = 4n+1$) for all $i \in \{3,7,\ldots,4n-1\}$, and $d_i = (m+2(n+1)+1)\delta$ for all others machines M_i.

The processing of a same part on the n machines $M_2, M_4, \ldots, M_{4i-1}, \ldots M_{4n-1}$ requires $\frac{n+1}{2}$ iterations of the cycle, thus $T_{\pi_{nw}} \geq 2(m+2n+1)\delta+\frac{2n}{n+1}(p-(m+2n+3)\delta)$. An example for $m = 13$ and $n = 3$ is shown on Figure 4.16. The equality

[1]Note that in this case $\frac{2n}{n+1} = \frac{2\alpha(n)-1}{\alpha(n)}$.

can be proven by considering the following waiting time vector (represented by the hatched aeras on Figure 4.16):

$$w_i = \begin{cases} 0 & \text{for } i \neq 2 \mod 4, i \in \{1, \ldots, m\} \\ \frac{2(p-(m+2n+3)\delta)}{n+1} & \text{for } i = 2 \mod 4, i \in \{1, \ldots, m\} \end{cases}$$

The sum of all waiting times is $\frac{2n}{n+1}(p-(m+2n+3)\delta)$, and during each processing, waiting is performed $\frac{n+1}{2}$ times, covering all the necessary waiting time. □

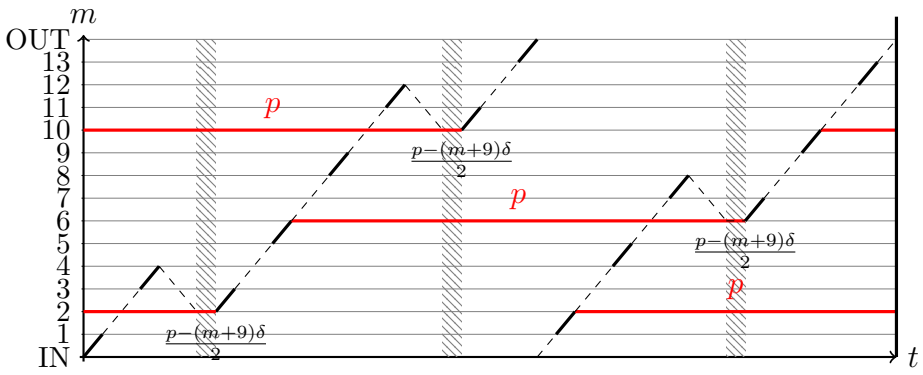


Figure 4.16: Cycle time of π_{nw} for n odd and $m = 4n + 1$: example for $m = 13$ and $n = 3$.

The following corollary is a direct consequence of Equation (4.15):

Corollary 4.3. *For $m = 4n$ and n odd, $T_{\pi_{nw}}(p) \leq 3(m+1)\delta$ if and only if $p \leq \frac{3}{2}m + \frac{7}{2} + \frac{2}{m}$. For $m = 4n+1$ and n odd, $T_{\pi_{nw}}(p) \leq 3(m+1)\delta$ if and only if $p \leq \frac{3}{2}m + \frac{7}{2} + \frac{4}{m-1}$.*

4.5.3 Dominance relations within (π_{nw}) family

In this section, we draw the consequences of Propositions 4.8 to 4.10 in terms of dominance between members of $(\pi_{nw})_{1 \leq n \leq \lfloor \frac{m}{4} \rfloor}$, for any $m > 8$. We show that at most 2 of them (depending on the value of m), along with $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$, are sufficient to dominate the entire family. As in the rest of the section, we only consider $(m+1)\delta < p < (3m-1)\delta$, as other cases are already solved (see Section 4.2).

π_{nw} with n even and $m = 4n$

For n even, $m = 4n$, one has $\Delta_{\pi_{nw}} = 2(\frac{3}{2}m + 1)\delta$ and $d_{min}(\pi_{nw}) = (\frac{3}{2}m + 1)\delta$. Thus, π_{nw} does not verify Equation (4.5) and is therefore dominated by $\{\pi_{oe}, \pi_{dh}\}$.

π_{nw} with n odd and $m \geq 4n+2$

We consider $n \geq 0$ odd and $m > 4n+2$. A lower bound on π_{nw} is given by Equation (4.14).

- For $(m+1)\delta < p \leq (m+2n-1)\delta$, one has $T_{\pi_{nw}}(p) \geq 2(m+2n+1)\delta$. Thus,

$$\begin{aligned} T_{\pi_{nw}}(p) - T_{\pi_{oe}}(p) &\geq 4n\delta - \frac{2\alpha-1}{\alpha}(p-(m+1)\delta) \\ &\geq 4n\delta - \frac{2\alpha-1}{\alpha}2n\delta \\ &\geq 0 \end{aligned}$$

  and hence π_{nw} is dominated by π_{oe}.

- For $(m+2n-1)\delta < p < (m+2n+3)\delta$,

$$T_{\pi_{nw}}(p) \geq 2(m+2n+1)\delta + p - (m+2n-1)\delta$$

  Thus,

$$T_{\pi_{nw}}(p) - T_{\pi_{oe}}(p) \geq 2n + \frac{1-\alpha}{\alpha}p + \frac{\alpha-1}{\alpha}m\delta + \frac{3\alpha-1}{\alpha}\delta$$

  As $p < (m+2n+3)$ and $\alpha > 1$, we get

$$\begin{aligned} T_{\pi_{nw}}(p) - T_{\pi_{oe}}(p) &\geq \frac{2n+2}{\alpha} \\ &\geq 0 \end{aligned}$$

  Here again, π_{nw} is dominated by π_{oe}.

- For $p \geq (m+2n+3)\delta$,

$$T_{\pi_{nw}}(p) \geq 2(m+2n+1)\delta + p - (m+2n-1) + \frac{\alpha(m-4n-1)-1}{\alpha(m-4n-1)}\Big(p-(m+2n+3)\delta\Big)$$

  We can verify that

$$p^* \geq m+2n+1 + \frac{\alpha(m-4n-1)}{2\alpha(m-4n-1)-1}(m+1-4n) - \frac{2}{2\alpha(m-4n-1)-1}$$

  Therefore by Corollary 4.2, $T_{\pi_{nw}}(p^*) \geq 3(m+1)\delta = T_{\pi_{oe}}(p^*)$.

  $T_{\pi_{nw}} > T_{\pi_{oe}}$ for both $p = m+2n+3$ and $p = p^*$, and the cycles $T_{\pi_{nw}}$ and $T_{\pi_{oe}}$ are both linear in p over $[m+2n+3, p^*]$. Thus, $T_{\pi_{nw}} \geq T_{\pi_{oe}}$ over the whole interval $[m+2n+3, p^*]$.

  Over $[p^*, (3m-1)\delta]$, one has $T_{\pi_{nw}} \geq 3(m+1)\delta \geq T_{\pi_{dh}}$.

Thus, π_{nw} is dominated by $\{\pi_{oe}, \pi_{dh}\}$.

π_{nw} and $\pi_{(n+2)w}$ with n even

We consider $n \geq 0$ even and $m > 4(n+2)$ (so that π_{nw} and $\pi_{(n+2)w}$ are both defined and not necessarily dominated by $\{\pi_{oe}, \pi_{dh}\}$) and compare the cycle times of π_{nw} and $\pi_{(n+2)w}$.

- For $p \leq (m+2n+1)\delta$, there is no waiting time for any of the two cycles: $T_{\pi_{nw}}(p) = 2(m+2n+1)\delta$ and $T_{\pi_{(n+2)w}}(p) = 2(m+2n+5)\delta$, so $T_{\pi_{nw}}(p) \leq T_{\pi_{(n+2)w}}(p)$ and π_{nw} dominates $\pi_{(n+2)w}$.
- For $(m+2n+1)\delta < p \leq (m+2n+5)\delta$, waiting is required only for cycle π_{nw}. The waiting time of π_{nw} is $\frac{2\alpha(m-4n)-1}{\alpha(m-4n)}(p-(m+2n+1)\delta)$. As $p \leq (m+2n+5)$ this waiting time is smaller than 8δ, which is the amount of extra travel time required by $\pi_{(n+2)w}$; thus, π_{nw} dominates $\pi_{(n+2)w}$.
- For $(m+2n+5)\delta < p \leq p^*_n$, waiting is required for both cycles. Noticing that, for any number β, one has $\frac{2\beta-1}{\beta} = 2 - \frac{1}{\beta}$, and that $\alpha(m-4n) = \alpha(m) - 2n$ (similarly, $\alpha(m-4(n+2)) = \alpha(m) - 2(n+2)$), the cycle times difference simplifies to:

$$T_{\pi_{nw}}(p) - T_{\pi_{(n+2)w}}(p) = \frac{4}{(\alpha(m) - 2(n+2))(\alpha(m) - 2n)}(p - (m + \alpha(m) + 1)\delta)$$

Note that as $m \geq 4(n+2)$, one has $(m+\alpha(m)+1)\delta \geq (m+2n+5)\delta$, and that $p^*_{n+2} \geq p^*_n$. Thus, we can split the values of p in 4 domains according to the relative performance of the cycles π_{nw}, $\pi_{(n+2)w}$ and π_{dh}:

- for $(m+1)\delta < p \leq (m+\alpha(m)+1)\delta$ we have $T_{\pi_{nw}} \leq T_{\pi_{(n+2)w}} < T_{\pi_{dh}}$, hence π_{nw} dominates $\pi_{(n+2)w}$ (and both dominate π_{dh}),
- for $(m+\alpha(m)+1)\delta < p < p^*_n$, we have $T_{\pi_{(n+2)w}} < T_{\pi_{nw}} < T_{\pi_{dh}}$, hence $\pi_{(n+2)w}$ dominates $\pi_{(n)w}$ (and both dominate π_{dh}).
- for $p^*_n \leq p < p^*_{n+2}$, by Corollary 4.1, we have $T_{\pi_{(n+2)w}} < T_{\pi_{dh}} \leq T_{\pi_{nw}}$, hence $\pi_{(n+2)w}$ dominates $\pi_{(n)w}$ (and π_{dh})
- for $p \geq p^*_{n+2}$, both π_{nw} and $\pi_{(n+2)w}$ are dominated by $T_{\pi_{dh}}$

Note that the threshold value of p from which $\pi_{(n+2)w}$ starts outperforming π_{nw} does not depend on n, and that p^*_n is increasing in n. So, for a given m, we get that among all possible cycles π_{nw} with n even, π_{oe} (π_{0w}) is the best one for $(m+1)\delta < p < (m+\alpha(m)+1)\delta$, and $\pi_{\lfloor \frac{m-1}{4} \rfloor w}$ (highest possible value of n) is the best one for $(m+\alpha(m)+1)\delta < p \leq p^*_{\lfloor \frac{m-1}{4} \rfloor}$.

π_{nw} and $\pi_{(n-1)w}$ with n odd and $m = 4n$

Consider n odd and $m = 4n$. By Proposition 4.10, for $p \leq (\frac{3}{2}m + 3)\delta$, $T_{\pi_{nw}} = (3m + 2)\delta$. In particular, $p^* \leq (\frac{3}{2}m + 3)\delta$ thus

$$T_{\pi_{nw}}(p^*) = (3m + 2)\delta < 3(m + 1)\delta = T_{\pi_{oe}}(p^*) = T_{\pi_{dh}}(p^*)$$

so contrary to the case where $m \geq 4n + 1$, the cycle π_{nw} is not dominated by $\{\pi_{oe}, \pi_{dh}\}$. Let us now compare π_{nw} to $\pi_{(n-1)w}$; as $m = 4n$, the cycle time of the latter, given by Proposition 4.8, simplifies to

$$T_{\pi_{(n-1)w}}(p) = (3m - 2)\delta + \max\left(0, \frac{3}{2}\left(p - \left(\frac{3}{2}m - 1\right)\delta\right)\right)$$

Thus, for $p \leq (\frac{3}{2}m + \frac{5}{3})\delta$, one has $T_{\pi_{(n-1)w}}(p) \leq T_{\pi_{nw}}(p)$, while for $(\frac{3}{2}m + \frac{5}{3})\delta \leq p \leq \frac{3}{2}m + 3)\delta$, one has $T_{\pi_{nw}}(p) \leq T_{\pi_{(n-1)w}}(p)$.

For $p \geq (\frac{3}{2}m + 3)\delta$, by Corollary 4.1 the cycle $\pi_{(n-1)w}$ is dominated by π_{dh}, so we do not consider it.

π_{nw} and $\pi_{(n-1)w}$ with n odd and $m = 4n + 1$

Consider n odd and $m = 4n + 1$. For $p \leq (\frac{3}{2}m + \frac{5}{2})\delta$, by Proposition 4.10, $T_{\pi_{nw}} = (3m + 1)\delta$. In particular,

$$T_{\pi_{nw}}(p^*) = (3m + 1)\delta < 3(m + 1)\delta = T_{\pi_{oe}}(p^*) = T_{\pi_{dh}}(p^*)$$

Let us compare π_{nw} to $\pi_{(n-1)w}$. As $m = 4n + 1$, the cycle time of the latter, given by Proposition 4.8, simplifies to

$$T_{\pi_{(n-1)w}}(p) = 3(m - 1)\delta + \max\left(0, \frac{5}{3}\left(p - \left(\frac{3}{2}(m - 1)\right)\delta\right)\right)$$

Thus, for $p \leq \frac{3}{2}(m+1)\delta$, one has $T_{\pi_{(n-1)w}}(p) \leq T_{\pi_{nw}}(p)$, while for $\frac{3}{2}(m+1)\delta \leq p \leq (\frac{3}{2}m + \frac{5}{2})\delta$, $T_{\pi_{nw}}(p) \leq T_{\pi_{(n-1)w}}(p)$. Note that $\frac{3}{2}(m + 1)\delta < (m + \alpha + 1)\delta$.

For $p \geq (\frac{3}{2}m + \frac{5}{2})\delta$, by Corollary 4.1, $\pi_{(n-1)w}$ is dominated by π_{dh}, so we do not consider it.

Putting everything together

For n odd and $m \geq 4n + 2$, or n even and $m = 4n$, the cycle π_{nw} is dominated by $\{\pi_{oe}, \pi_{dh}\}$. In other cases, π_{nw} outperforms both π_{oe} and π_{dh} for values of p close to p^*. Using results of this section as well as Corollary 4.1 and Corollary 4.2, we can deduce the best 1-cycle *among* $\{\pi_{id}, \pi_{dh}, \pi_{oe}\} \cup (\pi_{nw})_n$ for any instance[2]. For any given m, only 4 or 5 cycles are enough to cover all possible values of p; these results are summed up by Proposition 4.11.

[2]But not among *all* 1-cycles (yet)! (For spoilers, jump to Conjecture 4.1.)

Proposition 4.11. *For $m > 8$, the best 1-cycles within $\{\pi_{id}, \pi_{dh}, \pi_{oe}\} \cup (\pi_{nw})_{n \leq \frac{m}{4}}$ for $(m+1)\delta \leq p \leq (3m-1)\delta$ are given by the following:*

- *If $m = 4 \mod 8$ (matches the case n odd and $m = 4n$)*

$$\begin{array}{rcll} (m+1)\delta \leq & p & \leq (\frac{3}{2}m+1)\delta & \pi_{oe} \\ (\frac{3}{2}m+1)\delta \leq & p & \leq (\frac{3}{2}m+\frac{5}{3})\delta & \pi_{(\frac{m}{4}-1)w} \\ (\frac{3}{2}m+\frac{5}{3})\delta \leq & p & \leq (\frac{3}{2}m+\frac{7}{2}+\frac{2}{m})\delta & \pi_{(\frac{m}{4})w} \\ (\frac{3}{2}m+\frac{7}{2}+\frac{2}{m})\delta \leq & p & \leq (3m-1)\delta & \pi_{dh} \quad \textit{(by Corollary 4.3)} \end{array}$$

- *If $m = 5 \mod 8$ (matches the case n odd and $m = 4n+1$)*

$$\begin{array}{rcll} (m+1)\delta \leq & p & \leq \frac{3}{2}(m+1)\delta & \pi_{oe} \\ (\frac{3}{2}m+1-\frac{1}{2m})\delta \leq & p & \leq (\frac{3}{2}m+\frac{7}{2}+\frac{4}{m-1})\delta & \pi_{(\frac{m-1}{4})w} \\ (\frac{3}{2}m+\frac{7}{2}+\frac{4}{m-1})\delta \leq & p & \leq (3m-1)\delta & \pi_{dh} \quad \textit{(by Corollary 4.3)} \end{array}$$

- *Else*

$$\begin{array}{rcll} (m+1)\delta \leq & p & \leq (m+\alpha+1)\delta & \pi_{oe} \\ (m+\alpha+1)\delta \leq & p & \leq p^*_{2\lfloor\frac{m-1}{8}\rfloor} & \pi_{(2\lfloor\frac{m-1}{8}\rfloor)w} \\ p^*_{2\lfloor\frac{m-1}{8}\rfloor} \leq & p & \leq (3m-1)\delta & \pi_{dh} \quad \textit{(by Corollary 4.1)} \end{array}$$

with $\alpha = \left\lfloor \frac{m+1}{2} \right\rfloor$

and $p^*_{2\lfloor\frac{m-1}{8}\rfloor} = (m + 4\left\lfloor\frac{m-1}{8}\right\rfloor + 1)\delta + \frac{\alpha - 4\lfloor\frac{m-1}{8}\rfloor}{2\alpha - 8\lfloor\frac{m-1}{8}\rfloor - 1}(m + 1 - 8\left\lfloor\frac{m-1}{8}\right\rfloor)\delta.$

Notice that $2\left\lfloor\frac{m-1}{8}\right\rfloor$ *is the largest possible even number n such that $m \geq 4n+1$.*

In particular for $9 \leq m \leq 14$, this is consistent with the observations presented in Table 4.3, which indicated $\{\pi_{id}, \pi_{oe}, \pi_{dh}, \pi_{2w}\}$ as a dominant set for $9 \leq m \leq 11$ and $m = 14$, respectively $\{\pi_{id}, \pi_{dh}, \pi_{oe}, \pi_{2w}, \pi_{3w}\}$ for $m = 12$ and $\{\pi_{id}, \pi_{dh}, \pi_{oe}, \pi_{3w}\}$ for $m = 13$.

4.6 Smooth Sea or Heavy Swell ?

In Section 4.5, we presented a family of cycle, (π_{nw}), formed using regularly disposed size-1 waves (sequences A_iA_{i-1}). Intuitively, the waves reduce the waiting

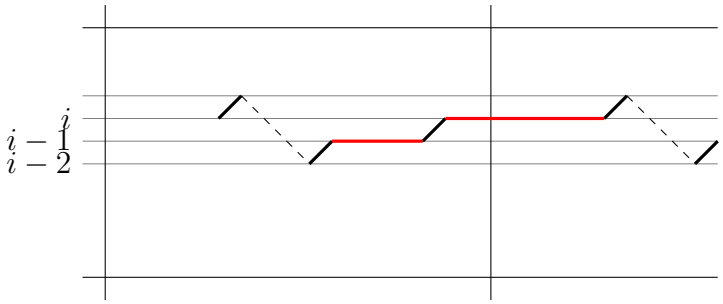


Figure 4.17: Sequence A_iA_{i-2}

time by increasing the travel time occurring between the loading and unloading of machines (d_i) (and thus the necessary waiting time) at the cost of an overall increase of the total travel time. The regular spacing of the waves means that d_i values are affected uniformly, which makes sense in balanced cells, where all machines have the same processing time. We showed that the trade-off between augmentation of the travel time and reduction of the waiting time is adequate for members of this family to outperform $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ for specific values of p. Moreover, for $9 \leq m \leq 15$, simulations suggest that a dominant set within 1-cycles can be formed using only $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ and a couple of cycles from (π_{nw}).

Could cycles similar in structure, but formed with larger waves (sequences A_iA_{i-h} with $h > 1$), become interesting for larger value of m? Size-2 waves can be easily dismissed by the following proposition:

Proposition 4.12. *Any cycle containing a size-2 wave (sequence A_iA_{i-2}) is dominated by π_{oe}.*

Proof. If a cycle contains the sequence A_iA_{i-2}, then activity A_{i-1} is comprised between A_{i-2} and A_i in a cyclic sense (see Figure 4.17). Therefore, the cycle time is at least $2p$ (at least p between A_{i-2} and A_{i-1} and at least p between A_{i-1} and A_i), which is larger than the cycle time of π_{oe}. □

However, this proof cannot be extended for $h > 3$. In this section, we use the bound defined by Proposition 4.5 to show that cycles formed similarly to members of (π_{nw}) with waves of size $h > 3$ are dominated.

Remember from Section 4.4 that potentially interesting cycles are formed by two turns and a number of waves, and may as such be decomposed in turbulence zones and still waters areas, which are similar in structure to the odd-even cycle (see Figure 4.9). Proposition 4.5 uses this decomposition to formulate a bound on the waiting time of a given cycle, $\hat{W}(l, d, p)$. This bound depends on the size of the still waters areas (l) and the time travelled between the loading and unloadings of machines located in this area (d). Basically, if two cycles space out successive

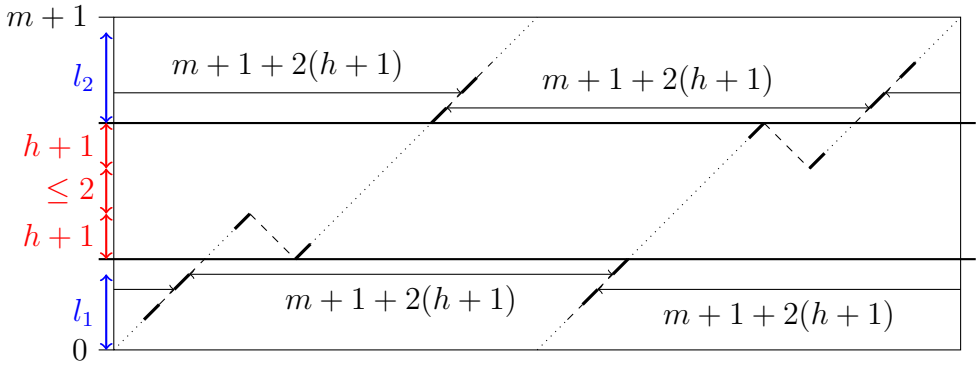


Figure 4.18: Representation of a cycle in $V_h(i_0, i_1)$

loadings and unloadings similarly (same value for d), then the one with the largest still water area (largest value of l), has the largest bound on its waiting time.

For cycles π_{nw} with n even, this bound is tight (Proposition 4.8). Thus π_{nw} performs at least as well as any cycle sharing the same value for d and l, and better for cycles sharing the same value for d and a larger value for l. For example, $\pi_{2w} = v(3, 6)$ performs at least as well as $v(5, 8)$ and better than $v(3, 5)$ or $v(5, 7)$.

In the following, using this bound on the waiting time, we show that cycles formed with an even number of size-h waves with $h > 2$, satisfying Equation (4.8) for every consecutive pair of waves, are always dominated by a member of (π_{nw}).

To better understand the idea, we first consider cycles with only 2 waves, for $h = 3$ and $h = 4$. We show that such cycles are dominated by π_{4w}. Then, we apply the same reasoning to cycles with an arbitrary even number of waves of size h, respectively for h odd and h even.

First, recall that for cycles π_{nw} with n even we have:

$$W_{\pi_{nw}}(p) = \hat{W}\Big((m - 4n)\delta, (m + 2n + 1)\delta, p\Big) \tag{4.16}$$

Let $\pi \in V_h(i_0, i_1)$ with $h > 2$ so that $i_1 - i_0 - h \in \{1, 2\}$. The cycle π is defined for $m \geq i_1 + h$; however it is dominated by π_{dh} for $m \leq 2(h+1)$. Indeed, the total travel time of π is $\Delta_\pi = 2(m+1)\delta + 4(h+1)$, which is greater than $T_{\pi_{dh}} = 3(m+1)\delta$ if $m \leq 2(h + 1)$. Thus, we only consider the case where $m > 2(h + 1)$. For all machines M_i with $i \leq i_0 - h$ or $i \geq i_1$, $d_i = (m+1)\delta + 2(h+1)\delta$ (see Figure 4.18). Then, by Proposition 4.5, we have:

$$W_\pi(p) \geq \hat{W}\Big((m - 2(h + 3))\delta, (m + 1 + 2(h + 1))\delta, p\Big) \tag{4.17}$$

h = 3 For $h = 3$, consider $\pi \in V_3(i_0, i_1)$ so that $i_1 - i_0 - 3 \in \{0, 2\}$. Let us compare π with π_{4w}, the cycle formed by four size-one waves. We assume $m \geq 16$

(otherwise π is dominated by π_{dh}), thus π_{4w} is defined. The travel time is identical for both cycles $\Delta_\pi(p) = \Delta_{\pi_{4w}}(p) = 2(m+1)\delta + 16\delta$. The additional spacing between loading and unloading for machines in the still waters areas is the same for both cycles, but the the size of these areas is larger for π_{nw} (same value for d, larger value for l). Formally, by Equation (4.17), the waiting time of π satisfies

$$W_\pi(p) \geq \hat{W}\Big((m-12)\delta, (m+9)\delta, p\Big)$$

while by Equation (4.16), the waiting time of π_{4w} is exactly

$$W_{\pi_{4w}}(p) = \hat{W}\Big((m-16)\delta, (m+9)\delta, p\Big)$$

thus, as $\hat{W}(l,d,p)$ is increasing in l, $W_\pi(p) \geq W_{\pi_{4w}}(p)$. Finally π is dominated by π_{4w}.

h=4 Now for $h = 4$, consider $\pi \in V_4(i_0, i_1)$ so that $i_1 - i_0 - 4 \in \{0, 2\}$. Its travel time is $T_\pi(p) = 2(m+1)\delta + 20\delta$. This case is more complicated than the previous one, as there is no cycle in the (π_{nw}) family as easily comparable to π: π_{4w} has less overall travel time, but also adds less travel between subsequent loadings and unloadings, while π_{6w} adds more travel between subsequent loadings and unloadings, but also has more overall travel time.

To show that π is indeed dominated by π_{4w}, we show that the potential increase in waiting time is compensated by the gain in overall travel time.

By Equation (4.17), the waiting time of π satisfies

$$W_\pi(p) \geq \hat{W}\Big((m-14)\delta, (m+11)\delta, p\Big)$$

As $\hat{W}(l,d,p)$ is increasing in l, we get

$$W_\pi(p) \geq \hat{W}\Big((m-16)\delta, (m+11)\delta, p\Big)$$

By expanding the expression of $\hat{W}$, we get

$$\begin{aligned} W_\pi(p) - W_{\pi_{4w}}(p) &\geq \hat{W}\Big((m-16)\delta, (m+11)\delta, p\Big) - \hat{W}\Big((m-14)\delta, (m+9)\delta, p\Big) \\ &\geq -2\frac{2\alpha(m-16)-1}{\alpha(m-16)}\delta \\ &\geq -4\delta \end{aligned}$$

As $\Delta_\pi - \Delta_{\pi_{4w}} = 4\delta$, $T_\pi - T_{\pi_{4w}} \geq 0$ and π is dominated by π_{4w}.

We can extend this reasoning to the case of n wave of arbitrary size $h > 2$ with n even; this is the following proposition:

Proposition 4.13. *Let $h > 2$, n even, and $i_0 < i_1 < \cdots < i_{n-1} \in \{0, \ldots, m+1\}$ so that for each $q \in [\![1, n-1]\!]$, $i_q - i_{q-1} - h \in \{0, 2\}$. Let $\pi \in V_h(i_0, \ldots, i_{n-1})$.*

If $m < 2n(h+1)\delta$ then π is dominated by π_{dh}; otherwise, π is dominated by $\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}$.

Proof. The travel time of π is $\Delta_\pi = 2(m+1)\delta + 2n(h+1)$. If $m < 2n(h+1)\delta$, then $\Delta_\pi \geq 3(m+1)\delta$, thus π is dominated by π_{dh}.

Now, we assume $m \geq 2n(h+1)\delta$. The cycle $\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}$ is then defined. Its travel and waiting time, depending on the parity of h are given respectively by:

$$\Delta_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}} = 2(m+1)\delta + 2n(h+1)\delta \text{ for } h \text{ odd} \tag{4.18}$$

$$\Delta_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}} = 2(m+1)\delta + 2nh\delta \text{ for } h \text{ even} \tag{4.19}$$

$$W_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}} = \hat{W}\Big((m-2n(h+1))\delta, (m+n(h+1)+1)\delta, p)\Big) \text{ for } h \text{ odd} \tag{4.20}$$

$$W_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}} = \hat{W}\Big((m-2nh)\delta, (m+nh+1)\delta, p\Big) \text{ for } h \text{ even} \tag{4.21}$$

For all machines M_i with $i \leq i_0 - h$ or $i \geq i_{n-1}$, one has $d_i = (m+1)\delta + 2(h+1)\delta$. Then, by Proposition 4.5, we have:

$$W_\pi(p) \geq \hat{W}\Big((m-n(h+3))\delta, (m+1+n(h+1))\delta, p\Big) \tag{4.22}$$

For h odd, $\Delta_\pi = \Delta_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}}$. Let us compare the waiting time of both cycles. As $h > 2$, we have $(m-n(h+3))\delta < (m-2n(h+1))\delta$, thus as $\hat{W}(l,d,p)$ is increasing in l, $W_\pi \geq W_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}}$, and π is dominated by $\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}$.

For n even, one has $\Delta_\pi - \Delta_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}} = 2n\delta$. Let us show that $W_\pi - W_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}} \geq -2n\delta$. As $h > 2$, one has $(m-n(h+3))\delta < (m-2n(h+1))\delta$. Thus as $\hat{W}(l,d,p)$ is increasing in l, one has:

$$W_\pi(p) \geq \hat{W}((m-2nh)\delta, 2(m+1+n(h+1))\delta, p).$$

From this, expanding the expression of $\hat{W}$, we get (with $\beta = \alpha(m-2nh)$):

$$\begin{aligned} W_\pi - W_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}} &\geq -\frac{2\beta-1}{\beta}n\delta \\ &\geq -2n\delta \end{aligned}$$

Thus $T_\pi - T_{\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}} \geq 0$ and π is dominated by $\pi_{(n\lfloor\frac{h+1}{2}\rfloor)^w}$. □

4.7 A conjecture on the best 1-cycle problem

In Section 4.5, we defined and studied a family of 1-cycles, (π_{nw}). The sequence of these cycles is formed of two forward turns around the cell, regularly altered by backwards sequences (waves) A_iA_{i-1}, disposed regularly so as to increase d_i values at each machine without creating discrepancies. We showed that cycles in this family outperforms the three classical cycles $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ for adequate instances. In Section 4.6, we consider cycles with the same regular structure, but with larger waves (A_iA_{i-h} with $h \geq 2$), and show that they are dominated by cycles in (π_{nw}): this suggests that larger waves are not interesting to form dominant cycles. This leads us to conjecture, consistently with the observations presented in Section 4.3, that considering only $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ and (π_{nw}) is enough to find a best 1-cycles for any instance:

***Conjecture* 4.1.** *$\{\pi_{id}, \pi_{dh}, \pi_{oe}\} \cup (\pi_{nw})_{n \leq \frac{m}{4}}$ is a dominant set within 1-cycles*

Remark 4.1. If Conjecture 4.1 is valid, then the best 1-cycles for $m \geq 8$ are given by Proposition 4.11.

For m small enough ($m \leq 11$), the number and size of possible waves of 1-cycles can be completely described with rather few conditions. Thus, in this case, we can conclude on the validity of the conjecture. From $m > 12$, the conditions are more tricky to describe.

Proposition 4.14. *Conjecture 4.1 is valid for $m \leq 11$.*

Proof. The case for $m \leq 8$ has been settled in Chapter 3, so we consider $8 \leq m \leq 11$.

Let π be a 1-cycle such that for some $p \in](m+1)\delta, 3(m+1)\delta[$, $T_\pi(p) < min(T_{\pi_{oe}}(p), T_{\pi_{dh}}(p))$. The travel time of π must be strictly less than $3(m+1)\delta$. As π cannot contain any size-two waves, π can contain either 1 size-1 wave (then $\Delta_\pi = (2(m+1)+4)\delta$), 2 size-1 waves ($\Delta_\pi = (2(m+1)+8)\delta$), 1 size-3 wave ($\Delta_\pi = (2(m+1)+8)\delta$).

Case 1: π contains 1 size-1 wave (there exists i such that $\pi = v(i)$). The lower bound of π_{1w}, given by Equation (4.14) also applies to π, thus we can apply the same reasoning to show that π is as well, dominated by $\{\pi_{oe}, \pi_{dh}\}$.

Case 2: π contains 1 size-3 wave (there exists i such that $\pi \in V_3(i)$).

We have $d_{min}(\pi) = (m+1)\delta$ and $\Delta_\pi = 2(m+1)\delta + 6\delta$, thus $T_\pi(p) \geq 2(m+1)\delta + 6\delta + (p-(m+1))\delta$. Using this bound and the cycle time of π_{oe}, we get that if $T_\pi(p) < T_{\pi_{oe}}(p)$, then $p > (m+1)\delta + 8\frac{\alpha}{\alpha-1}\delta$. We can verify that for $m \leq 11$, $p^* < (m+1)\delta + 8\frac{\alpha}{\alpha-1}\delta$, thus $T_\pi(p) \geq T_{\pi_{oe}}(p) \geq T_{\pi_{dh}}(p)$ for $p \leq p^*$ which contradicts Proposition 4.1. Thus π is dominated by $\{\pi_{oe}, \pi_{dh}\}$.

Case 3: π contains 2 size-1 wave (there exists i, j such that $\pi = V_1(i, j)$).

If $j-i-1 \in \{0, 2\}$, then by Proposition 4.5 $W_\pi(p) \geq \hat{W}(m-8, (m+5)\delta, p) = W_{\pi_{2w}}$, thus π is dominated by π_{2w}. Otherwise, we have $d_{min}(\pi) = (m+1)\delta$, thus $T_\pi(p) \geq 2(m+1)\delta + 8\delta + (p-(m+1))\delta$: similarly to Case 2, π is dominated by $\{\pi_{oe}, \pi_{dh}\}$.

□

If Conjecture 4.1 is valid, then the best 1-cycles for $m \geq 8$ are given by Proposition 4.11. As an instance of the problem is given by (m, p, δ), this would make the best 1-cycle problem solvable in linear time (thus polynomial).

Another interesting consequence would be the performance ratio of the classical cycles $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ within 1-cycles. Compared to cycles in the family (π_{nw}), these three cycles are easy to describe and understand by an human operator, whereas π_{nw} cycles do not seem to greatly improve the cycle time. Let R be the performance ratio of $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ within $\{\pi_{id}, \pi_{dh}, \pi_{oe}\} \cup (\pi_{nw})$:

$$R = \frac{min(T_{\pi_{dh}}, T_{\pi_{oe}}, T_{\pi_{id}})}{min_n(T_{\pi_{dh}}, T_{\pi_{oe}}, T_{\pi_{id}}, T_{\pi_{nw}})} \tag{4.23}$$

As the slope of π_{oe} is 2 and the slope of any cycle π_{nw} is at most 2, this ratio is worst for $p = p^*$, where $T_{\pi_{oe}}(p) = T_\pi(p) = 3(m+1)\delta$. Figure 4.19 shows the computed ratio at p^* for $9 \leq m \leq 200$, using Proposition 4.11 and the expression of the cycle time given by Proposition 4.8. We can verify algebrically that, if R_m denotes the ratio at p^* for m machines, each one of the sub-sequences $(R_m)_{m=i \mod 8}$ converges linearly towards 1, and thus the sequence (R_m) converges towards 1.

4.8 Conclusion

In this chapter, we addressed the best 1-cycle problem in arbitrarily large cells ($m > 8$, the case $m \leq 8$ being settled in Chapter 3). Though known to be NP-hard for circular regular cells, the complexity of this problem is still open in the balanced case. Observations of simulations up to $m \leq 14$ showed that 1-cycles non-dominated by $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ appear to follow a constrained structure, which we subsequently investigated. We first showed that non-dominated cycles necessarily take advantage of the circular configuration, in a similar way to the odd-even cycle π_{oe}: during one iteration, the robot circles the cell twice, forward, with scattered backward sequences ("waves") along the way which allow to decrease the waiting time at some machines while increasing the overall travel time. This leaves the question of the size, number, and placement of the waves in order to optimize the trade-off between waiting and travel time.

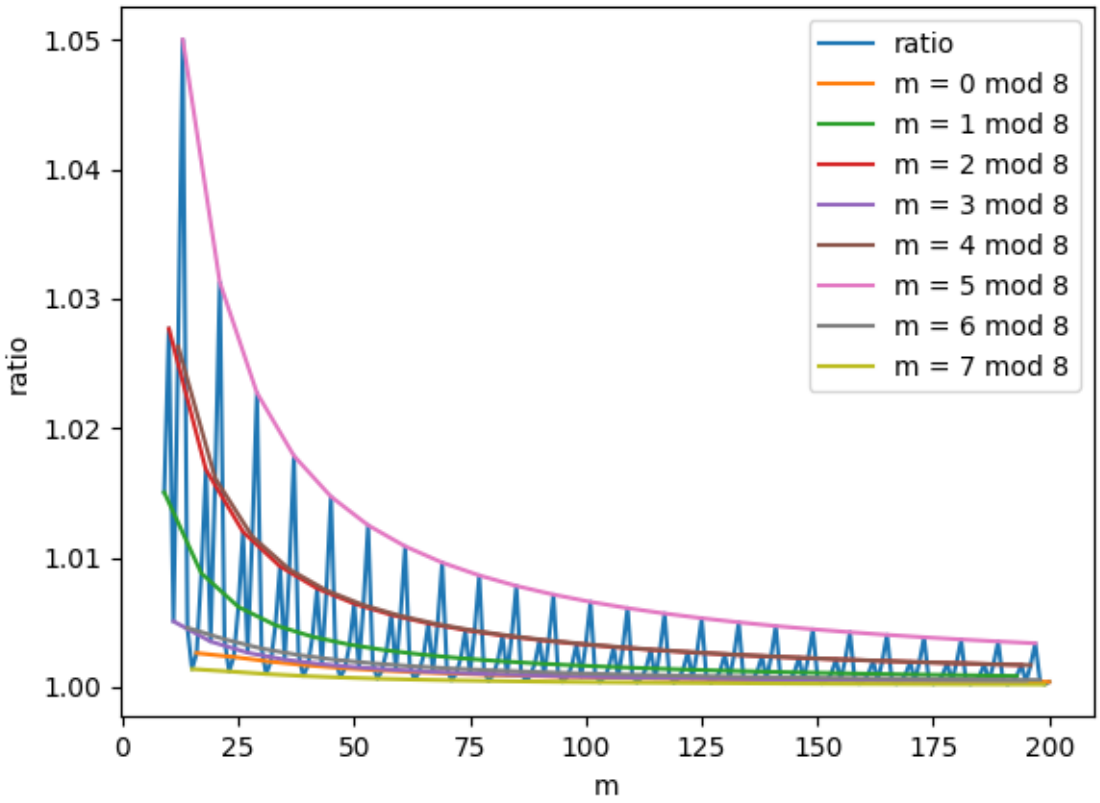


Figure 4.19: Performance ratio R of $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ within $\{\pi_{id}, \pi_{dh}, \pi_{oe}\} \cup (\pi_{nw})_n$ for $p = p^*$, as a function of the number of machines m.

Generalizing our observations, we were able to define and fully study a family of cycles, (π_{nw}) whose members outperform the classical cycles $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$ under the right conditions, formed with small, regularly disposed waves. We also showed that similar cycles formed using larger waves are dominated.

This led us to conjecture on the dominance of this family, alongside with the classical cycles $\{\pi_{id}, \pi_{dh}, \pi_{oe}\}$, over 1-cycles. If proven true, this would allow to settle the best 1-cycle problem in circular regular balanced cell in polynomial time. The main step to prove this conjecture for any m would be to rule out cycles with unevenly disposed waves (formally, not satisfying Equation (4.8)), or unevenly combining several wave sizes; however, for small enough cells, the low number of possibilities allows to settle it completely, thus describing the optimal 1-cycles for $m \leq 11$ machine circular regular balanced robotic cells.

Chapter 5

A Greedy Approach for a Rolling Stock Management Problem using Multi-Interval Constraint Propagation – ROADEF/EURO Challenge 2014

This chapter, a joint work with Hugo Joudrier, presents our contribution to the ROADEF/Challenge 2014. The competition gathered 37 teams from 20 countries for an industrial problem proposed by the french railway transportation company. We won the first prize in the junior category (see Figure 5.1) and ended up fourth in the overall ranking[1]. This chapter reproduces the resulting paper, published in Annals of Operation Research in June 2017 (Joudrier and Thiard, 2017). Other contributions to the competition include (Buljubašić et al., 2017, Catusse and Cambazard, 2014, Geiger et al., 2017).

5.1 Introduction

In this chapter, we present our approach of the Rolling Stock Unit Management problem presented in the ROADEF/EURO Challenge 2014 (Ramond and Nicolas, 2014). This problem combines in a single formulation various sub-problems of rolling stock management, such as assignment of trains to departures, platform assignment, routing inside the station, planning of maintenance operations... We propose a greedy algorithm to build an initial solution, allowing an incomplete

[1] http://challenge.roadef.org/2014/en/prizeResults.php

Figure 5.1: That was hard to fit in the luggage compartment.

coverage of arrivals and departures. This solution could serve as a basis for an optimization algorithm. Compared to the proposed problem, we make some simplifications by forbidding maintenance, junction and disjunction operations (note that train convoys might be used as such), which limits the number of coverable departures. Our solution is based on a greedy progressive assignment algorithm to assign arrivals (and corresponding trains) to departures, and a routing algorithm using multi-interval constraint propagation to prevent conflicts with already scheduled trains while keeping as much flexibility as possible.

The rest of this chapter is organized as follows: Section 5.2 presents a simplified model of the rolling stock management problem presented in the ROADEF/EURO Challenge 2014. Section 5.3 describes the pre-processing phase and the model used to represent the structure of the station. Section 5.4 details the multi-interval routing algorithm, while Section 5.5 describes the general assignment procedure. Note that the assignment problem is not treated as a whole, separately from the routing problem, but progressively. Finally Section 5.6 presents an overview of the results and some perspectives. In the following, we will use the concepts and notations introduced by Ramond and Nicolas (2014) (available at `https://hal.archives-ouvertes.fr/hal-01057324/file/Challenge_sujet_phaseFinale_140224.pdf`).

5.2 Simplified model

The original formulation, as presented by Ramond and Nicolas (2014), aimed at treating the rolling stock management problem as a whole, integrating constraints and costs of various nature. Focusing mainly on the routing constraints, we chose

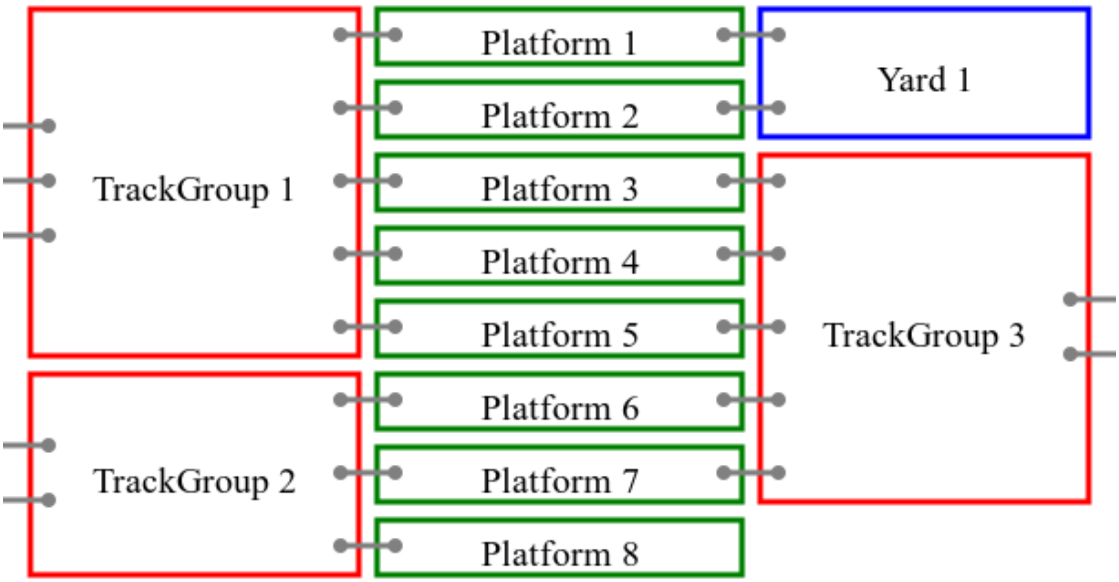


Figure 5.2: Example instance: railway station

to overlook some aspects of the problem in order to work on a simplified model while still providing valid solutions.

The original industrial problem consists in handling trains within a station over a biweekly planning horizon, by deciding their route and schedule inside the station, and assigning them to compatible departures. A station is composed of resources of several types (track-groups, single tracks and yards for parking, maintenance facilities for performing maintenance, platforms for arrival and departures), linked by gates. Routing of the trains inside the station must respect the length and capacity of the resources, as well as the trains' order on the resources and train/resource compatibility constraints. To be allowed to circulate, trains must be submitted to regular maintenance. Therefore, a train might be assigned to a departure only if it has enough time and mileage left before a maintenance is required for the journey. Maintenance operations can be performed in the stations on appropriate resources, at a cost. Some arrivals (joint arrivals) are composed of several trains joined together; likewise, some departures (joint departures) must be satisfied by several joint trains. Convoys of several joint trains might be formed (respectively separated) by performing junction and disjunction operations.

The two main simplifications that were made are described below. Their main consequence is to avoid modifying characteristics of the trains in the solution.

- Not performing maintenance operations. This choice simplifies the assignment decision process: the compatibility of a given train with a given departure depends only on the instance data (including the time / distance

remaining before maintenance), and not on the choice to perform maintenance on it.

- Not performing junction nor disjunction operations. That way, trains can be treated as immutable *convoys* from their arrivals to their departure. A *convoy* might be formed of a single train, or of several trains coming from a joint-arrival, which might then satisfy a joint-departure.

In this section we first extend the notations defined by Ramond and Nicolas (2014). Then, we use these new notations to formalize our simplified model.

5.2.1 Definitions

Definition : Convoys
A convoy v is defined by a set of trains t in $\mathcal{T}$. We note $\mathcal{V}$ the set of all the convoys.

$$\mathcal{V} := P(\mathcal{T}) \tag{5.1}$$

where P is the power set operator. The length and size of a convoy v in $\mathcal{V}$ can be accessed with:

$$size(v) = \#v \tag{5.2}$$
$$length(v) = \sum_{t \in v} length(t) \tag{5.3}$$

We also define $\mathcal{V}^+$, the set of convoys actually used in the solution.

Definitions : About arrivals and departures

As our model treats convoys and not individual trains, joint arrivals (respectively joint departures) can be treated as one single arrival (respectively departure). Thus, we work with a set of *extended arrivals* (respectively departures), defined in this section. An element of this set is either a unitary arrival (an arrival which is not a member of a joint arrival), or a joint arrival (a set of arrivals joint together, treated as a single element).

Let $\mathcal{U}_{arr}$ be the set of unitary arrival in opposition with $\mathcal{J}_{arr}$ the set of joint-arrivals (and similarly the set of unitary departures).

$$\mathcal{U}_{arr} := \{a \in \mathcal{A} \mid \forall \mathcal{B} \in \mathcal{J}_{arr}, a \notin \mathcal{B}\} \tag{5.4}$$
$$\mathcal{U}_{dep} := \{d \in \mathcal{D} \mid \forall \mathcal{B} \in \mathcal{J}_{dep}, d \notin \mathcal{B}\} \tag{5.5}$$

Now, we use the sets $\mathcal{J}_{arr}$ and $\mathcal{U}_{arr}$ to build a new set $Ext\mathcal{A}rr$ called set of all extended arrivals (and the set $Ext\mathcal{D}ep$ with $\mathcal{J}_{dep}$ and $\mathcal{U}_{dep}$).

$$Ext\mathcal{A}rr := \mathcal{J}_{arr} \cup \{\{a\} \mid a \in \mathcal{U}_{arr}\} \tag{5.6}$$

$$Ext\mathcal{D}ep := \mathcal{J}_{dep} \cup \{\{d\} \mid d \in \mathcal{U}_{dep}\} \tag{5.7}$$

Definitions : compatibility

For each convoy $v \in \mathcal{V}$ and departure in $d \in Ext\mathcal{D}ep$, a value $Comp(v, d) \in \{0, 1\}$ is defined. If $Comp(v, d) = 1$, v and d are said to be compatible. This value takes into account several parameters of the initial formulation such as size, categories, time and distance remaining before maintenance, and so on.

Similarly, a value $Comp(v, a) \in \{0, 1\}$ is defined to express compatibility between a convoy $v \in \mathcal{V}$ and an arrival $a \in Ext\mathcal{A}rr$.

Similarly, for each convoy $v \in \mathcal{V}$ and resource $r \in \mathcal{R}$, $Comp(v, r) \in \{0, 1\}$ expresses the compatibility of all trains of v with the resource r.

Definition : Connectors

Connectors are couples of gates, representing transition between resources. Using this concept, we can model the station as a directed graph.

The set of connectors noted $\mathcal{CO}$ is defined by

$$\mathcal{CO} = \left\{ (g, g') \in \mathcal{G}_r \times \mathcal{G}_{r'} \;\middle|\; \begin{array}{l} neigh_g = g' \\ neigh_{g'} = g \end{array} \right\} \tag{5.8}$$

Then a connector $co \in \mathcal{CO}$ is a couple of gates (g, g') such that a convoy can leave the resource r ($g \in \mathcal{G}_r$) by the gate g to enter the resource r' by g' ($g' \in \mathcal{G}_{r'}$).

It is important to note that the connector (g, g') is not equivalent to the connector (g', g). The first one means an exit by g and an entry by g' whereas the second one means an exit by g' and an entry by g.

The train station can then be modeled by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where the vertices represent connectors and the arcs represent resources (Figure 5.3). This allows to represent reverse and non-reverse resources[2].

Some values are associated with each edge of the graph $\mathcal{G}$ such as the $minTrTime$ and the $maxTrTime$ which are the minimal and the maximal traveling time to go from a connector to an other one through a resource. These parameter values aggregate several parameters from the original problem formulation, depending on the resources type, such as $trTime$ for trackgroups, $maxDwellTime$ for platforms, $revTime$ for the reverse operations on single tracks, platform, yards...

[2]All resources have two (opposite) sides. On *reverse resources*, convoys are allowed to enter and exit on the same side, whereas this is forbidden on *non-reverse resources*.

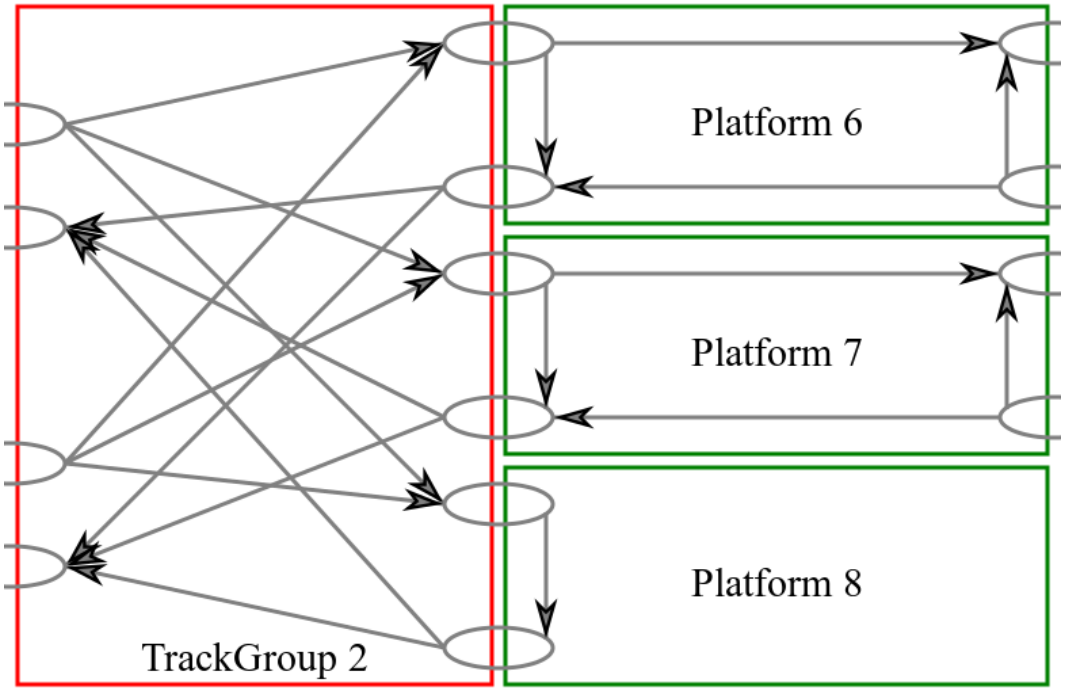


Figure 5.3: Directed Graph Modelisation

Definition : Path through the station
A path through the station is defined by an alternate sequence of connectors in $\mathcal{CO}$ and resources in $\mathcal{R}$.

$$p := (co_0, r_0, co_1, r_1, co_2 \dots co_{n-2}, r_{n-2}, co_{n-1}, r_{n-1}, co_n) \tag{5.9}$$

The size of the path p is accessible via $size(p)$ and returns the number of resources crossed along the path.

$$size((co_0, r_0, co_1, r_1, co_2 \dots co_{n-2}, r_{n-2}, co_{n-1}, r_{n-1}, co_n)) = n \tag{5.10}$$

An n-sized path going through resources $r_0 \dots r_{n-1}$ requires $n + 1$ connectors $co_0 \dots co_n$.

Schedule of a path
The schedule of a path p, noted $sched_p$, is a tuple of instants t in the horizon $\mathcal{H}$ which are in bijection with the connectors of the path. These values describe the time of transition from a resource to the other. Let p be a path composed by n resources (Eq. 5.9). Then a schedule for p is defined by

$$sched_p = (t_0, t_1, t_2 \dots t_{n-2}, t_{n-1}, t_n) \tag{5.11}$$

where t_i in $sched_p$ is paired with co_i in p, meaning that at t_i the connector co_i is used to leave resource r_{i-1} and resource r_i.

5.2.2 Decision Variables

Arrival and Departure assignments

A convoy may come from an arrival (all trains composing the convoy enter the system as elements of this arrival), and be assigned to a departure (all trains composing the convoy leave the system satisfying elements of this departure).

$$\forall v \in \mathcal{V}, \quad \begin{cases} \forall a \in Ext\mathcal{A}rr,\ from(v,a) \in \{0,1\} \\ \forall d \in Ext\mathcal{D}ep,\ assigned(v,d) \in \{0,1\} \end{cases} \tag{5.12}$$

Paths and schedules of convoys

A convoy v in $\mathcal{V}$ may be routed, thus associated with a path $path(v)$ which is paired with a schedule $sched(v)$, of size n_v.

$$path(v) = (co_0, r_0, co_1, r_1, co_2 \dots co_{n-2}, r_{n-2}, co_{n-1}, r_{n-1}, co_{n_v}) \tag{5.13}$$

$$sched(v) = (t_0, t_1, t_2 \dots t_{n-2}, t_{n-1}, t_{n_v}) \tag{5.14}$$

5.2.3 Constraints

Convoys paired with Arrivals and Departures

Recall that $\mathcal{V}^+$ is the set of all convoys scheduled in the solution. Let v be a convoy in $\mathcal{V}$. If v is scheduled in the solution then v is paired with an arrival a in $Ext\mathcal{A}rr$ and a departure d in $Ext\mathcal{D}ep$.

$$v \in \mathcal{V}^+ \Rightarrow \begin{cases} \sum_{a \in Ext\mathcal{A}rr} from(v,a) = 1 \\ \sum_{d \in Ext\mathcal{D}ep} assigned(v,d) = 1 \end{cases} \tag{5.15}$$

Otherwise, v is not an element of $\mathcal{V}^+$ and there is no arrival nor departure assigned to it.

$$v \notin \mathcal{V}^+ \Rightarrow \begin{cases} \forall a \in Ext\mathcal{A}rr, from(v,a) = 0 \\ \forall d \in Ext\mathcal{D}ep, assigned(v,d) = 0 \end{cases} \tag{5.16}$$

A convoy v cannot be assigned to an incompatible arrival a or departure d. In other words :

$$Comp(v,a) = 0 \Rightarrow from(v,a) = 0 \tag{5.17}$$
$$Comp(v,d) = 0 \Rightarrow assigned(v,d) = 0 \tag{5.18}$$

Usage of resources

The usage in term of capacity and length has to be respected. Let v be a convoy, with the schedule $sched(v)$ associated to the path $path(v)$. For each convoy v in $\mathcal{V}^+$ and each instant t in $\mathcal{H}$, we define $use(v,t)$ the resource r in $\mathcal{R}$ used by the convoy v at t.

$$use(v,t) = \begin{cases} r_i \in path(v) & \textbf{if } \exists t_i, t_{i+1} \in sched_v, t_i \leq t < t_{i+1} \\ \emptyset & \textbf{otherwise} \end{cases} \tag{5.19}$$

Then we define the value $use(v,t,r)$, which is 1 if and only if the resource r is used at time t in the schedule $sched(v)$ of the path $path(v)$ associated to the convoy v.

$$use(v,t,r) = \begin{cases} 1 & \textbf{if } use(v,t) = r \\ 0 & \textbf{otherwise} \end{cases} \tag{5.20}$$

Length and capacity constraints

$$\forall r \in \mathcal{R}, \forall t \in \mathcal{H}, \begin{cases} \sum_{v \in \mathcal{V}^+} (use(v,t,r) * size(v)) \leq capa(r) \\ \sum_{v \in \mathcal{V}^+} (use(v,t,r) * length(v)) \leq length(r) \end{cases} \tag{5.21}$$

Valid path

Let v be a convoy in $\mathcal{V}^+$, and $path(v)$ the associated path, of size n_v. To be valid, the path, defined as in (Eq. 5.13), followed by v must satisfy the following constraints.

$$co_0 = (\emptyset, g_{initial}) \tag{5.22}$$
$$co_{n_v} = (g_{final}, \emptyset) \tag{5.23}$$

This means the convoy enters the station via the first connector of the path, and leaves it via the last connector of the path.

Let us now consider the i-th connector co_i, composed of two gates g_{out} and g_{in}. By definition of connectors, $neighg_{out} = g_{in}$ and $neigh(g_{in}) = g_{out}$. Additionally, for the path to be valid, each resource must be consistent with the adjacent connectors:

$$r_{g_{out}} = r_{i-1} \tag{5.24}$$
$$r_{g_{in}} = r_i \tag{5.25}$$

A convoy v cannot use an incompatible resource r:

$$\forall r \in \mathcal{R}, Comp(v, r_i) = 0 \Rightarrow r_i \notin path(v) \tag{5.26}$$

Since v is a convoy of the solution, it is assigned to an arrival a in $Ext\mathcal{A}rr$ as well as a departure d in $Ext\mathcal{D}ep$. Then the path followed by the convoy v must begin with the arrival sequence $arrSeq_a$ (Eq. 5.27) and end with the departure sequence $depSeq_d$ (Eq. 5.28).

$$arrSeq_a = (tg_0^a, \ldots, tg_{n_1}^a) \tag{5.27}$$
$$depSeq_d = (tg_0^d, \ldots, tg_{n_2}^d) \tag{5.28}$$

The arrival (resp. departure) sequence is composed by $n_1 + 1$ (resp. $n_2 + 1$) track-groups.

$$\forall r_i \in path(v), \begin{cases} i \leq n_1 \Rightarrow r_i = tg_i^a \\ n_v - n_2 - 1 \leq i \Rightarrow r_i = tg_j^d \textbf{ with } j = n_v - n_2 - 1 + i \end{cases} \tag{5.29}$$

Since the arrival and the departure must take place on a platform, we have:

$$r_{n_1+1} \in \mathcal{P} \quad \textbf{and} \quad r_{n_v-n_2-2} \in \mathcal{P} \tag{5.30}$$

Valid schedule

Let $v \in \mathcal{V}^+$ be a convoy of the solution. The schedule $sched(v)$ must respect the delay imposed by the graph $\mathcal{G}$ to go from one connector to another through a resource.

$$\forall t_i, t_{i+1} \in sched(v), \begin{cases} minTrTime(co_i, co_{i+1}) \leq t_{i+1} - t_i \\ t_{i+1} - t_i \leq maxTrTime(co_i, co_{i+1}) \end{cases} \tag{5.31}$$

The arrival time $arrTime_a$ and departure time $depTime_d$ must be respected. We assume the arrival sequence is composed by n_1+1 track-groups and the departure sequence by n_2+1 track-groups.

$$\forall t_i \in sched(v), \begin{cases} i = n_1 + 1 \Rightarrow t_i = arrTime_a \\ i = n_v - n_2 - 1 \Rightarrow t_i = depTime_d \end{cases} \tag{5.32}$$

Conflicts between convoys

On single resources Let v_1, v_2 in $\mathcal{V}^+$ be two convoys of the solution, $path(v_1) = (co^1_0, r^1_0, \ldots, co^1_{n_{v_2}})$, $path(v_2) = (co^2_0, r^2_0, \ldots, co^2_{n_{v_2}})$ their respective paths, and $sched(v_1) = (t^1_0, \ldots t^1_{n_{v_1}})$, $sched(v_2) = (t^2_0, \ldots t^2_{n_{v_2}})$ the respectively associated schedules.

If a same single resource (platform, single track or maintenance facility) belongs to both path, the associated entry times must be different :

$$\forall r^1_i \in sched(v_1), r^2_j \in sched(v_2),\ r^1_i = r^2_j \Rightarrow t^1_i \neq t^2_j \tag{5.33}$$

Moreover, depending on the entry and exit connectors of both convoys, the following constraints on their entry and exit times must be enforced to preserve the order of the convoys on the resource (without loss of generality, we assume $t^i_1 < t^2_j$) :

$$\begin{aligned} &\forall r^1_i \in sched(v_1), r^2_j \in sched(v_2) \mid r^1_i = r^2_j, t^i_1 < t^2_j, \\ &\begin{cases} (co^2_j = co^1_{i+1} \wedge co^2_j \neq co^2_{j+1}) \Rightarrow t^2_j > t^1_{i+1} \\ (co^2_{j+1} = co^1_{i+1} \wedge co^2_j \neq co^2_{j+1}) \Rightarrow t^2_{j+1} > t^1_{i+1} \\ (co^2_j = co^1_{i+1} \wedge co^2_j = co^2_{j+1}) \Rightarrow (t^2_{j+1} > t^1_{i+1} \vee t^2_j > t^1_{i+1}) \end{cases} \end{aligned} \tag{5.34}$$

On track groups On track groups, a specific conflict constraint applies, depending on the respective order of the entry and exit gates of both convoys. Though this constraint was implemented in the final solution, we do not reproduce it here for the sake of simplicity. One can refer to Ramond and Nicolas (2014).

At most one convoy assigned per extended arrival and departure

$$\forall a \in Ext\mathcal{A}rr, \sum_{v \in \mathcal{V}^+} from(v, a) \leq 1 \tag{5.35}$$

$$\forall d \in Ext\mathcal{D}ep, \sum_{v \in \mathcal{V}^+} assigned(v, d) \leq 1 \tag{5.36}$$

Example (Figure 5.4)
Let us consider the following instance: the sets of arrivals $\mathcal{A} = \{a_1 \dots a_5\}$ and departures $\mathcal{D} = \{d_1 \dots d_6\}$; The sets $\mathcal{J}_{arr} = \{\{a_2, a_3, a_4\}\}$ and $\mathcal{J}_{dep} = \{\{d_4, d_5, d_6\}\}$. Then

$$Ext\mathcal{A}rr = \{ea_1 = \{a_1\}, ea_2 = \{a_2, a_3, a_4\}, ea_3 = \{a_5\}\}$$
$$Ext\mathcal{D}ep = \{ed_1 = \{d_1\}, ed_2 = \{d_2\}, ed_3 = \{d_3\}, ed_4 = \{d_4, d_5, d_6\}\}$$

The convoys v_1 and v_2 where

- $v_1 = \{t_1\}$, with $from(v_1, ea_1) = 1$ and $assigned(v_1, ed_2) = 1$
- $v_2 = \{t_2, t_3\}$, with $from(v_2, ea2) = 1$ and $assigned(v_2, ed_4) = 1$

are represented on the Figure 5.4.

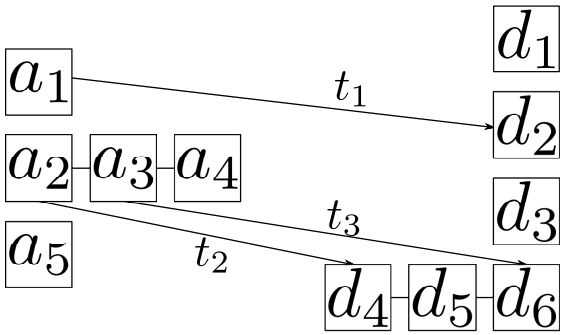


Figure 5.4: Convoys with provenance and assignment to departures.

5.2.4 Objective function

The original problem featured two optimization criteria assembled in a weighted sum: maximization of the covered arrival and departures, and minimization of performance costs (platform usage costs, preferred reuse of trains, etc.). We focused on the maximization of covered arrival and departures (while of course limited by our choices to disregard maintenance, junction and disjunction operations). With the formalism defined in this section, maximizing arrivals and departures coverage comes down to maximizing the number of trains used in the solution:

$$\max \sum_{v \in \mathcal{V}^+} size(v) \tag{5.37}$$

5.3 Data Structures and Pre-processing

In the remainder of the chapter, we will use the instance represented on Figure 5.2 as an example.

A maximum amount of convoys needs to be scheduled using the graph $\mathcal{G}$ previously defined. In order to reduce the cost of the routing phase by avoiding redundant treatment of very similar paths, we need a simplified structure with a higher aggregation level.

The simplification consists of aggregating resources in groups (Figure 5.5) of resources sharing the same characteristics and neighborhood (Rogers et al., 1991). We define the set $\mathcal{RG}$ of resources groups such that each resource $r \in \mathcal{R}$ is contained in one group $rg \in \mathcal{RG}$ with the following properties:

$$\forall rg \in \mathcal{RG}, \forall r1, r2 \in rg, neighSet_{r1} = neighSet_{r2} \wedge type(r1) = type(r2) \quad (5.38)$$

where *type* is defined as follow:

$$type : r \in \mathcal{R} \rightarrow \begin{cases} 0 \ if \ r \in \mathcal{K} \\ 1 \ if \ r \in \mathcal{P} \\ 2 \ if \ r \in \mathcal{S} \\ 3 \ if \ r \in \mathcal{Y} \\ 4 \ if \ r \in \mathcal{F} \end{cases} \quad (5.39)$$

The goal of this process is to break the systematic and redundant treatment of similar resources while allowing for a more global reasoning.

Once the simplification is done, we can define a new graph $\mathcal{G}'((\mathcal{V}', \mathcal{E}'))$, more global, enclosing the graph $\mathcal{G}$ where vertices are groups connectors $\mathcal{GCO}$ and arcs are resources groups $\mathcal{RG}$. These two graphs represent the station with different levels of granularity: the level which needs to be used depends on the step of the algorithm.

To simplify further the routing phase, we choose to consider the shortest path as the preferred path between two resources depending on the category[3]. Thus, shortest paths between any two groups of resources, or rather group connectors, are computed during the pre-processing phase using *Floyd-Warshall* algorithm (Floyd, 1962). During the process, a matrix of minimal and maximal travel time along this path is computed. Each resource $r \in R$ can be considered to have a minimum and a maximum usage time. For any track-group k, the minimum bound is equal to the maximum one: $trTime_k$. The other resources have a minimum bound equal

[3]Trains of the same *category* share similar characteristics (such as length).Some resources may only be compatible with some trains categories (Ramond and Nicolas, 2014).

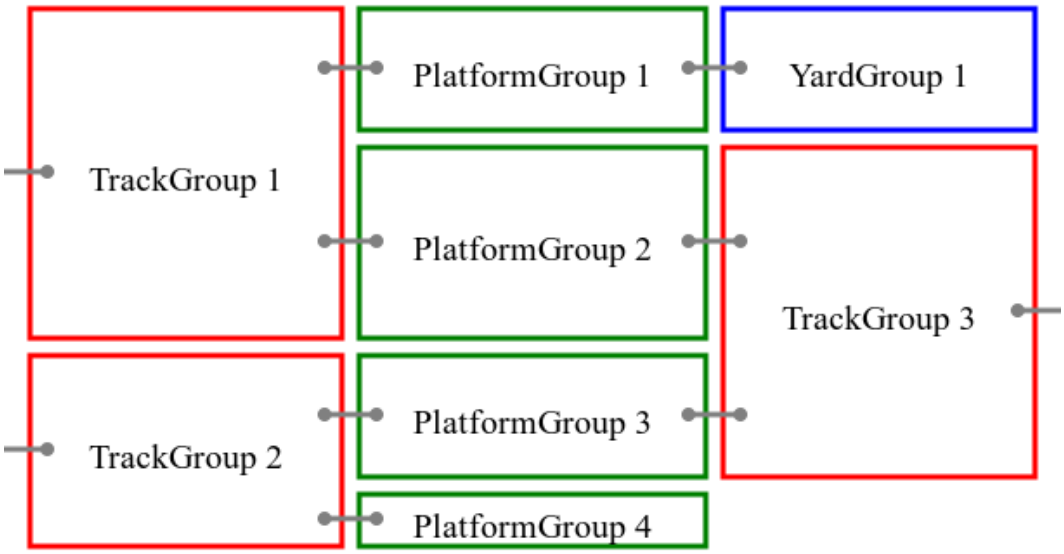


Figure 5.5: Resource Aggregation

to $minResTime$ or $\max(minResTime, revTime)$ (in case of similar input/output side) and the platforms have an upper bound $maxDwellTime$.

Let $gc1$, $gc2$ be two group connectors and c a category. First, we compute the shortest path to reach $gc2$ from $gc1$. Then we note $minTrTime_c(gc1, gc2)$ the lower time bound and $maxTrTime_c(gc1, gc2)$ the upper time bound to travel from the input connector $gc1$ to the output connector $gc2$ on the path found compatible with category c.

5.4 Routing Procedure

This section deals with the effective scheduling and routing of a train along a given group-level path. To schedule a train, the routing procedure considers the resource group-level path computed during the pre-processing phase, then tries to schedule the train at the resource-level, taking into account resource capacities, maximum lengths, minimal and maximal transition time, and conflicts with already scheduled trains. This is done by constraint propagation on multi-interval variables. Section 5.4.1 introduce the notion of multi-interval variables in this context, and Section 5.4.2 details the routing procedure.

5.4.1 Multi-interval variables

We use constraint propagation (Benhamou and Granvilliers, 2006) on hull and boxes (ILOG, 1999) using multi-intervals variables to filter solutions of the *Path*

Scheduling problem. First we introduce the notion of time interval $\mathcal{IH}$ (Eq. 5.40) in the planning horizons $\mathcal{H}$, as defined by Moore (1966):

$$\mathcal{IH} := \left\{ [\underline{h}, \overline{h}] \middle| \begin{array}{l} (\underline{h}, \overline{h}) \in \mathcal{H}^2 \\ \underline{h} \leq \overline{h} \end{array} \right\} \tag{5.40}$$

We note $[h]$ the interval $[\underline{h}, \overline{h}]$. Let $[h_1]$, $[h_2]$ be two intervals, we define the set operations $\cap$, $\cup$:

$$\begin{array}{l} [h_1] \cap [h_2] = [max(\underline{h_1}, \underline{h_2}), min(\overline{h_1}, \overline{h_2})] \\ [h_1] \cup [h_2] = [min(\underline{h_1}, \underline{h_2}), max(\overline{h_1}, \overline{h_2})] \end{array} \tag{5.41}$$

A large number of interval arithmetic libraries exist, see (Knüppel, 1994, Lerch et al., 2006).

A generalization of $\mathcal{IH}$ is the power set $\mathcal{P}(\mathcal{IH})$. We call $\mathcal{MIH} = \mathcal{P}(\mathcal{IH})$ the set of all the time multi-intervals in the planning horizon. Let mi_1 and mi_2 be two elements of $\mathcal{MIH}$, the multi-interval intersection is defined as below:

$$mi_1 \cap mi_2 = \bigcup_{\substack{i_1 \in mi_1 \\ i_2 \in mi_2}} i_1 \cap i_2 \tag{5.42}$$

We introduce the operators *lo* and *up* on intervals and multi-intervals, which represent the lower and upper bounds of these quantities (see Figure 5.6 for an example).

$$\forall x \in \mathcal{IH} \left\{ \begin{array}{l} lo(x) = \underline{x} \\ up(x) = \overline{x} \end{array} \right. \quad \textbf{and then} \quad \forall y \in \mathcal{MIH} \left\{ \begin{array}{l} lo(y) = \min\limits_{x \in y} lo(x) \\ up(y) = \max\limits_{x \in y} up(x) \end{array} \right. \tag{5.43}$$

Mathematical comparison operators can be defined to deal with intervals and multi-intervals. Let x be an interval or a multi-interval quantity. Then the comparison of x with a real value y implies conditions over the bounds of x.

$$x \leq y \quad \Leftrightarrow \quad up(x) \leq y \tag{5.44}$$

$$y \leq x \quad \Leftrightarrow \quad y \leq lo(x) \tag{5.45}$$

The inequality constraints are used to reduce the domain of the values through a set of methods called contractors (Chabert and Jaulin, 2009). Atomic and meta-contractors can be considered as basic elements of a new paradigm called contractor programming in order to perform efficient global optimization algorithm (Ninin, 2015) exploiting the properties of groups of constraints. In the next section we present a contraction method for the *Path Scheduling Problem* previously introduced.

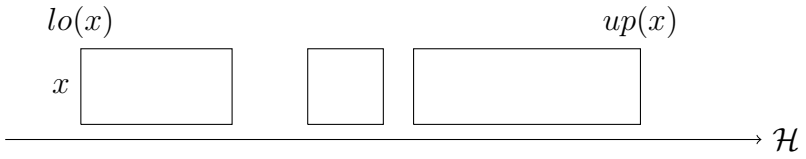


Figure 5.6: Example of a multi-interval, with its lower and upper bounds.

The idea behind the routing procedure, described in the next section, is to consider scheduling variables as two types of multi-interval variables, *transition* variables t_i and *usage* variables u_i. That way, for a given convoy's schedule, the multi-interval variable u_i represents the time intervals during which the corresponding resource of the convoy's path r_i is available for the convoy (with respect, amongst other, to length and capacity constraints). Similarly, the multi-interval variable t_i represents the time intervals during which the convoy may exit the previous resource and enter resource r_i through connector co_i (with respect to other convoy's transitions).

The path and schedule definition presented in the simplified model are extended accordingly, by adding to the schedule usage time variables u_i in bijection with the resources r_i in the path:

$$path(v) = (co_0, r_0, co_1, r_1, co_2 \dots co_{n-2}, r_{n-2}, co_{n-1}, r_{n-1}, co_n) \tag{5.46}$$

$$sched(v) = (t_0, u_0, t_1, u_1, t_2 \dots t_{n-2}, u_{n-2}, t_{n-1}, u_{n-1}, t_n) \tag{5.47}$$

The procedure consists in refining those multi-intervals by constraint propagation.

However, propagating constraints on multi-intervals variables is very expensive. Computing speed may be increased using lazy evaluation (Madsen and Jensen, 1999). Laziness is especially used and developed in functional programming languages (Johnsson, 1984) such as *Haskell* (Hudak et al., 2007). It consists into a set of methods in order to limit the number of computations by non-evaluating quantities which are not required. This approach is not yet fully implemented in our solution, so there is still room for progress and potential speed-up.

Definition : $\mathcal{H}_{r,l,s}$
Let r be in $\mathcal{R}$, and l, s in $\mathbb{N}$. We define the set $\mathcal{H}_{r,l,s}$ as all the instants t in the planning horizons $\mathcal{H}$ such that the resource can be used by a convoy of size s with a total length l.

$$\mathcal{H}_{r,l,s} = \left\{ t \in \mathcal{H} \;\middle|\; \begin{array}{c} \sum_{v \in \mathcal{V}^+} (use(v,t,r) \times size(v)) \leq capa(r) - s \\ \sum_{v \in \mathcal{V}^+} (use(v,t,r) \times length(v)) \leq length(r) - l \end{array} \right\} \tag{5.48}$$

5.4.2 Algorithm

In this section, we detail the procedure used to route and schedule a convoy along a group-level path (Algorithm 1).

Let us consider a m-sized *convoy*, of total length l, and a group-level path *GlobalPath*. *GlobalPath* is formalized as an alternating list of connector-groups and resource-groups (gco_0, gr_0, ..., gco_1, gr_{n-1}, gco_n).

The procedure returns, if possible, a resource-level *path* (formalized as an alternating list of connectors and resources (co_0, r_0, ..., co_1, r_{n-1}, co_n) and a matching *schedule*. A *Schedule* is an alternate list of transition multi-interval variables, and usage multi-interval variables $(t_0, u_0, \ldots, t_{n-1}, u_{n-1}, t_n)$. t_i represents the available intervals for transition between resources r_{i-1} and r_i on connector co_i, and u_i the available intervals for usage of resource r_i.

Schedule is initialized using *InitialSchedule*. In practice, we initialize all multi-interval variables to the full planning horizon, except for the transitions corresponding to the arrival and departure of the convoy, which are reduced to the arrival (respectively departure) time (see Figure 5.7a).

Algorithm (Alg. 1) consists of an iteration over every path $path := (co_0, r_1, \ldots, r_{n-1}, co_n)$ enclosed in the *GlobalPath*. Each iteration is divided in 2 steps (*FilterResourceUsagePath*, and *FilterConnectorPath*), described below.

Algorithm 1 : *Routing*

Input: *convoy*
Input: $globalPath := (gco_0, gr_0, \ldots, gco_1, gr_{n-1}, gco_n)$
Input: $initSchedule := (t_0, u_0, \ldots, t_{n-1}, u_{n-1}, t_n)$
1: **for all** $path = (co_0, r_0, \ldots, co_{n-1}, r_{n-1}, co_n) \in globalPath$ **do**
2: $schedule \leftarrow FilterResourceUsagePath(convoy, path, initSchedule)$
3: **if** $(schedule) \neq \emptyset$ **then**
4: $schedule \leftarrow FilterConnectorPath(convoy, path, schedule)$
5: **if** $(schedule) \neq \emptyset$ **then**
6: **return** $(path, schedule)$
7: **end if**
8: **end if**
9: **end for**
10: **return** $(\emptyset, \emptyset)$

1. Refinement of the multi-interval scheduling variables (t_0, u_0, ..., t_{n-1}, u_{n-1}, t_n) associated with the path p by filtering on each resource r_k, taking into account capacity, length and usage time constraints (Method *FilterResourceUsagePath*) using the following contractors. An example of this

step is unfolded (Figure 5.7) on the simplified instance represented on Figure 5.2, using the parameters given in Tables 5.1 and 5.2, with the parameters $minResTime = 00:02:00$ and $revTime = 00:05:00$.

$$u_k \subseteq u_{max_k} \tag{5.49}$$

where u_{max_k} is the minimal multi-interval containing $\mathcal{H}_{r_k,l,m}$. This constraint (Eq. 5.49) represents the limitation in term of length and/or capacity of the resource (Figure 5.7a, 5.7b).

$$lo(t_k) \leq u_k \leq up(t_{k+1}) \tag{5.50}$$

This constraint (Eq. 5.50) translates the fact that the convoy cannot occupy the resource before entering it and after exiting it (Figure 5.7b, 5.7c).

$$\begin{aligned} minTrTime_{r_k}(co_k, co_{k+1}) &\leq lo(t_{k+1}) - lo(t_k) \\ up(t_{k+1}) - up(t_k) &\leq maxTrTime_{r_k}(co_k, co_{k+1}) \end{aligned} \tag{5.51}$$

These two constraints (Eq. 5.51) express the relation between the enter time t_k and the exit time t_{k+1} depending on the minimal and maximal usage time of the resource r_k (Figure 5.7c, 5.7d).

$$\forall u \in u_k, \{u\} \cap t_k \neq \emptyset \wedge \{u\} \cap t_{k+1} \neq \emptyset \tag{5.52}$$

This constraint (Eq. 5.52) means that the resource r_k is able to welcome the convoy from its entrance to its exit (Figure 5.7d, 5.7e).

$$\begin{aligned} t_k &\subseteq u_k \\ t_{k+1} &\subseteq u_k \end{aligned} \tag{5.53}$$

This constraint (Eq. 5.53) means that the convoy can enter and exit the resource r_k only when r_k is able to receive it (Figure 5.7e, 5.7f).

2. Similarly, procedure *FilterConnectorPath* contracts the multi-interval variables $(t_0, u_1, \ldots, u_{n-1}, t_n)$ associated with *path* by computing for each resource r_k the conflicts of enter time t_{k-1} and exit times t_{k+1} with already scheduled convoys occupying r_k and propagating the associated constraints. This step also propagates again, when necessary, some of the previous constraints (Eq. 5.51, 5.52).

After both filtering procedures, if *schedule* is non-empty (which means it is possible to schedule the train along *path*), the algorithm returns, if not, it moves on to the next path.

Id	arrTrain	Time	Sequence	Platform	idealDwell	maxDwell
Arr1	Train1	08:00:00	TrackGroup1	Platform1	00:03:00	00:12:00
Dep1		08:17:00	TrackGroup1	Platform2	00:04:00	00:10:00

Table 5.1: Example instance: arrivals/departures listing

Resource	Begin	End
Yard1	08:01:00	08:02:00
Yard1	08:08:00	08:09:00
Yard1	08:15:00	08:16:00

Table 5.2: Example instance: imposed consumptions listing

Finally, the train is scheduled according to an "earliest possible time" strategy within the allowed time intervals : each transition multi-interval is contracted to its lower bound, and usage multi-intervals are contracted accordingly. Note that the malleability of the structure leaves the possibility of implementing other strategies to avoid conflict with subsequent train schedules or optimize the enter and exit time on each resource to reduce penalty.

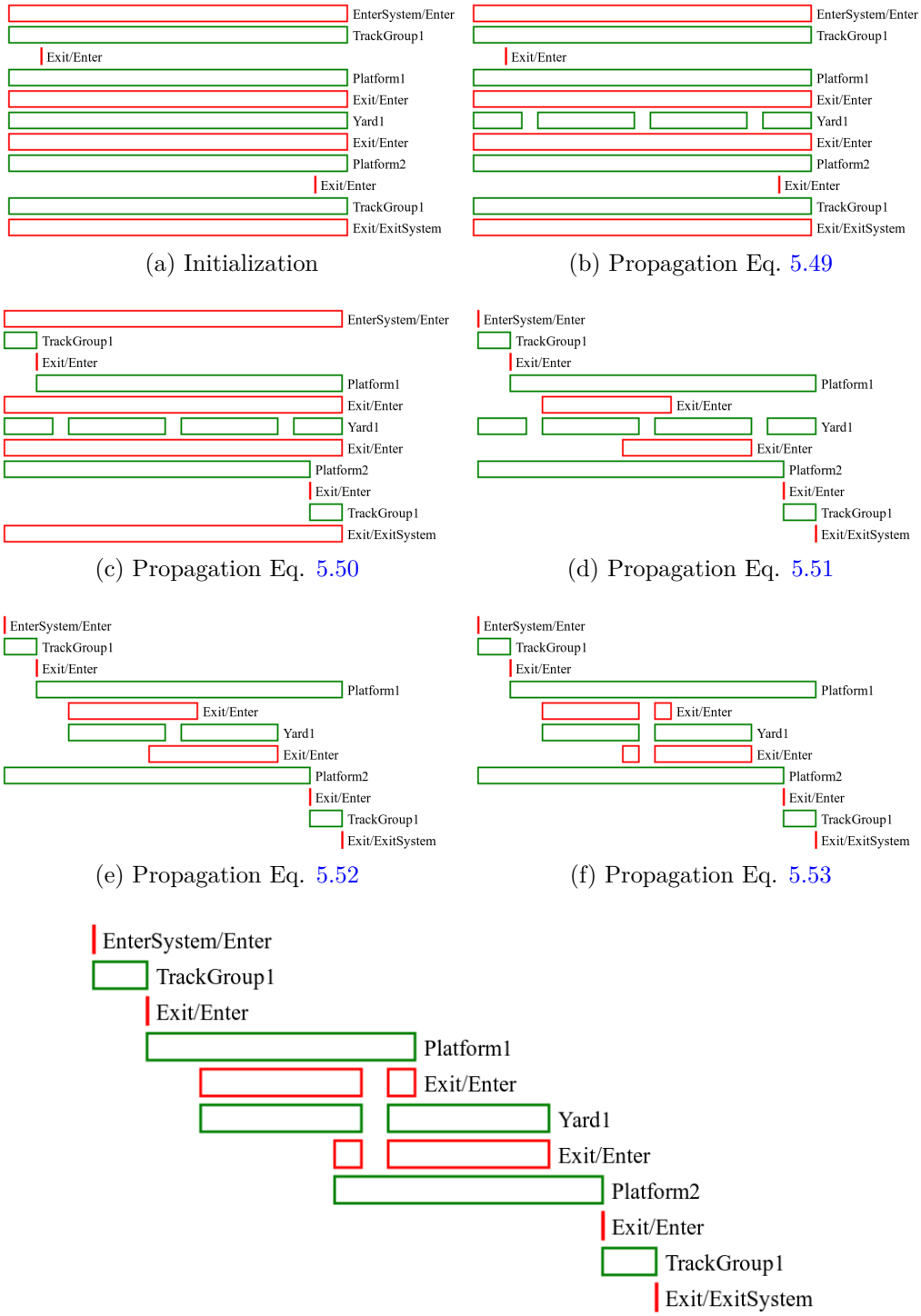
Example of propagation (Fig 5.7)

Figure 5.7 illustrates an example of propagation. Each line represents a multi-interval variable; lines labeled "Exit/Enter" are transition variables, and lines labeled with a resource name are usage variables.

Initially (Figure 5.7a), none of the variables are restricted, except for the transitions corresponding to the arrival and departure (first and last "Exit/Enter"), which are restricted to the exact arrival and departure times. Then, in Figure 5.7b the usage of resource $Yard1$ is restricted according to the imposed consumptions listed in Table 5.2. In Figure 5.7c, usage of resources $Platform1$ and $TrackGroup1$ are restricted according to the preceding "Exit/enter" transition variables: the convoy cannot use a resource before it enters it. Then, the transition variables are restricted to allow sufficient time between the entry and the exit on each resource (Figure 5.7d).

Let us now focus on resource $Yard1$: the last interval of its usage variable doesn't intersect with the following transition variable, so the yard cannot be used during this interval. It is thus removed in Figure 5.7e. The enter and exit transitions are adjusted accordingly in Figure 5.7f, as the convoy may not enter or exit the yard at a given instant if it is not available for use.

Figure 5.7g shows the result once a similar propagation has been performed on $Platform1$ and $Platform2$.



(a) Initialization

(b) Propagation Eq. 5.49

(c) Propagation Eq. 5.50

(d) Propagation Eq. 5.51

(e) Propagation Eq. 5.52

(f) Propagation Eq. 5.53

(g) Constraint Propagation Result

Figure 5.7: Multi-Interval Filtering on the example instance, from $Arr1$ to $Dep1$ (see Table 5.1) with the consumptions indicated in Table 5.2.

5.5 Greedy Assignment Algorithm

The problem is complex due to the routing problem in the station. Resource usage is crucial and we should avoid to engorge one resource of the station. This idea leads to the strategy of the greedy algorithm we present in this section.

We call *immediate arrivals-departure* arrivals immediately followed by a departure on the same platform. In this case, the convoy only has to be routed through the arrival sequence and departure sequence, so the path is set except for the platform, and the convoy occupies few resources.

If a departure is not an immediate departure, the convoy has to be routed to a parking spot and parked between its arrival and its departure. We chose to park on yards only.

Our objective is the maximization of covered arrivals and departures as stated in Section 2. However, to comply with the multi-criteria objective of the original problem and limit penalties, considerations on trains reuses, preferred platform and platform usage penalties are implemented in the solution.

The algorithm is composed of three main steps.

1. First, we try to schedule immediate arrivals-departures while considering train reuses as hard constraints.

2. Then, we relax these constraints and try to schedule immediate arrivals-departures.

3. Finally, we try to schedule remaining trains from the arrival to departure with a parking phase (on yards only) in-between, first considering train reuses as hard constraints, then relaxing these constraints.

To improve coverage of joint departures without performing junctions nor disjunctions, each of these three main assignment phase follows the three steps described below.

(i) Try and assign joint arrivals to joint departures. Both departures and arrivals are taken in decreasing order of convoy size. Between a given joint arrival and a given joint departure, simple dynamic programming allows to compute an assignment maximizing the number of arrivals and departures satisfied, while respecting compatibility and trains order. When at least some members of a joint arrival/departure have been assigned, all the other members are locked and cannot be used in the subsequent assignment steps.

(ii) Try and assign unitary arrivals to unitary departures.

(iii) Try and assign members from remaining joint arrivals to remaining unitary departures (joint or not) and symmetrically remaining unitary arrivals to members of remaining joint departures. When one member is assigned, the other members are locked.

Formally, these sub-steps are described by Alg. 2, using the following definitions:

- $\mathcal{J}^+_{arr}$ (respectively $\mathcal{J}^+_{dep}$) the set of joint arrivals (respectively departures) used in the solution,

$$\mathcal{J}^+_{arr} = \{a \in \mathcal{J}_{arr} | \exists j = jointArr_a, \exists a' \in jaList_j, a' \in \mathcal{A}^+\}$$
$$\mathcal{J}^+_{dep} = \{d \in \mathcal{J}_{dep} | \exists j = jointDep_d, \exists d' \in jdList_j, d' \in \mathcal{D}^+\}$$

- $\mathcal{A}^+$ (respectively $\mathcal{D}^+$) the set of unitary arrivals (arrivals which are not members of any joint arrival) used in the solution,

$$\mathcal{A}^+ = \{a \in \mathcal{A} | arrTrain_a \in \mathcal{T}^+\} \cup \mathcal{J}^+_{arr}$$
$$\mathcal{D}^+ = \{d \in \mathcal{D} | depTrain_d \in \mathcal{T}^+\} \cup \mathcal{J}^+_{dep}$$

- by complement, the sets of joint / unitary arrivals (respectively departures) not used in the solution: $\mathcal{A}^- = \mathcal{A} - \mathcal{A}^+$, $\mathcal{D}^- = \mathcal{D} - \mathcal{D}^+$, $\mathcal{J}^-_{arr} = \mathcal{J}_{arr} - \mathcal{J}^+_{arr}$, $\mathcal{J}^-_{dep} = \mathcal{J}_{dep} - \mathcal{J}^+_{dep}$.

Algorithm 2 Assignment Greedy Algorithm Step

1: $Assign(\mathcal{J}^-_{arr}, \mathcal{J}^-_{dep})$
2: $Assign(\mathcal{A}^- - \mathcal{J}^-_{arr}, \mathcal{D}^- - \mathcal{J}^-_{dep})$
3: $Assign(\mathcal{A}^-, \mathcal{D}^-)$

Non-assigned departures and compatible arrivals are taken in increasing order. Each time, the *Routing* procedure is used to schedule the corresponding convoy. This continues until an arrival with a feasible routing has been found (then the arrival is assigned to the departure), or until there is no more fit arrivals (then the departure remains unsatisfied).

To limit the number of infeasible attempts, arrivals and departures are filtered, taking into account compatibility characteristics such as category and remaining time/distance before maintenance, but also existence of a path between two arrival/departure sequences and time between arrival and departure.

$$arrTime_a + minTrTime_{cat_{T_a}}(P_a, P_d) \leq depTime_d \tag{5.54}$$

The routing algorithm (Alg. 1) is used inside the greedy subroutine *Assign* (Alg. 2). The latter consists in a serie of attempts, iterating on the arrivals and the departures sets. For each couple arrival/departure, a compatible convoy is selected and a resource group-level path (computed during the pre-processing phase) is selected to link the arrival sequence with the departure sequence. Thus, the routing algorithm is applied to this convoy and global path. If the process fails to provide a valid resource-level path (Alg. 1, line 10), we move on to the next compatible couple. Otherwise, the convoy is scheduled along the path, thus the arrival and the departure are respectively removed from the available arrivals and the available departures sets.

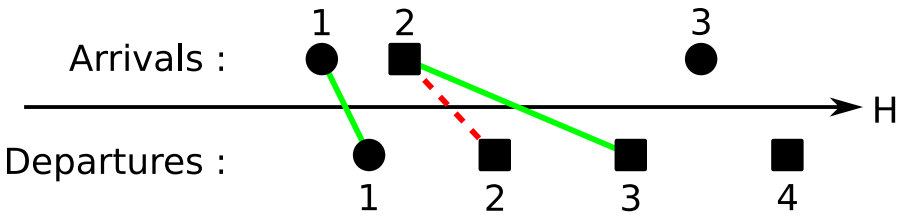


Figure 5.8: Greedy Algorithm : example

Figure 5.8 shows a simplified example on an instance with three arrivals and four departures. A departure is compatible with an arrival if they are represented using the same shape (disc, square) and if the arrival takes place before the departure on the time line. On Figure 5.8, a line from an arrival to a departure is an attempt to route and schedule a convoy through the station from the arrival to the departure. A red dotted line is a failure while a solid green line is a success. Here, the couple $arr1/dep1$ is compatible and is succesfully routed and scheduled. The couple $arr2/dep2$ is compatible but the routing procedure fails, so the next compatible couple $arr2/dep3$ is considered. The routing procedure successfully returns a valid schedule, so $arr2$ and $dep3$ are assigned to each other, and the convoy is routed and scheduled. The last arrival, $arr3$, does not have any compatible departure, so no routing attempt is made.

In order to limit platform assignment penalties, when scheduling a convoy from an arrival to a departure, the preferred platform is tried in priority. To limit the platform usage penalties, we limit the time that a train can spend on a platform. For immediate departures, this amount of time is computed using the ideal dwell times of the departure and the arrival, and the costs defined in parameters, in order to ensure that the platform usage cost for this train does not exceed the $Uncov$ penalty for not covering the arrival. We filter the arrivals verifying this condition. For arrivals and departures separated by a parking phase, we ensure during the routing phase that the ideal dwell times are exactly respected.

5.6 Results and Conclusions

The experiments were performed on a computer featuring an Intel Core 2 Duo E8400 3.00GHz CPU with 4Go RAM, running Xubuntu 64 bits. Table 5.3 presents the results obtained on instance set B, provided by the challenge organizers. The execution was stopped after 10 minutes of running time. Table 5.3 reports the results obtained when stopping after the first two stages of the algorithm (which means allowing only immediate departures), and the results obtained after full execution of the algorithm (which allows parking during the last stage). In each case, we give the number and proportion of covered departures. We also give, for information purposes, the value of the objective function used in the original formulation of the ROADEF Challenge problem (Ramond and Nicolas, 2014), which aggregates various performance costs as well as penalties for uncovered departures and arrivals. However, within our simplified model, the number of covered departures alone is a more pertinent indicator.

	immediate departures only			with parking phase (complete algorithm)		
	t (s)	objective	covered dep	t (s)	objective	covered dep
B1	24	1812600	35% (443/1235)	t/o	1749000	38% (474/1235)
B2	23	1682080	39% (490/1235)	t/o	1656180	40% (503/1235)
B3	21	1711330	38% (472/1235)	t/o	1665330	40% (495/1235)
B4	71	2578160	36% (649/1780)	t/o	2561360	36% (657/1780)
B5	136	3190890	34% (747/2153)	t/o	3155790	35% (764/2153)
B6	72	2584130	36% (645/1780)	t/o	2559330	36% (657/1780)
B7	2	456112	34% (105/304)	68	420112	40% (123/304)
B8	2	473484	31% (96/304)	83	439884	37% (113/304)
B9	72	2978598	29% (590/1967)	t/o	2858098	32% (649/1967)
B10	< 1	364594	18% (36/196)	226	321794	29% (57/196)
B11	3	1984144	16% (190/1122)	t/o	1858544	22% (250/1122)
B12	1	1031864	16% (94/570)	t/o	982164	20% (118/570)

Table 5.3: Summary of the results on set instance B. The first 3 columns present the results for the algorithm limited to immediate departures only, the last 3 when subsequently allowing an intermediate parking phase in yards between arrival and departure. For each case, we give the execution time in seconds, the value of the ROADEF Challenge objective function and the proportion of covered departures. "t/o" means that the execution has reached the timeout, set to 10 minutes.

The solutions provided within the allowed time cover 20% to 40% of departures (and as much arrivals). Of course, this is very few and not suited for a practical application. Although, considering the complexity of the problem and the size of

the search space, it is actually quite reasonable for a simple greedy assignment algorithm. Moreover, given that our approach does not allow maintenance, junction and disjunction operations, the number of departures which can be covered is limited. This suggests that the routing algorithm itself is quite powerful, and that paired with a more advanced assignment optimization algorithm, it could lead to interesting results.

From Table 5.3, one can notice that most covered departures are "immediate departures", which are departures following directly the arrival on the same platform, assigned during the first two stages of the algorithm. The next stage, allowing trains to change platforms between arrival and departure by transiting through yards, gains only a few percent and is time consuming. Thus, considering only immediate departures seems a suitable approach to compute rather quickly an initial solution.

When considering immediate departures, the path and timing is already precisely defined by the arrival and departure sequences. In this case, the *Routing* procedure only serves to determine the feasibility of this path and timing for a given couple arrival/departure and select a platform, given the current state of the solution and avoiding conflict with other trains. When allowing parking, additional decisions must be made, namely the selection of the parking yard and the duration of stay. Thus, in this case the exploration performed by the *Routing* procedure is heavier and more time consuming.

Total execution times appear shorter for instances with few departures and arrivals (B7, B8, B10), as less routing attempts need to be made. The comparatively high execution time for instance B10 might result of the high proportion of joint departures and arrivals in this instance, as members of a joint arrival or departure may be tentatively routed several time, as convoys and as unitary trains.

The most interesting point in the solution exposed is without any doubt the association of the three-level routing algorithm with the greedy algorithm. Indeed, the multi-interval constraint propagation is flexible, efficient and return a set of possible range-value to enter and exit from each resource, allowing to find non-trivial routes within the range of possibilities.

Note that as a canonical path between any pair of resources is computed during the preprocessing phase, trains traveling with a given origin and destination are always routed along the same path. Choosing among a set of preferred paths, or computing on the fly alternative paths forbidding certain resources could be useful to avoid engorgement.

However the solution we propose here is robust, and the optimizer always return a feasible solution. An important aspect of the greedy algorithm is the ability to build a correct solution really fast on all the instances and to build a better solution if more time is available.

A simple optimization process could merely consist of choosing a random train, then removing it from the solution and finally try to build another schedule. Rather than just choosing randomly, one could select the train which generates the most conflicts: each time the algorithm tries and fails to schedule a train, the conflict value of the trains which prevented the scheduling is incremented ; at the end, the train with the highest conflict value is removed from the solution. This could be used as the basis for a local search optimization procedure.

Chapter 6

Conclusion

> I don't need you to agree with me. I'll go away happy with a little bit of doubt. Doubt is good. It's an emotion we can build on. Perhaps if we feed it with curiosity it will blossom into something useful, like suspicion – and action.
>
> Jasper Fforde, *Shades of Grey*

This work presented two takes on problems mixing scheduling and transportation.

In the first (and main) part of this work, we studied, from a theoretical viewpoint, cyclic production of identical parts in robotic cells. This model allows to take into account handling resources in the form of a robotic arm handling parts between machines. Of the two main possible arrangements of the cell, linear and circular, the latter is less understood and appears to be more complex. Our goal was to improve the understanding of the circular configuration, by focusing on the special case of regular balanced cell, where the problem description is particularly compact (3 numbers). Some of the results appeared in the proceedings of the 8th IFAC Conference on Manufacturing Modelling, Management & Control (Thiard et al., 2016).

We considered two classical problems of robotic cells, still open for this configuration: the dominance of the one-part production cycles (1-cycle conjecture) over all production cycles, and finding the best 1-cycle (NP-hard for circular cells, but still open in the balanced case).

We derived the exact expression of cycle time of a crucial 1-cycle specific to circular cells, the odd-even cycle, which was not correctly understood until now, highlighting the specificity of its structure. Lower and upper bounds of its cycle

time can be deduced for the non-balanced case, although in this case the full algebraic expression is yet to determine. In the balanced case, this expression allowed us to delimit parameters regions where the best 1-cycle is one of three classical cycles: the aforementioned odd-even cycle, and two cycles also used in linear cells, the identity cycle and downhill cycle.

By studying simple structural properties of performant cycles in the remaining parameters regions, we were able to completely settle the best 1-cycle problem for small cells (up to 8 machines). We were also able to exhibit a 2-cycle outperforming 1-cycles on a given instance of a 6 machine circular balanced cell, thus proving false the 1-cycle conjecture in this case (interestingly, the same conjecture is on the contrary valid for the *linear* counterpart of this configuration). The 1-cycle conjecture is still open for smaller circular cell: although we did not find, by numeric computation, any 2-cycle outperforming 1-cycles for $3 \leq m \leq 5$ machines, a formal proof of this is still needed, still leaving the question of higher degree cycles.

For larger cells (more than 8 machines), numeric simulations of remaining 1-cycles led us to define and fully study a new, easy to describe, family of cycles formed by specific alterations of the odd-even cycle, whose members outperform the classical cycles in some instances. Studying variations of this structure led us to a conjecture on the dominance of this family, along with classical cycles, over 1-cycles, which we settled for up to 11 machines. If proven valid, this conjecture would allow to settle the best 1-cycle problem on circular regular balanced in polynomial time.

The cyclic structures developed in this work could also be applied to improving approximation for non-balanced circular cell: for instance, the current approximation algorithm uses a combination of the identity and downhill sequences, which is efficient for a mix of short and long tasks; combining the odd-even and downhill structure seems a better fit for a mix of medium and long tasks.

The methods developed and the family of cycles derived in this document could also be extended to other layouts close to the circular one or with similar properties.

The last part of this work is devoted to a practical rolling stock management problem, proposed within the scope of the EURO/ROADEF Challenge 2014. The contribution presented, published in Annals of Operation Research (Joudrier and Thiard, 2017), is a joint work with Hugo Joudrier. The goal is to manage trains within a station; to achieve this, the original formulation mixes several problems and constraints of routing, scheduling and resources assignment. By simplifying the formulation, we were able to propose a flexible routing algorithm based on multi-interval constraint propagation. Coupled with a greedy assignment algorithm, it allows to quickly compute a feasible solution, which can serve as basis for further optimization.

Conclusion (Français)

Dans ce travail, nous avons présenté deux types de problèmes mêlant ordonnancement et transport.

La majeure partie de ce travail était dédiée à une approche théorique de la production cyclique de pièces identiques dans les cellules robotisées. Ce modèle permet de prendre en compte le transport de pièces au sein d'un atelier de production par le biais d'un bras robotisé, chargé de déplacer les pièces d'une machine à l'autre. Parmi les deux principales dispositions physiques de l'atelier, linéaire ou circulaire, la seconde est moins bien comprise et semble plus complexe. Notre objectif était donc d'approfondir la compréhension de la configuration circulaire : pour y parvenir nous nous sommes concentré·e·s sur le cas particulier de cellules régulières et équilibrées, dans lequel la description d'une instance est particulièrement compacte (3 nombres). Certains de nos résultats apparaissent dans les actes de la conférence IFAC MIM 2016 (Thiard et al., 2016).

Nous nous sommes intéressé·e·s à deux problèmes classiques des cellules robotisées, toujours ouverts pour la configuration choisie : la dominance des cycles de production d'une pièce (conjecture des 1-cycles) sur l'ensemble des cycles de production, et la recherche des meilleurs 1-cycles (NP-difficile dans les cellules circulaires, mais de complexité inconnue dans le cas équilibré).

Nous avons développé l'expression algébrique du temps de cycle du cycle pair-impair, un cycle classique crucial, propre aux cellules circulaires et jusqu'ici incorrectement compris, en mettant en évidence sa structure spécifique. Dans le cas non équilibré, nous pouvons en déduire des bornes inférieures et supérieures du temps de cycle, mais l'expression exacte reste à déterminer. Dans le cas équilibré, cette expression nous a permis de préciser des intervalles de paramètres ou le meilleur cycle est l'un de trois cycles classiques : le cycle pair-impair mentionné ci-dessus et deux cycles de base également pertinents dans les cellules linéaires, le cycle identité et le cycle descendant.

Pour les intervalles restants, l'étude de simples propriétés structurelles des cycles performants nous ont permis de fermer complètement le problème du meilleur 1-cycle pour les petites cellules circulaires équilibrées (jusqu'à huit machines). Nous avons également identifié un 2-cycle strictement plus performant que

le meilleur 1-cycles sur certaines instances pour une cellule de 6 machines, réfutant ainsi la conjecture des 1-cycles dans ce cas (à noter que cette conjecture est en revanche valide pour la configuration linéaire correspondante). La conjecture des 1-cycles est toujours ouverte pour de plus petites cellules circulaires : bien que les simulations numériques n'aient mis en évidence aucun 2-cycle plus performant que les 1-cycles pour des cellules de 3 à 5 machines, il manque encore une preuve formelle, ainsi que la question des cycles d'ordre supérieur à deux.

Pour des cellules arbitrairement grandes (plus de huit machines), la simulation des 1-cycles restants nous a mené vers la définition et l'étude complète d'une nouvelle famille de cycles, facile à décrire, dont les membres sont plus performants que les cycles classiques sur certaines instances. L'étude de variations de cette structure nous a conduit à la formulation d'une conjecture sur la dominance de cette famille, associée aux cycles classiques, sur l'ensemble des 1-cycles. La preuve de cette conjecture permettrait de résoudre le problème du meilleur 1-cycle en temps polynomial.

Les structures de cycles développées dans ce travail pourraient également permettre une meilleure approximation pour les cellules ciculaires non-équilibrées : par exemple, l'algorithme actuel utilise une combinaison des séquences montantes (identité) et descendantes, qui est efficace lorsques les tâches sont de durées courtes et longues ; combiner les séquences paire-impaires et descendantes semble une meilleure approche dans le cas ou les taches sont de durées moyennes et longues.

La dernière partie de ce travail était dédiée à un problème pratique de gestion ferroviaire, proposé dans le cadre du Challenge EURO/ROADEF 2014. La contribution présentée, publiée dans Annals of Operation Research (Joudrier and Thiard, 2017), est un travail commun avec Hugo Joudrier. Le but était de gérer les train au sein d'une gare ; à cette fin, la formulation originale mêlait divers problèmes et contraintes de routage, ordonnancement et affectation de ressources. En simplifiant la formulation, nous avons pu proposer un algorithme de routage souple, basé sur de la propagation de contraintes multi-intervalles. Associé à un algorithme d'affectation glouton, celui-ci permet de calculer une première solution réalisable qui peut servir de base à une optimisation.

Notations

These are the notations used throughout Chapters 1 to 4 (robotic cells). For each notation, we give the page where it is first introduced and defined.

Chapter 5 is independent and uses its own set of notations (please refer to the chapter itself or to Ramond and Nicolas (2014)[1]).

A_i for $i \in \{0 \dots m\}$, activity A_i, i.e. the elementary sequence of moves where the robot unloads a part from machine M_i, travels to machine M_{i+1} and loads the part on to machine M_{i+1}. 5

α $\left\lfloor \frac{m+1}{2} \right\rfloor$, where m is the number of machine. 30

$\alpha(n)$ for any $n \in \mathcal{N}$, $\left\lfloor \frac{n+1}{2} \right\rfloor$. 66

c denotes a generic cycle. 20

δ the travel time between any two consecutive machines M_i and M_{i+1} (both ways) with $i \in \{0, \dots, m\}$. 20

Δ_c the total travel time of the robot during one iteration of the specified cycle c. 24

$d_i(\pi)$ the travel time of the robot between the loading of machine M_i and its subsequent unloading (in a cyclic sense) for the specified 1-cycle π. 24

$d_{i,l}(c)$ the travel time of the robot between the l-th loading of machine M_i and its subsequent unloading (in a cyclic sense) for the specified cycle c. 24

$d_{min}(c)$ minimum value of $d_{i,l}$: $\min_{i,l} d_{i,l}(c)$, or for a 1-cycle, $\min_{i \in \{1,\dots,m\}} d_i(c)$. 24

m the number of machines. 20

[1] https://hal.archives-ouvertes.fr/hal-01057324/file/Challenge_sujet_phaseFinale_140224.pdf

M_0 the input buffer. 20

M_{m+1} the output buffer. 20

$M_1 \dots M_m$ the m machines. 20

$m(i)$ the number of times the robot travels between machines M_i and M_{i+1} in either direction (forward or backward). 58

$m^-(i)$ the number of times the robot travels backward between machines M_i and M_{i+1}. 58

$m^+(i)$ the number of times the robot travels forward between machines M_i and M_{i+1} during one iteration. 58

p_i for $i \in \{1, \dots, m\}$, the processing time on machine M_i (in the non balanced case). 20

p the processing time on any machine $M_1 \dots M_m$ (in the balanced case). 20

p^* the value of parameter p for which π_{dh} and π_{oe} have the same cycle time ($T_{\pi_{oe}}(p^*) = T_{\pi_{dh}}(p^*) = 3(m+1)\delta$. Its expression is $p^* = \frac{3\alpha-1}{2\alpha-1}(m+1)\delta$. 41

p_n^* for n even, the value of parameter p for which π_{nw} and π_{dh} have the same cycle time ($T_{\pi_{nw}}(p^*) = T_{\pi_{dh}}(p^*) = 3(m+1)\delta$). Its expression is $p^* = (m+2n+1)\delta + \frac{\alpha(m-4n)}{2\alpha(m-4n)-1}(m+1-4n)\delta$. 75

π denotes a 1-cycle. 20

π_{dh} the downhill cycle $A_0 A_m A_{m-1} ... A_1$. 28

π_{id} the identity cycle $A_0 A_1 \dots A_m$. 27

π_{oe} the odd-even cycle $A_0 A_2 A_4 \dots A_1 A_3 A_5 \dots$. 29

π_{2w} the cycle defined by $\pi_{2w} = v(3,7)$, which expands to $\pi_{nw} = A_0 A_3 A_2 A_5 A_8 \dots A_{2i} \dots A_1 A_4 A_7 A_6 A_9 \dots A_{2i+1}$; special case of π_{nw}. 69

π_{nw} for $n \in \mathcal{N}$, the cycle defined by $\pi_{nw} = v(3, 7, \dots, 4i-1, \dots, 4n-1)$; generalization of π_{2w}. 73

$\mathrm{SE}_{i,j}$ the ordered sequence of all activities A_l with l *even*, $i \le l \le j$. 66

$\mathrm{SO}_{i,j}$ the ordered sequence of all activities A_l with l *odd*, $i \le l \le j$. 66

T_c the long run average cycle time of the specified cycle c. When useful, the cycle time may be expressed as a function of p and denoted $T_c(p)$. If c is a k-cycle, its cycle length is given by $\frac{T_c}{k}$ (obviously, the cycle length of a 1-cycle π is simply T_π). 20

$v(i_1, i_2, \ldots, i_n)$ the cycle defined by the n waves of size 1: $A_{i_1}A_{i_1-1}$, $A_{i_2}A_{i_2-1}$ $\ldots A_{i_n}A_{i_n-1}$. 65

$V_h(i_1, i_2, \ldots, i_n)$ the set of cycles containing the n waves of size h: $A_{i_1}A_{i_1-h}, A_{i_2}A_{i_2-h}, \ldots, A_{i_n}A_{i_n-h}$. 65

$W_c(p)$ the long run average total waiting time of the robot for the specified cycle c, depending on p. The cycle time of c can be decomposed as $T_c(p) = \Delta_c(p) + W_c(p)$. 65

$\hat{W}(n, d, p)$ for any $n > 0$, $p \geq 0$, $d \geq 0$, $\hat{W}(n, d, p) = \frac{2\alpha(n)-1}{\alpha(n)} \max(0, p - d)$. By convention, $\hat{W}(0, d, p) = 0$. 66

w_i the waiting time of the robot at machine M_i of the cycle. 71

w_i^j the waiting time of the robot at machine M_j during the i-the iteration of the cycle (when several consecutive iterations are considered).. 31

List of Figures

List of Tables

Bibliography

Agnetis, A. (2000). Scheduling no-wait robotic cells with two and three machines. *European Journal of Operational Research*, 123(2):303–314. 8, 9, 10, 14

Agnetis, A. and Pacciarelli, D. (2000). Part sequencing in three machine no-wait robotic cells. *Operations Research Letters*, 27:185–192. 9

Akturk, M., Gultekin, H., and Karasan, O. (2005). Robotic cell scheduling with operational flexibility. *Discrete Applied Mathematics*, 145(3):334–348. 18

Aneja, Y. P. and Kamoun, H. (1999). Scheduling of parts and robot activities in a two machine robotic cell. *Computers & Operations Research*, 26(4):297–312. 12

Asfahl, C. R. (1985). *Robots and manufacturing automation*. John Wiley & Sons, New York, NY. 2

Bagchi, T. P., Gupta, J. N., and Sriskandarajah, C. (2006). A review of tsp-based approaches for flowshop scheduling. *European Journal of Operational Research*, 169(3):816–854. 7

Batur, G. D., Karasan, O. E., and Akturk, M. S. (2012). Multiple part-type scheduling in flexible robotic cells. *International Journal Of Production Economics*, 135(2):726–740. 19

Benhamou, F. and Granvilliers, L. (2006). Continuous and interval constraints. *Foundations of Artificial Intelligence*, 2:571–603. 103

Brauner, N. (1999). *Ordonnancement dans des cellules robotisées*. Thèse de doctorat, Université Joseph Fourier, Grenoble, France. 15, 20, 32, 33, 42

Brauner, N. (2008). Identical part production in cyclic robotic cells: Concepts, overview and open questions. *Discrete Applied Mathematics*, 156(13):2480–2492. SWITZERLAND, AUG, 2006. 3, 7, 14, 15, 20, 35, 45

Brauner, N. and Finke, G. (1999). On a conjecture about robotic cells: new simplified proof for the three-machine case. *Journal of Information Systems and Operational Research - INFOR: Scheduling in Computer and Manufacturing Systems*, 37(1):20–36. 14

Brauner, N. and Finke, G. (2001). Cycles and permutations in robotic cells. *Mathematical and Computer Modelling*, 34(5-6):565–591. 14

Brauner, N., Finke, G., and Kubiak, W. (2003). Complexity of one-cycle robotic flow-shops. *Journal of Scheduling*, 6(4):355–371. 13

Brauner, N. and G.Finke (2005). A note on agnetis' conjecture for classical robotic cells. Technical Report 120, Les cahiers du laboratoire Leibniz, Grenoble, France. 14

Buljubašić, M., Vasquez, M., and Gavranović, H. (2017). Two-phase heuristic for sncf rolling stock problem. *Annals of Operations Research*. 91

Carlier, J., Haouari, M., Kharbeche, M., and Moukrim, A. (2010). An optimization-based heuristic for the robotic cell problem. *European Journal of Operational Research*, 202(3):636–645. 5, 13

Catusse, N. and Cambazard, H. (2014). Roadef Challenge 2014: A Modeling Approach, Rolling stock unit management on railway sites. working paper or preprint. 91

Chabert, G. and Jaulin, L. (2009). Contractor programming. *Artificial Intelligence*, 173(11):1079–1100. 104

Che, A., Chabrol, M., Gourgand, M., and Wang, Y. (2012). Scheduling multiple robots in a no-wait re-entrant robotic flowshop. *International Journal Of Production Economics*, 135(1):199–208. 11

Che, A. and Chu, C. (2005). Multi-degree cyclic scheduling of two robots in a no-wait flowshop. *IEEE Transactions on Automation Science and Engineering*, 2(2):173–183. 11

Che, A. and Chu, C. (2008). Optimal scheduling of material handling devices in a PCB production line: problem formulation and a polynomial algorithm. *Mathematical Problems in Engineering*, 2008. 11

Che, A. and Chu, C. (2009). Multi-degree cyclic scheduling of a no-wait robotic cell with multiple robots. *European Journal of Operational Research*, 199(1):77–88. 11

Che, A., Chu, C., and Levner, E. (2003). A polynomial algorithm for 2-degree cyclic robot scheduling. *European Journal of Operational Research*, 145(1):31–44. 9

Che, A., Hu, H., Chabrol, M., and Gourgand, M. (2011). A polynomial algorithm for multi-robot 2-cyclic scheduling in a no-wait robotic cell. *Computers & Operations Research*, 38(9):1275–1285. 11

Chen, H., Chu, C., and Proth, J.-M. (1997). Sequencing of parts in robotic cells. *International Journal of Flexible Manufacturing Systems*, 9(81-104). 12

Chu, C. (2006). A faster polynomial algorithm for 2-cyclic robotic scheduling. *Journal of Scheduling*, 9(5):453–468. 9

Crama, Y., Kats, V., van de Klundert, J., and Levner, E. (2000). Cyclic scheduling in robotic flowshops. *Annals of Operations Research: Mathematics of Industrial Systems*, 96(1):97–124. 7

Crama, Y. and van de Klundert, J. (1997a). Cyclic scheduling of identical parts in a robotic cell. *Operations Research*, 45(6):952–965. 5, 14, 15, 19, 24, 29, 49

Crama, Y. and van de Klundert, J. (1997b). Robotic flowshop scheduling is strongly NP-complete. In W.K. Klein Haneveld, O. V. and Kallenberg, L., editors, *Ten Years LNMB*, pages 277–286, CWI Tract 122, Amsterdam, The Netherlands. 3, 7

Crama, Y. and van de Klundert, J. (1999). Cyclic scheduling in 3-machine robotic flow shops. *Journal of Scheduling*, 2:35–54. 6, 14

Dawande, M., Geismar, H., and Sethi, S. (2005a). Dominance of cyclic solutions and challenges in the scheduling of robotic cells. *SIAM Review*, 47(4):709–721. 5

Dawande, M., Geismar, H. N., Sethi, S. P., and Sriskandarajah, C. (2005b). Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling*, 8(5):387–426. 4

Dawande, M., Pinedo, M., and Sriskandarajah, C. (2008). Multiple part-type production in robotic cells: Equivalence of two real-world models. *Manufacturing and Service Operations Management*. 17

Dawande, M., Sriskandarajah, C., and Sethi, S. (2002). On throughput maximization in constant travel-time robotic cells. *Manufacturing and Service Operations Management*, 4(4):296–312. 5, 14, 24

Dawande, M. W., Geismar, H. N., Sethi, S. P., and Sriskandarajah, C. (2007). *Throughput Optimization in Robotic Cells*. Springer. 6, 7, 13

Drobouchevitch, I., Sethi, S., Sidney, J., and Sriskandarajah, C. (2004). A note on scheduling multiple parts in two-machine dual gripper robotic cell: Heuristic algorithm and performance guarantee. *International Journal of Operations and Quantitative Management*, 10(4):297–314. 16

Drobouchevitch, I. G., Geismar, H. N., and Sriskandarajah, C. (2010). Throughput optimization in robotic cells with input and output machine buffers: A comparative study of two key models. *European Journal of Operational Research*, 206(3):623–633. 16

Drobouchevitch, I. G., Sethi, S. P., and Sriskandarajah, C. (2006). Scheduling dual gripper robotic cell: One-unit cycles. *European Journal of Operational Research*, 171(2):598–631. 16

Elmi, A. and Topaloglu, S. (2014). Scheduling multiple parts in hybrid flow shop robotic cells served by a single robot. *International Journal of Computer Integrated Manufacturing*, 27(12):1144–1159. 8

Elmi, A. and Topaloglu, S. (2016). Multi-degree cyclic flow shop robotic cell scheduling problem: Ant colony optimization. *Computers & Operations Research*, 73:67–83. 13

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345. 102

Foumani, M., Gunawan, I., and Ibrahim, Y. (2014). Scheduling rotationally arranged robotic cells served by a multi-function robot. *International Journal of Production Research*, 52(13):4037–4058. 16

Foumani, M. and Jenab, K. (2013). Analysis of flexible robotic cells with improved pure cycle. *International Journal Of Computer Integrated Manufacturing*, 26(3):201–215. 19

Garey, M. R., Johnson, D. S., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129. 1

Geiger, M. J., Huber, S., Langton, S., Leschik, M., Lindorf, C., and Tüshaus, U. (2017). Multi-attribute assignment of trains to departures in rolling stock management. *Annals of Operations Research*. 91

Geismar, H., Sethi, S., Sidney, J., and Sriskandarajah, C. (2005a). A note on productivity gains in flexible robotic cells. *International Journal of Flexible Manufacturing Systems*, 17(1):5–21. 15, 18, 26

Geismar, H. N., Chan, L. M. A., Dawande, M., and Sriskandarajah, C. (2008). Approximations to optimal k-unit cycles for single-gripper and dual-gripper robotic cells. *Production and Operations Management*, 17(5):551–563. 14, 16

Geismar, H. N., Dawande, M., and Sriskandarajah, C. (2004). Robotic cells with parallel machines: Throughput maximization in constant travel-time cells. *Journal of Scheduling*, 7(5):375–395. 8

Geismar, H. N., Dawande, M., and Sriskandarajah, C. (2005b). Approximation algorithms for k-unit cyclic solutions in robotic cells. *European Journal of Operational Research*, 162(2):291–309. 13, 14

Geismar, H. N., Dawande, M., and Sriskandarajah, C. (2006). Throughput optimization in constant travel-time dual gripper robotic cells with parallel machines. *Production and Operations Management*, 15(2):311–328. 16

Geismar, H. N., Dawande, M., and Sriskandarajah, C. (2007). A (10/7)-approximation algorithm for an optimum cyclic solution in additive travel-time robotic cells. *IIE Transactions*, 39(2):217–227. 14

Geismar, N., Manoj, U. V., Sethi, A., and Sriskandarajah, C. (2012). Scheduling robotic cells served by a dual-arm robot. *IIE Transactions*, 44(3):230–248. 17

Gilmore, P. C. and Gomory, R. E. (1964). Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. *Operations Research*, 12(5):655–679. 1, 8, 12

Gultekin, H., Akturk, M., and Karasan, O. (2006). Cyclic scheduling of a robotic cell with tooling constraints. *European Journal Operational Research*, 174(2):777–796. 18

Gultekin, H., Akturk, M. S., and Karasan, O. E. (2007). Scheduling in a three-machine robotic flexible manufacturing cell. *Computers & Operations Research*, 34(8):2463–2477. 18

Gultekin, H., Akturk, M. S., and Karasan, O. E. (2008). Scheduling in robotic cells: process flexibility and cell layout. *International Journal Of Production Research*, 46(8):2105–2121. 18, 19

Gultekin, H., Akturk, M. S., and Karasan, O. E. (2010). Bicriteria robotic operation allocation in a flexible manufacturing cell. *Computers & Operations Research*, 37(4):779–789. 19

Gultekin, H., Dalgıç, Ö. O., and Akturk, M. S. (2017). Pure cycles in two-machine dual-gripper robotic cells. *Robotics and Computer-Integrated Manufacturing*, 48:121–131. 18

Gultekin, H., Karasan, O. E., and Akturk, M. S. (2009). Pure cycles in flexible robotic cells. *Computers & Operations Research*, 36(2):329–343. Scheduling for Modern Manufacturing, Logistics, and Supply Chains. 18, 19

Gundogdu, E. and Gultekin, H. (2016). Scheduling in two-machine robotic cells with a self-buffered robot. *IIE Transactions*, 48(2):170–191. 17

Hall, N. G., Kamoun, H., and Sriskandarajah, C. (1997). Scheduling in robotic cells: Classification, two and three machine cells. *Operations Research*, 45(3):421–439. 4, 12, 13, 14

Hall, N. G., Kamoun, H., and Sriskandarajah, C. (1998). Scheduling in robotic cells: Complexity and steady state analysis. *European Journal of Operational Research*, 109(1):43–65. 12

Hall, N. G. and Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 44(3):510–525. 1, 3

Hudak, P., Hughes, J., Peyton Jones, S., and Wadler, P. (2007). A history of haskell: being lazy with class. In *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, pages 12–1. ACM. 105

Hurink, J. and Knust, S. (2001). Makespan minimization for flow-shop problems with transportation times and a single robot. *Discrete Applied Mathematics*, 112(1-3):199–216. 5

ILOG, S. (1999). Revising hull and box consistency. In *Logic Programming: Proceedings of the 1999 International Conference on Logic Programming*, page 230. MIT press. 103

Jiang, Y. and Liu, J. (2007). Multihoist cyclic scheduling with fixed processing and transfer times. *IEEE Transactions on Automation Science and Engineering*, 4(3):435–450. 11

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68. 1

Johnsson, T. (1984). Efficient compilation of lazy evaluation. *SIGPLAN Not.*, 19(6):58–69. 105

Jolai, F., Foumani, M., Tavakoli-Moghadam, R., and Fattahi, P. (2012). Cyclic scheduling of a robotic flexible cell with load lock and swap. *Journal of Intelligent Manufacturing*, 23(5):1885–1891. 17, 19

Joudrier, H. and Thiard, F. (2017). A greedy approach for a rolling stock management problem using multi-interval constraint propagation. *Annals of Operations Research*. 91, 118, 120

Jung, K. S., Geismar, H. N., Pinedo, M., and Sriskandarajah, C. (2015). Approximations to optimal sequences in single-gripper and dual-gripper robotic cells with circular layouts. *IIE Transactions*, 47(6):634–652. Accepted manuscript. 5, 15, 17

Kamalabadi, I., Gholami, S., and Mirzaei, A. (2008). A new solution for the cyclic multiple-part type three-machine robotic cell problem based on the particle swarm meta-heuristic. *Journal of Industrial and Systems Engineering*, 1(4):304–317. 12

Kamoun, H., Hall, N. G., and Sriskandarajah, C. (1999). Scheduling in robotic cells: Heuristics and cell design. *Operations Research*, 47(6):821–835. 12, 13

Kats, V., Lei, L., and Levner, E. (2008). Minimizing the cycle time of multiple-product processing networks with a fixed operation sequence, setups, and time-window constraints. *European Journal of Operational Research*, 187(3):1196–1211. 7

Kats, V. and Levner, E. (1997). A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling. *Operations Research Letters*, 21(4):171–179. 9, 10

Kats, V. and Levner, E. (2002). Cycle scheduling in a robotic production line. *Journal of Scheduling*, 5(1):23–41. 10

Kats, V. and Levner, E. (2009). A polynomial algorithm for 2-cyclic robotic scheduling: A non-Euclidean case. *Discrete Applied Mathematics*, 157(2):339–355. 10

Kats, V., Levner, E., and Meyzin, L. (1999). Multiple-part cyclic hoist scheduling using a sieve method. *IEEE Transactions on Robotics and Automation*, 15(4). 10

Kharbeche, M., Carlier, J., Haouari, M., and Moukrim, A. (2011). Exact methods for the robotic cell problem. *Flexible Services and Manufacturing Journal*, 23(2):242–261. 13

Kise, H. (1991). On an automated two-machine flowshop scheduling problem with infinite buffer. *Journal of the Operations Research Society of Japan*, 34(3):354–361. 5

Knüppel, O. (1994). Profil/bias—a fast interval library. *Computing*, 53(3-4):277–287. 104

Lehoux-Lebacque, V. (2007). *Théories et applications en ordonnancement : contraintes de ressources et tâches agrégées en catégories*. PhD thesis, Université Joseph Fourier. Thèse de doctorat dirigée par Finke, Gerd et Brauner, Nadia Informatique Université Joseph Fourier (Grenoble) 2007. 4, 20

Lerch, M., Tischler, G., Gudenberg, J. W. V., Hofschuster, W., and Krämer, W. (2006). Filib++, a fast interval library supporting containment computations. *ACM Transactions on Mathematical Software (TOMS)*, 32(2):299–324. 104

Leung, J., Kelly, L., and Anderson, J. H. (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Inc., Boca Raton, FL, USA. 1

Leung, J. M. and Levner, E. (2006). An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times. *Operations Research Letters*, 34(4):465–472. 11

Levner, E., Kats, V., Alcaide Lopez de Pablo, D., and Cheng, T. C. E. (2010). Complexity of cyclic scheduling problems: A state-of-the-art survey. *Computers & Industrial Engineering*, 59(2):352–361. 7

Levner, E., Kats, V., and Levit, V. E. (1997). An improved algorithm for cyclic flowshop scheduling in a robotic cell. *European Journal of Operational Research*, 97(3):500–508. 8, 9

Levner, E., Kats, V., and Sriskandarajah, C. (1996). A geometric algorithm for finding two-unit cyclic schedules in no-wait robotic flowshop. In *Proceedings of the International Workshop in Intelligent Scheduling of Robots and FMS, WISOR*, volume 96, pages 101–112. 9

Liu, J. and Jiang, Y. (2005). An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem. *Operations Research*, 53(2):313–327. 11

Logendran, R. and Sriskandarajah, C. (1996). Sequencing of robot activities and parts in two-machine robotic cells. *International Journal of Production Research*, 34(12):3447–3463. 4

Madsen, A. L. and Jensen, F. V. (1999). Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1):203–245. 105

Majumder, A. and Laha, D. (2016). A new cuckoo search algorithm for 2-machine robotic cell scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation*, 28:131–143. 4

Mangione, F. (2003). *Ordonnancement des ateliers de traitement de surface pour une production cyclique et mono-produit*. PhD thesis, Institut National Polytechnique de Grenoble. 10

Mangione, F., Brauner, N., and Penz, B. (2003). Optimal cycles for the robotic balanced no-wait flow shop. In *Proceedings IEPM'03, International Conference of Industrial Engineering and Production Management*, volume 2, pages 539–547, Porto, Portugal. 10

Middendorf, M. and Timkovsky, V. (2002). On scheduling cycle shops: classification, complexity and approximation. *Journal Of Scheduling*, 5(2):135–169. 12

Moore, R. E. (1966). *Interval Analysis*. Prentice-Hall series in automatic computation. Prentice-Hall. 104

Ninin, J. (2015). Global optimization based on contractor programming: an overview of the ibex library. In *International Conference on Mathematical Aspects of Computer and Information Sciences*, pages 555–559. Springer. 104

Papadimitriou, C. H. and Kanellakis, P. C. (1980). Flowshop scheduling with limited temporary storage. *Journal of the ACM*, 27(3):533–549. 1

Pavlov, S. V. (2013). On optimal cycles for regular balanced no-wait robotic cell problems. *Diskretn. Anal. Issled. Oper.*, 20(1):45–57. 10

Phillips, L. W. and Unger, P. S. (1976). Mathematical programming solution of a hoist scheduling program. *A I I E Transactions*, 8(2):219–225. 7

Pinedo, M. L. (2008). *Scheduling Theory, Algorithms, and Systems*. Springer. 1

Rajapakshe, T., Dawande, M., and Sriskandarajah, C. (2011). Quantifying the impact of layout on productivity: An analysis from robotic-cell manufacturing. *Operations Research*, 59(2):440–454. 5, 15, 17, 19, 26, 49

Ramond, F. and Nicolas, M. (2014). Trains don't vanish ! ROADEF EURO 2014 Challenge Problem Description. https://hal.archives-ouvertes.fr/hal-01057324/. 91, 92, 94, 100, 102, 113, 123

Röck, H. (1984). The three-machine no-wait flow shop is np-complete. *Journal of the ACM*, 31(2):336–345. 8

Rogers, D. F., Plante, R. D., Wong, R. T., and Evans, J. R. (1991). Aggregation and disaggregation techniques and methodology in optimization. *Operations Research*, 39(4):553–582. 102

Sethi, S., Sidney, J., and Sriskandarajah, C. (2001). Scheduling in dual gripper robotic cells for productivity gains. *IEEE Transactions on Robotics and Automation*, 17(3):423–341. 16

Sethi, S. P., Sriskandarajah, C., Sorger, G., Blazewicz, J., and Kubiak, W. (1992). Sequencing of parts and robot moves in a robotic cell. *International Journal of Flexible Manufacturing Systems*, 4:331–358. 2, 4, 5, 6, 8, 12, 15, 26, 35

Song, W., Zabinsky, Z. B., and Storch, R. L. (1993). An algorithm for scheduling a chemical processing tank line. *Production Planning and Control*, 4(4):323–332. 8

Soukhal, A. (2001). *Ordonnancement simultané des moyens de transformation et de transport*. PhD thesis, Université François Rabelais. Thèse de doctorat dirigée par Proust, Christian Informatique Tours 2001. 8

Soukhal, A. and Martineau, P. (2005). Resolution of a scheduling problem in a flowshop robotic cell. *European Journal Of Operational Research*, 161(1):62–72. 5, 13

Sriskandarajah, C., Drobouchevitch, I., Sethi, S. P., and Chandrasekaran, R. (2004). Scheduling multiple parts in a robotic cell served by a dual-gripper robot. *Operations Research*, 52(1):65–82. 9, 16

Sriskandarajah, C., Hall, N., and Kamoun, H. (1998). Scheduling large robotic cells without buffers. *Annals of Operations Research*, 76(0):287–321. 13

Steiner, G. and Xue, Z. (2005). Scheduling in reentrant robotic cells: Algorithms and complexity. *Journal Of Scheduling*, 8(1):25–48. 12

Thiard, F., Catusse, N., and Brauner, N. (2016). Good Production Cycles for Circular Robotic Cells. In *8th IFAC Conference on Manufacturing Modelling, Management & Control (MIM 2016)*, Troyes, France. 117, 119

Yang, Y., Chen, Y., and Long, C. (2016). Flexible robotic manufacturing cell scheduling problem with multiple robots. *International Journal of Production Research*, 0(0):1–14. Published online. 5

Yildiz, S., Karasan, O. E., and Akturk, M. S. (2012). An analysis of cyclic scheduling problems in robot centered cells. *Computers & Operations Research*, 39(6):1290–1299. Special Issue on Scheduling in Manufacturing Systems. 19

Zabihzadeh, S. S. and Rezaeian, J. (2016). Two meta-heuristic algorithms for flexible flow shop scheduling problem with robotic transportation and release time. *Applied Soft Computing*, 40:319–330. 8

Zahrouni, W. and Kamoun, H. (2012). Sequencing and scheduling in a three-machine robotic cell. *International Journal of Production Research*, 50(10):2823–2835. 12

Zarandi, M. H. F., Mosadegh, H., and Fattahi, M. (2013). Two-machine robotic cell scheduling problem with sequence-dependent setup times. *Computers & Operations Research*, 40(5):1420–1434. 4

Zhou, Z., Che, A., and Yan, P. (2012). A mixed integer programming approach for multi-cyclic robotic flowshop scheduling with time window constraints. *Applied Mathematical Modelling*, 36(8):3621–3629. 7

Résumé La première partie de ce travail concerne la production cyclique pour l'optimisation du taux de production dans les flowshops robotisés, où un robot est chargé du transport des pièces. Les cellules robotisées peuvent être disposées de façon linéaire ou circulaire. Les principaux résultats théoriques concernant la disposition linéaire ne peuvent être étendus à la configuration circulaire. En particulier, trouver le meilleur cycle de production de une pièce (1-cycle) est un problème polynomial dans le cas des cellules linéaires additives, mais NP-difficile pour la configuration correspondante circulaire.

Nous nous concentrons principalement sur le cas des cellules circulaires équilibrées, où le temps d'usinage est identique sur toutes les machines. Après avoir présenté des outils pour l'analyse cyclique dans les cellules circulaires, nous établissons des propriétés nécessaires des 1-cycles performants, ce qui permet de conclure sur le problème du meilleur 1-cycle jusqu'à 8 machines. Toutefois, nous fournissons un contre-exemple pour 6 machines à la conjecture classique des 1-cycles, toujours ouverte dans cette configuration.

Ensuite, nous étudions la structure des 1-cycles performants pour des cellules circulaires équilibrées arbitrairement grandes. Nous définissons et étudions les propriétés d'une nouvelle famille de cycles basée sur cette structure et formulons une conjecture sur sa dominance sur les 1-cycles qui conduirait à un algorithme polynomial pour le problème du meilleur 1-cycle dans ce cas. Cette structure permet de déterminer le meilleur 1-cycle jusqu'à 11 machines.

Dans la deuxième partie, nous présentons le travail réalisé sur un problème industriel proposé par la SNCF dans le cadre du challenge ROADEF/EURO. Nous proposons un algorithme glouton pour ce problème combinant divers aspects de la gestion des trains au sein d'une gare.

Mots-clefs Ordonnancement, Flow-shop, Cellule robotisée, Ressource de transport, Production cyclique, Gestion de matériel ferroviaire

Abstract The first part of this work deals with cyclic production for throughput optimization in robotic flow-shops, where a robot is in charge of the material handling of parts. Robotic cells may have a linear or a circular layout. Most theoretical results for the linear layout do not hold for the circular layout. In particular, the problem of finding the best one part production cycle (1-cycle), which is a polynomial problem for linear additive cells, has been proved NP-hard for the corresponding circular configuration.

We mainly focus on a special case of circular balanced cells, where the processing times are identical for all machines. After presenting tools for cyclic analysis in circular cells, we study necessary properties of efficient 1-cycles. These results allow to conclude on the best one part production cycle for any parameters in circular balanced cells up to 8 machines. However, we provide a counter-example to the classical 1-cycle conjecture, still open for this configuration.

Then, we study the structure of efficient one part production cycles in arbitrarily large circular balanced cells. We introduce and study a new family of cycles based on this structure, and formulate a conjecture on its dominance over one part-production cycles, which would lead to a polynomial algorithm for finding the best 1-cycle for circular balanced cells. This structure allows to settle the best one part production cycle for cells with up to 11 machines.

In a second part, we present work on an industrial problem of railway stock scheduling proposed by the French railway company in the context of the ROADEF/EURO competition. We propose a greedy algorithm for this problem combining the various aspects of trains handling inside a station.

Keywords Scheduling, Flowshop, Robotic cell, Material Handling System, Cyclic Production, Circular Layout, Railway Stock Management