N° d'ordre : 02/2015-D/M.T

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE « HOUARI BOUMEDIENE »
FACULTE DE MATHEMATIQUES

THESE

Présentée pour l'obtention du diplôme de DOCTORAT EN SCIENCES

EN : MATHEMATIQUES

Spécialité : Recherche Opérationnelle : Génie Mathématiques

Par : LABBI Wafaa

Sujet

ORDONNANCEMENT SOUS CONTRAINTES DE PREPARATION

Soutenu publiquement le 04 Mars 2015, devant le jury composé de :

M. BELBACHIR Hacène,	Professeur à l'USTHB,	Président
M. BOUDHAR Mourad,	Professeur à l'USTHB,	Directeur de thèse
M. OULAMARA Ammar,	Professeur à l'U. de Lorraine (France),	Co-Directeur de thèse
M. AIT HADDADENE Hacene,	Professeur à l'USTHB,	Examinateur
M. CHERGUI Mohamed El-Amine,	Maître de Conférences/A à l'USTHB,	Examinateur
M. REBAINE Djamel,	Professeur à l'UQAC (Canada),	Examinateur

Remerciements

Tous mes remerciements à "ALLAH" le tout puissant.

Je tiens à témoigner ma profonde gratitude et ma reconnaissance à Monsieur Mourad BOUDHAR, mon directeur de thèse, Professeur à l'U.S.T.H.B., sans qui ce travail n'aurait pas vu le jour, de m'avoir proposée ce thème et de m'avoir encadrée, pour sa rigueur, son expérience, ses précieuses orientations et ses conseils avisés qui m'ont guidé tout au long du parcours de cette thèse. Je le remercie également de m'avoir aidée à mieux comprendre les problèmes d'ordonnancement et de m'avoir intégrée dans son équipe de recherche. Je le remercie aussi pour sa présence et sa patience durant toutes ses années.

J'exprime ensuite mes plus vifs remerciements à Monsieur Ammar OULAMARA, Professeur à l'université de Lorraine, qui fut pour moi un Co-directeur de thèse attentif et disponible malgré ses nombreuses charges. Je le remercie d'avoir participé à la proposition de ce thème, d'avoir accepté de co-encadrer cette thèse et de suivre sa progression. Je tiens également à le remercier de m'avoir embarquée sur des pistes de travail intéréssantes, pour son invitation et son accueil à l'université de Lorraine durant mon séjour scientifique.

Mes sincères remerciements s'adressent ensuite à l'ensemble des membres de mon jury :

Monsieur Hacène BELBACHIR, Professeur à l'U.S.T.H.B., d'avoir accepté de présider ce jury.

Monsieur AIT HADDADENE Hacene, Professeur à l'U.S.T.H.B., Monsieur CHERGUI Mohamed El-Amine, Maître de conférences à l'U.S.T.H.B., et Monsieur REBAINE Djamel, Professeur à l'université du Quebec à Chicoutimi, d'avoir accepté d'évaluer cette thèse.

Je remercie toutes les personnes formidables qui font partie de mon équipe de recherche, pour la bonne ambiance, merci à Amina, Nadjat, Karim, Abdelhakim et un grand remer-

ciement particulier à Mohamed Bendraouche pour ses encouragements, sa gentillesse, sa disponibilité et pour avoir corrigé mon papier rédigé en anglais.

Merci également à tous mes professeurs de l'U.S.T.H.B et à tou(te)s mes ami(e)s, spécialement Sara.

Enfin, les mots les plus simples étant les plus forts, j'adresse toute mon affection à ma famille (ma soeur Jalila et son mari Rabah, ma chère petite nièce Lina, mes frères : Bilel et sa femme Zahra, Mohamed et Omar, ma cousine Wafia, ma grand mère,...) et en particulier à ma mère, malgré son éloignement depuis de nombreuses années, son amour me porte et me guide tous les jours.

Résumé

Nous traitons dans cette thèse, le problème d'ordonnancement sous contraintes de préparation. Ce problème consiste à ordonnancer un ensemble de tâches indépendantes sur un ensemble de machines parallèles identiques. Nous supposons qu'il existe k types de ressources renouvelables nécessaires à la préparation des tâches. Chaque tâche possède pour son traitement un temps d'exécution qui doit être précédée par une phase de préparation réalisée par un sous ensemble de ressources. Le critère d'optimalité étudié est la minimisation de la date de fin de traitement de l'ensemble des tâches, connu sous le nom de makespan. Nous avons étudié deux sous problèmes. Dans le premier, le nombre de types de ressources est arbitraire, chacun disponible en une unité. Pour le deuxième sous problème considéré, nous supposons qu'un seul type de ressources est disponible en q unités. Pour chaque sous problème, nous avons montré que certains cas particuliers sont NP-difficiles, et d'autres sont résolubles en des temps polynomiaux. Nous avons également proposé des heuristiques et des métaheuristiques suivies d'une étude expérimentale.

Mots clés : Complexité, heuristiques, machines parallèles, makespan, métaheuristiques, ordonnancement, ressources, temps de préparation.

Abstract

In this thesis, we study the scheduling problem under preparation time constraints. This problem consists in scheduling a set of independent jobs on a set of identical parallel machines. We assume that there are k types of renewable resources needed for the preparation of the jobs. Each job has an execution time and requires prior to its execution a preparation time completed by a subset of resources. The objective is to find a schedule that minimizes the makespan. We have considered two subproblems. In the first, the number of resource types is arbitrary of one unit of each type. In the second sub-problem, we assume that only one type of resource is available in q units. For each sub-problem, we have investigated special cases for which we either proved their NP-hardness or are well solvable. We also have proposed heuristics and metaheuristics and conducted with numerical experiments.

Keywords : Complexity, heuristics, parallel machines, makespan, metaheuristics, Scheduling, resources, preparation times.

Table des matières

Introduction **8**

1 Concepts de base **11**

1.1 Introduction 11

1.2 Concepts de base en ordonnancement 12

1.2.1 Les tâches 12

1.2.2 Les ressources 12

1.2.3 Les contraintes 13

1.2.4 Objectifs 13

1.3 Ordonnancement avec des temps de préparation 14

1.4 Classification des problèmes d'ordonnancement 15

1.5 Réductions entre problèmes d'ordonnancement 16

2 Définition du problème et état de l'art **19**

2.1 Introduction 19

2.2 Définition et notation du problème 20

2.3 Motivation 21

2.4 Etat de l'art 23

3 Etude du problème avec différents types de ressources 30

3.1 Définition du problème 30

3.2 Résultats de complexité 31

3.2.1 Problem $P2|res^t \cdot 11, s_i \in \{1,3,5\}, p_i = 1|C_{max}$ 32

3.2.2 Problème $P2|res^t k11, s_i \in \{1,3\}, p_i = 1, r_i \in \{0,r\}|C_{max}$ 36

3.3 Cas particuliers 39

3.3.1 Problème $Pm|res^t 111, s_i \geq 1, p_i = 1|C_{max}$ 39

3.3.2 Problème $P2|res^t \cdot 11, s_i = s, p_i = p|C_{max}$ 40

3.4 Bornes inférieures 44

4 Approche de résolution pour le problème $Pm|res^t \cdot 11|C_{max}$ 45

4.1 Approche heuristique 45

4.2 Expérimentations numériques 49

4.3 Approche métaheuristique 53

4.3.1 Recuit simulé 53

4.3.2 Algorithme génétique 55

4.3.3 Algorithme génétique hybride 59

4.4 Expérimentations numériques 62

5 Etude du problème à 2 machines et un seul type de ressources 67

5.1 Définition et notations 67

5.2 Modélisation mathématique 68

5.2.1 Les variables 68

5.2.2 Les contraintes 69

5.2.3 Taille du modèle 71

5.3 Quelques sous problèmes 72

5.3.1 Problème $P2|res^t1\cdot\cdot, s_i = s, p_i = p|C_{max}$, $q \geq 2$ 73

5.3.2 Problème $P2|res^t12\cdot, s_i = s, p_i = p|C_{max}$ 74

5.4 Approche de résolution 75

5.4.1 Bornes inférieures 75

5.4.2 Approche heuristique 78

5.4.3 Expérimentations numériques 79

Conclusion et perspectives **85**

Bibliographie **86**

A Expérimentations numériques du problème $Pm|res^t\cdot 11|C_{max}$ **93**

Introduction

Les problèmes d'optimisation peuvent être rencontrés dans tous les domaines de la vie courante : les secteurs économiques (la production manufacturière, le transport et la gestion du personnel), les administrations, l'armé, les hôpitaux, les institutions scolaires et universitaires (on veut créer des horaires pour les étudiants en affectant des locaux et des enseignants aux différents cours dont l'objectif est de minimiser le nombre de conflits d'horaires, ...), etc. Par exemple, dans une usine, on cherche à ordonnancer la chaine de production pour qu'elle satisfasse différents objectifs comme terminer le travail le plus rapidement possible ou encore minimiser le nombre de commandes livrées en retard aux différents clients. La résolution de ce type de problèmes met à profit différents domaines scientifiques tels que l'informatique, les mathématiques et la recherche opérationnelle.

Cette thèse s'inscrit dans le domaine de l'ordonnancement qui constitue une branche de la recherche opérationnelle. L'ordonnancement vise à organiser l'utilisation des ressources technologiques et humaines présentes dans les ateliers de l'entreprise pour satisfaire soit directement les demandes des clients, soit les demandes issues d'un plan de production préparé par la fonction de planification de l'entreprise. Compte tenu de l'évolution des marchés et de leurs exigences, la fonction d'ordonnancement doit organiser l'exécution simultanée de multiples travaux à l'aide de ressources polyvalentes disponibles en quantités limitées, ce qui constitue un problème complexe à résoudre.

De très nombreuses études se sont portées sur la résolution des problèmes d'ordonnancement déterministes où les données sont parfaitement connues. Les auteurs proposent généralement des méthodes spécifiques et adaptées pour générer un ordonnancement satisfaisant les contraintes du système et ayant des performances optimales ou proches des optimums. Il faut remarquer que les problèmes d'ordonnancement sont des problèmes d'optimisation combinatoire dont la majeur partie sont NP-diffciles. Donc la résolution par des méthodes exactes est peu réaliste pour des problèmes de grande dimension, ce qui justifie le recours à des méthodes heuristiques performantes.

Dans la littérature classique de l'ordonnancement, les problèmes sont classés par famille

selon la structure de l'atelier. Notre travail de recherche est principalement axé sur les ateliers à plusieurs machines parallèles identiques avec prise en compte de contraintes supplémentaires telles que les contraintes de préparation et de ressources.

La plupart des études des problèmes d'ordonnancement considèrent que les temps de préparation des tâches sont inclus dans les temps d'exécution, ou négligeables, ce qui en réalité n'est pas toujours le cas. La prise en compte de ces temps est essentielle en pratique car elle peut avoir un impact important sur l'ordonnancement. En séparant les temps de préparation des temps d'exécution, plusieurs résultats ont été publiés et résumés dans [4, 72, 5].

Dans plusieurs problèmes d'ordonnancement, les tâches peuvent avoir besoin de ressources renouvelables qu'elles soient humaines ou matérielles pour mener à bien leur exécution. Ce genre de problèmes est connu dans la littérature sous le terme Resource-constrained scheduling problems. En général, lorsque les ressources sont considérées, elles sont attribuées aux tâches qui sont en cours d'exécution. En prenant en compte cette hypothèse, plusieurs travaux ont été publiés dans la littérature pour différents problèmes d'ordonnancement, spécialement pour les problèmes à machines parallèles [25, 26, 44, 68]. Cependant, dans notre étude, une tâche n'a pas besoin de ressources durant son exécution comme il est généralement supposé dans la littérature, mais seulement pendant un temps donné qui précède le début de l'exécution, appelé temps de préparation.

L'objectif de l'étude présentée dans cette thèse est de prendre en compte les activités de préparation et de ressources qui peuvent être nécessaires avant d'exécuter une tâche. Notre démarche de recherche a évolué selon deux axes, d'une part les nouveaux résultats concernant la complexité du problème, d'autre part, la construction de nouvelles méthodes de résolution.

Cette thèse est constituée de 5 chapitres.

Le chapitre 1 est une brève présentation aux problèmes d'ordonnancement auxquels nous nous intéressons dans ce travail. Nous décrivons ces problèmes ainsi que les notations utilisées. Nous donnons également un bref rappel sur la classification et les réductions entres les problèmes d'ordonnancement déterministes.

Le chapitre 2 présente le problème général et les notations essentielles pour une bonne compréhension des modèles sous contraintes de préparation et de ressources. Un état de l'art concernant les problèmes d'ordonnancement à machines parallèles avec temps de préparation est également exposé.

Les chapitres 3 et 4 sont consacrés à l'étude du problème d'ordonnancement sur m machines parallèles identiques avec différents types de ressources où chacun est disponible en une unité. Ces deux derniers sont organisés comme suit : Le troisième chapitre porte sur la défintion du problème et à sa complexité. Nous prouvons la NP-complétude de deux cas particuliers où les temps d'exécution des tâches sont unitaires, dans le premier cas, les temps de préparation n'ont que trois valeurs possibles et pour le second cas, les temps de préparation ne peuvent prendre que deux valeurs possibles et les tâches ont des dates de disponibilité qui ne peuvent prendre que deux valeurs. Nous étudions également deux cas spéciaux avec des temps d'exécution identiques et nous proposons des bornes inférieures pour le problème.

Le quatrième chapitre est divisé en quatre sections. Dans la première section, nous présentons un ensemble d'heuristiques pour résoudre le problème, dans la seconde section, nous effectuons une série de tests pour évaluer les heuristiques proposées. La troisième section est consacrée aux métaheuristiques, nous adoptons le recuit simulé ainsi que l'algorithme génétique à notre problème, aussi nous présentons une méthode hybride basée sur les algorithmes de recherches locales et l'algorithme génétique. La dernière section porte sur l'évaluation numérique de la performance des méthodes proposées.

Le chapitre 5 est dédié au problème d'orodonnancement avec un seul type de ressources disponible en q unités. Ce chapitre est organisé comme suit : la première section définit le problème, la deuxième section présente une modélisation mathématiques sous forme d'un programme linéaire en nombres entiers du problème. Dans la troisième section nous étudions deux sous problèmes où les temps de préparation et d'exécution sont constants, et dans la dernière section, nous proposons une approche de résolution où nous présentons les bornes inférieures, les heuristiques ainsi que des expérimentations numériques.

Enfin, une conclusion clôt ce tapuscrit avec un résumé des principaux résultats auxquels nous avons aboutis ainsi que des perspectives pour des recherches futures.

Chapitre 1

Concepts de base

Ce chapitre est dédié à la présentation des concepts de base des problèmes d'ordonnancement qui seront utiles pour les chapitres suivants. Le but de ce dernier n'est pas d'effectuer une présentation exhaustive sur l'ordonnancement, mais plutôt d'introduire de manière succincte les problèmes, les critères et les contraintes qui nous intéressent. Pour plus de détails sur la théorie de l'ordonnancement nous pouvons citer les livres de Carlier et Chrétienne [22], Pinedo [63], Brucker [17] et Blazewicz et al. [14].

1.1 Introduction

L'organisation et la gestion de la production conditionnent le succès des projets du monde de l'entreprise et de la recherche. Dans ce processus, la fonction ordonnancement vise à organiser l'utilisation des ressources technologiques ou humaines pour répondre à une demande ou satisfaire un plan de production préparé par la fonction planification. Ainsi, des programmes ambitieux privés ou publics ont recours à la fonction ordonnancement pour appréhender la complexité, améliorer les délais ou même s'adapter à des évènements imprévus. Les problèmes d'ordonnancement apparaissent dans de nombreux domaines : l'industrie (atelier, gestion de production), la construction (suivi de projets), l'informatique (gestion des processus) et l'administration (emplois du temps).

Un problème d'ordonnancement est composé de façon générale d'un ensemble de tâches soumises à certaines contraintes, et dont l'exécution nécessite des ressources. Résoudre un problème d'ordonnancement consiste à organiser ces tâches, c'est-à-dire à déterminer leurs dates de début et fin, et à leur attribuer des ressources, de telle sorte que les contraintes soient respectées. Les problèmes d'ordonnancement sont très variés. Ils sont

caractérisés par un grand nombre de paramètres relatifs aux tâches (morcelables ou non, indépendantes ou non, durées fixes ou non), aux ressources (renouvelables ou consommables), aux types de contraintes portant sur les tâches (contraintes temporelles, fenêtres de temps, etc.), aux critères d'optimalité liés au temps (délai total, retards, etc.), aux ressources (quantité utilisée, taux d'occupation, etc.) ou à d'autres coûts (production, stockage, etc.). Donc, un problème d'ordonnancement s'énonce classiquement à partir de quatre notions fondamentales : les tâches, les ressources, les contraintes et les objectifs à optimiser.

1.2 Concepts de base en ordonnancement

1.2.1 Les tâches

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début t_i ou de fin C_i, dont la réalisation nécessite une durée p_i ($C_i = t_i + p_i$ si la tâche s'exécute sans interruption), et qui consomme un moyen selon une certaine intensité. Dans cette thèse, l'ensemble des tâches est noté $T = \{T_1, T_2, ..., T_n\}$ où n désigne le nombre de tâches du problème. D'autres paramètres peuvent aussi être définis : d_i (date échue), r_i (date de disponibilité), w_i (un poids) et $f_i = C_i - r_i$ (durée de séjour).

Dans certains problèmes, les tâches peuvent être exécutées par morceaux et dans d'autres au contraire, on ne peut pas interrompre une tâche une fois commencée. On parle alors respectivement de problèmes *préemptifs* et *non préemptifs*.

1.2.2 Les ressources

Les ressources constituent les moyens de production de l'atelier et sont donc l'élément fondamental. La difficulté du problème d'ordonnancement avec prise en considération de ressources vient du fait que leur capacité est limitée. Deux types de ressources sont à distinguer, les ressources *renouvelables* deviennent à nouveau disponibles en même quantité après avoir été utilisées par une ou plusieurs tâches (les machines, les hommes, l'équipement en général, etc.). Dans le cas contraire, elles sont *consommables* (matière première, budget, etc.).

Parmi les ressources *renouvelables*, on distingue les ressources *disjonctives* (ou non partageables) qui ne pouvant exécuter qu'une tâche à un instant donné (machine-outil, robot

manipulateur) et les ressources *cumulatives* (ou partageables) pouvant exécuter plusieurs tâches simultanément (équipe d'ouvriers, poste de travail). Le cumul des demandes d'une ressource *cumulative* à un instant ne peut dépasser la disponibilité totale de la ressource à cet instant.

Dans la littérature classique, les problèmes sont classés selon la structure de l'atelier, on distingue :

– Atelier à une machine : composé d'une seule machine pour exécuter toutes les tâches.

– Atelier à machines parallèles : composé de m machines parallèles. Chaque tâche doit être exécutée sur une machine à sélectionner dans l'ensemble des machines parallèles. Les machines peuvent être alors identiques (P), uniformes (Q) ou indépendantes (R).

– Atelier flowshop (F) : les machines sont disposées en ligne où chaque tâche doit passer sur chaque machine et cela dans le même ordre pour toutes les tâches.

– Atelier jobshop (J) : chaque tâche dispose de son propre ordre de passage sur les machines.

– Atelier openshop (O) : les opérations de chaque tâche doivent être exécutées sur des machines différentes mais dans un ordre quelconque.

1.2.3 Les contraintes

Les contraintes des problèmes d'ordonnancement expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision. On distingue deux types, *les contraintes temporelles* qui sont relatives aux dates limites des tâches (délais de livraisons, disponibilité des approvisionnements) ou à la durée totale d'un projet. *Les contraintes de ressources* expriment la nature et la quantité des moyens utilisés par les tâches, les caractéristiques d'utilisation de ces moyens, la disponibilité des ressources et la quantité des moyens disponibles au cours du temps.

1.2.4 Objectifs

Lorsque l'on aborde la résolution d'un problème d'ordonnancement, on peut choisir entre deux grands types de stratégies, visant respectivement à l'optimalité des solutions par

rapport à un ou plusieurs critères, ou à leur admissibilité vis-à-vis des contraintes. L'approche par optimisation suppose que les solutions candidates à un problème puissent être ordonnées de manière rationnelle selon un ou plusieurs critères d'évaluation numériques permettant d'apprécier la qualité des solutions. On cherchera à minimiser (maximiser) de tels critères. On note par exemple ceux :

- liés au temps : le temps total d'exécution ou le temps moyen d'achèvement d'un ensemble des tâches ; les différents types de retards par rapport aux dates échues fixées :
  - $C_{max} = \max\limits_{1 \leq i \leq n} \{C_i\}$.
  - $\overline{C} = \frac{1}{n} \sum\limits_{1 \leq i \leq n} C_i$.
  - $\overline{C_w} = \frac{\sum\limits_{1 \leq i \leq n} w_i C_i}{\sum\limits_{1 \leq i \leq n} w_i}$.
  - $L_{max} = \max\limits_{1 \leq i \leq n} \{C_i - d_i\}$.
  - $\overline{D} = \frac{1}{n} \sum\limits_{1 \leq i \leq n} max\{0, C_i - d_i\}$.
  - etc.
- liés aux ressources : la quantité -maximale, moyenne ou pondérée - de ressources nécessaires pour réaliser un ensemble de tâches ; la charge de chaque ressource.
- liés aux coûts de lancement, de production, de transport, de stockage, etc.

1.3 Ordonnancement avec des temps de préparation

Certaines ressources nécessitent avant l'exécution d'une tâche, la réalisation d'une activité de préparation ayant pour objectif de les amener dans une configuration compatible avec la tâche à réaliser. Dans notre étude, cette activité correspond au temps de préparation qui englobe les temps nécessaires aux travaux effectués sur les machines ou sur les tâches avant de commencer l'exécution. Cette préparation peut être de différentes natures. A titre d'exemple, on peut citer le débranchement, le montage, le démontage, le nettoyage, etc. Selon le contexte, cette activité de préparation peut être ou bien intégrée dans la valeur donnée pour le temps d'exécution d'une tâche, ou bien doit être explicitement prise en compte. Le premier contexte est valable si la durée de préparation est faible par rapport au temps d'exécution, si la préparation nécessite la présence de pièces pour être réalisée, si elle ne nécessite pas de ressources autres que celles utilisées (même machine par exemple), ou bien si la préparation peut s'effectuer en temps masqué, c'est-à-dire sans occuper les ressources. L'objectif ici est de prendre un deuxième contexte, pour lequel la préparation peut être anticipée avant l'arrivée de la tâche (arrivée des pièces sur les machines, par exemple). On distingue deux cas :

– Les temps de préparation indépendants de la séquence (no sequence dependent preparation times). Le temps de préparation dépend seulement de la tâche à effectuer, noté s_i.
– Les temps de préparation dépendants de la séquence (sequence dependent preparation times), où la préparation dépend à la fois de la tâche à effectuer T_j et de celle qui la précède T_i, noté s_{ij}.

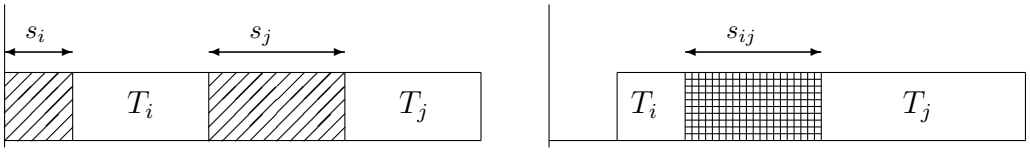


FIG. 1.1 – Différentes formes des temps de préparation

Le critère à minimiser lié aux temps de préparation le plus fréquemment utilisé est le temps total de préparation, noté $\sum s_i$ ou $\sum s_{ij}$, selon le cas. Ce critère peut être combiné avec d'autres critères classiques sous la forme d'une somme pondérée. La préparation peut aussi être absente du critère, l'objectif étant la minimisation d'un critère classique (C_{max}, L_{max}, etc.) avec prise en compte des contraintes de préparation.

1.4 Classification des problèmes d'ordonnancement

Plusieurs notations ont été introduites dans la littérature pour spécifier un problème d'ordonnancement. Un schéma de classification proposé par Graham et al. [38] structure les données d'un problème d'ordonnancement sur la base d'une notation à trois champs distincts $\alpha/\beta/\gamma$. Le champ α décrit l'environnement machine qui est décomposé en deux sous-champs, le premier indique la nature de l'atelier (α_1) et le second précise le nombre de machines (α_2). Le champ β décrit les contraintes liées aux tâches, il peut inclure plusieurs sous-champs et le champ γ décrit le ou les critères à optimiser.

En ajoutant les contraintes des ressources renouvelables, Blazewicz et al. [16] ont élargi cette classification. En effet, la présence des ressources renouvelables est spécifié par $res\ \lambda\sigma\delta$ où les paramètres λ, σ et δ représentent respectivement, le nombre de types de ressources, la quantité totale de ressources disponibles par unité de temps, et le besoin maximale en ressources des tâches. Ces paramètres sont caractérisés comme suit :

– Si $\lambda, \sigma, \delta = \cdot$, le nombre de types de ressources, la quantité totale de ressources dispo-

nibles et le besoin maximal en ressources des tâches sont respectivement arbitraires.

– Si $\lambda, \sigma, \delta = k$, le nombre de types de ressources est égal à k, chaque ressource est disponible en k quantité et le besoin maximal en ressources de chaque tâche est égal à k unités.

Exemple 1. *L'utilisation de ces notations est illustrée par les exemples suivants.*

1. *$P2|si|L_{max}$: ce problème traite un ensemble de tâches indépendantes sur deux machines parallèles identiques où chaque tâche nécessite un temps de préparation avant son exécution, l'objectif est la minimisation de la grande tardiveté.*

2. *$P2, S1|si = 1|IT$: c'est le problème à deux machines parallèles identiques avec la présence d'un seul serveur pour la préparation des tâches dont l'objectif est de minimiser le temps mort.*

3. *$P|res\ sor, p_i = 1|C_{max}$: ce problème consiste à ordonnancer un ensemble de tâches indépendantes sur des machines parallèles identiques, le nombre de types de ressources, la quantité totale de ressources disponibles, et le besoin maximale en ressources des tâches sont fixés à s, o, r respectivement. Les temps d'exécution des tâches sont unitaires. L'objectif est de minimiser la date de fin de traitement de l'ensemble des tâches.*

1.5 Réductions entre problèmes d'ordonnancement

Un effort considérable a été fait pour établir une hiérarchie décrivant les relations entre certains problèmes d'ordonnancement. L'étude de ces relations revêt un grand intérêt, dans la mesure où cela permet d'appliquer des algorithmes de résolution connus pour certaines classes de problèmes à d'autres classes qui leurs sont réductibles. En comparant entre la complexité des différents problèmes d'ordonnancement, il est intéressant de voir comment la modification d'un seul élément de la classification d'un problème peut affecter sa complexité. Pour cela, nous exposons quelques schémas représentant les relations entre les différents problèmes. Une flèche indique la direction de la transformation polynomiale. Ces simples transformations sont très utiles dans de nombreuses situations pour analyser de nouveaux problèmes d'ordonnancement. Pour une représentation plus complète de ces relations, il est possible de se référer à Pinedo [63] (Figure 1.2, Figure 1.3) et Blazewicz et al. [14] (Figure 1.4).

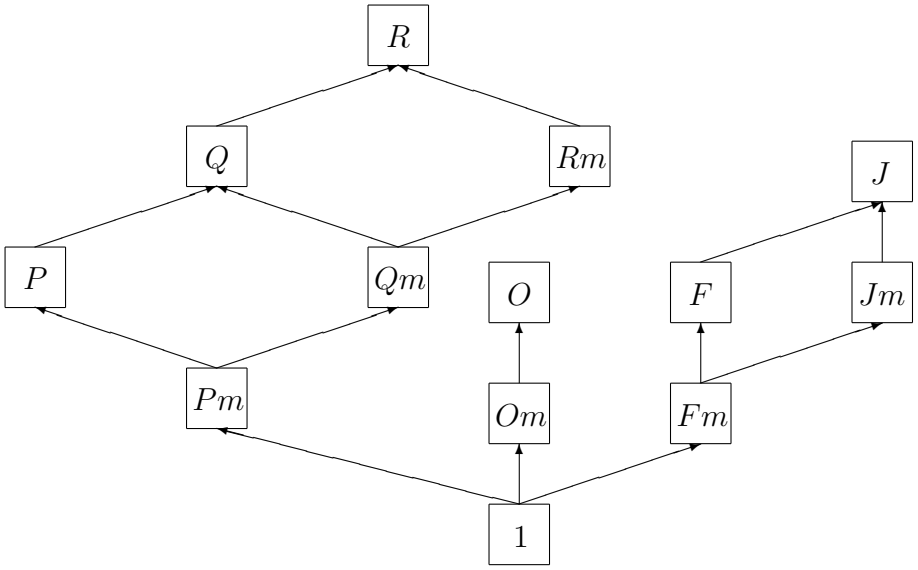


FIG. 1.2 – Hiérarchie de complexité (Environnement des machines)

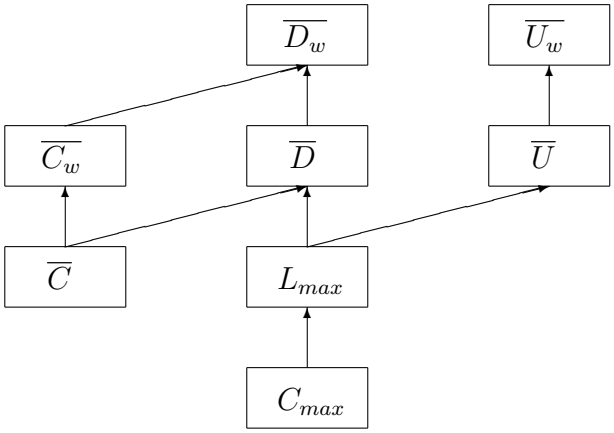


FIG. 1.3 – Hiérarchie de complexité (Critères d'optimalité)

La Figure 1.2 indique la hiérarchie de complexité entre des problèmes de configurations de machines différentes. Comme nous le montre la figure, le cas le plus simple est le cas à une machine. Le système devient plus complexe si on considère plusieurs machines en parallèle. La difficulté augmente encore, si le nombre de machines est non borné et si l'on considère, non plus des machines identiques, mais des machines uniformes ou indépendantes.

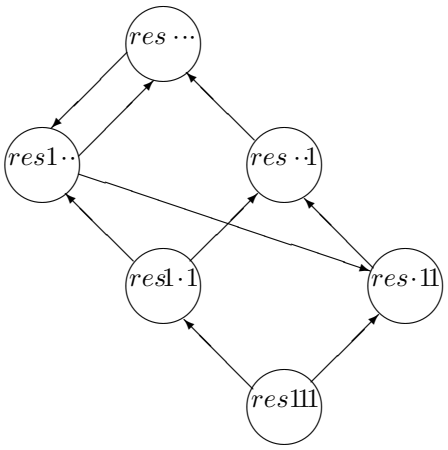


FIG. 1.4 – Réductions polynomiales entre les problèmes d'ordonnancement sous contraintes de ressources.

La Figure 1.3 représente la relation existant entre les principaux critères d'optimisation. Par exemple, le problème $1||\sum C_j$ est un cas particulier du problème $1||\sum w_jC_j$. Ainsi $1||\sum C_j$ est réductible à $1||\sum w_jC_j$, qui lui est plus général et tout algorithme développé pour $1||\sum w_jC_j$ peut être appliqué au problème $1||\sum C_j$.

La Figure 1.4 présente les transformations possibles entre les problèmes d'ordonnancement qui nécessitent des ressources renouvelables et qui se diffèrent seulement par leurs besoins en ressources. Toutes, sauf deux de ces transformations sont tout à fait évidentes. La transformation entre les problèmes $\Pi(res\cdot\cdot\cdot)$ et $\Pi(res1\cdot\cdot)$ a été prouvée dans [33] et la deuxième transformation entre les problèmes $\Pi(res1\cdot\cdot)$ et $\Pi(res\cdot11)$ a été prouvée dans [11].

Chapitre 2

Définition du problème et état de l'art

Dans ce chapitre, nous décrivons le problème d'ordonnancement traité ainsi que sa notation. Un exemple illustratif est aussi donné. Nous présentons ensuite une motivation du problème, ainsi qu'un état de l'art qui résume les travaux existant dans la littérature et qui ont des liens avec notre problème.

2.1 Introduction

L'intérêt de la théorie de l'ordonnancement a surgi dans les années 1950 et n'a cessé d'augmenter ces dernières années. Depuis, des centaines de papiers pour différents problèmes d'ordonnancement sont apparus dans la littérature et la majorité suppose que les temps de préparation sont négligeables ou faisant partie de la durée d'exécution. Bien que cette hypothèse simplifie l'analyse et la résolution de certaines applications, dans d'autres elle affecte sur la qualité de la solution des différents problèmes d'ordonnancement qui nécessitent un traitement explicite des temps de préparation. Ce n'est qu'à partir des années 1960 que les chercheurs ont commencé à s'intéresser aux problèmes d'ordonnancement qui traitent les temps de préparation séparément des temps d'exécution. Les principaux résultats ont été résumés dans [4, 72, 5].

Le problème que nous abordons dans cette thèse est un problème d'ordonnancement sur machines parallèles identiques dont le but est de minimiser la durée totale. Nous nous intéressons particulièrement au cas où une tâche nécessite avant son exécution un

temps de préparation indépendant de la séquence réalisée par un ensemble de ressources renouvelables.

2.2 Définition et notation du problème

Le problème d'ordonnancement sous contraintes de préparation peut être décrit comme suit : Un ensemble $T = \{T_1, T_2, ..., T_n\}$ de n tâches indépendantes doit être exécuté sur un ensemble de m machines parallèles identiques $M = \{M_1, M_2, ..., M_m\}$. Outre que les machines, nous supposons qu'il existe un ensemble de k types de ressources renouvelables $R = \{R_1, R_2, ..., R_k\}$, qui sont disponibles en quantités $q_1, q_2, ..., q_k$ unités, respectivement. Chaque tâche T_i, $i = 1, ..., n$ possède un temps d'exécution p_i et avant son exécution, elle nécessite une phase de préparation qui est réalisée par un sous-ensemble de ressources représenté par le vecteur $R(T_i) = [R_1(T_i), R_2(T_i), ..., R_k(T_i)]$ et prend s_i unités du temps. $R_l(T_i)$ $(0 \leq R_l(T_i) \leq q_l)$, l=1,2...,k, défini le nombre d'unités de la ressource R_l nécessaire pour la préparation de la tâche T_i. Durant la phase de préparation, la machine n'est pas disponible pour une autre tâche et l'exécution d'une tâche doit être effectuée immédiatement après sa préparation comme l'indique la Figure 2.1. Nous supposons que toutes les tâches sont disponibles pour la préparation à la date 0. L'interruption de la préparation et de l'exécution des tâches n'est pas autorisée. L'objectif est de trouver un ordonnancement minimisant la date de fin de traitement de l'ensemble des tâches (makespan) noté C_{max}.

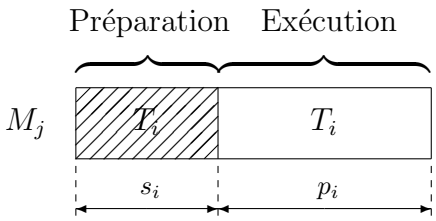


FIG. 2.1 – Traitement de la tâche T_i sur la machine M_j. La partie hachurée correspond à la phase de préparation.

Afin de faciliter la description et la classification des problèmes d'ordonnancement sous contraintes de préparation, nous allons adopter le même formalisme cité dans le chapitre précédent $(\alpha/\beta/\gamma)$ et l'adapter au problème étudié. Le champ β est décomposé en quatre sous-champs $\beta = \beta_1\beta_2\beta_3\beta_4$:

– $\beta_1 \in \{\phi, p_i = p\}$: les temps d'exécution des tâches sont soit quelconques, soit égaux à p.

– $\beta_2 \in\{\phi,\ s_i = s\ \}$: les temps de préparation des tâches sont soit quelconques, soit égaux à s

– $\beta_3 \in\{res^t\ \lambda\sigma\delta,\ \phi\}$: correspond à l'existence ou non des contraintes de ressources en relation avec les temps de préparation.

– $\beta_4 \in\{\phi,\ r_i\ \}$: les dates de disponibilité des tâches sont soit nulles, soit arbitraires.

Notre problème est donc noté $Pm|res^t \cdots |C_{max}$.

Pour bien illustrer le problème défini dans le paragraphe précédent, considérons l'exemple suivant :

Exemple 2. *Considérons un problème de réalisation de* 8 *tâches* $T_1, T_2, ..., T_8$ *dans cet ordre sur quatre machines identiques* M_1*,...,* M_4*. Le nombre de types de ressources est égal à trois, la disponibilité de chaque type de ressource est donné par la Table 2.1. Les temps de préparation, les durées d'exécution des tâches ainsi que les demandes en ressources pour chaque tâche sont donnés par la Table 2.2.*

R_i	R_1	R_2	R_3
q_i	2	2	4

Tab. 2.1 – Disponibilité de chaque type de ressource

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
s_i	2	2	1	3	2	2	3	1
p_i	2	1	3	1	1	2	2	3
$R(T_i)$	[1, 0, 3]	[1, 1, 1]	[2, 1, 0]	[0, 1, 3]	[2, 2, 4]	[1, 0, 0]	[2, 1, 3]	[0, 0, 3]

Tab. 2.2 – Caractéristiques des tâches

Le diagramme associé à cette séquence de tâches est donné par la Figure 2.2 :

2.3 Motivation

Le problème défini peut trouver plusieurs applications en industrie, par exemple, il peut être issu du processus de fabrication de pneumatique [62, 7]. La fabrication d'un pneu

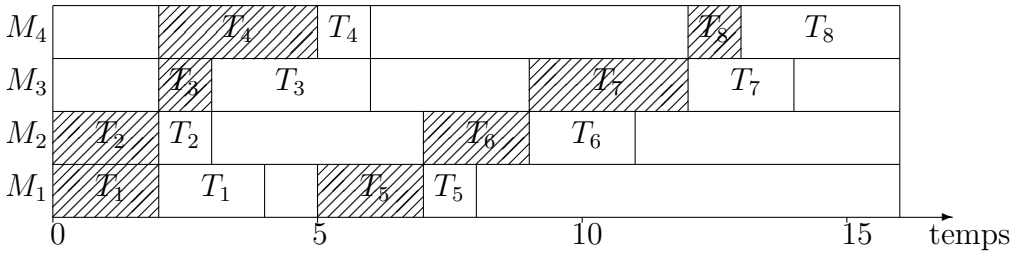


FIG. 2.2 – Une solution réalisable pour l'exemple 2 avec $C_{max} = 16$.

passe par plusieurs phases et chaque phase du processus de fabrication requérant une grande précision, et d'importants contrôles.

La phase préparatoire est constituée de deux étapes : a) La fabrication de la gomme qui consiste à mélanger les matières premières (caoutchoucs naturels ou synthétiques, huiles, noir de carbone, antioxydants, souffre et autres additifs). b) La fabrication des différentes couches du pneu.

La phase de construction est aussi constituée de deux étapes : a) L'assemblage, qui comprend deux sous étapes : 1) Le processus qui consiste à assembler les semi-finis suivant un ordre d'assemblage et de positionnement bien précis. Ce dernier s'effectue sur une machine spéciale, elle est essentiellement constituée par un tambour tournant, sur lequel on superpose les semi-finis. 2) La finition, commence par la conformation de la carcasse à la forme du futur pneumatique. b) La cuisson : le pneu est cuit dans des presses de vulcanisation. Ces presses peuvent contenir deux pneus, qui peuvent être de types différents, mais leur temps de cuisson est identique. Lorsque le moule d'une presse est fermé, la cuisson commencée on ne peut l'ouvrir qu'à la fin de la cuisson.

La première phase du processus ne nous intéresse pas, car la fabrication des composantes est souvent sous-traitée dans d'autres usines spécialisées. Nous nous intéressons de prés dans ce travail à l'étape de cuisson qui nécessite un ensemble de ressources (tel que la main-d'œuvre, les outils) qualifié pour commencer la période de préparation de la charge de moule et le positionnement des pneus sur les machines, le pneu est au curry après la phase de préparation. Dans cette situation, le temps de traitement d'une tâche sur une machine est ainsi décomposé en deux parties : le temps de préparation et le temps d'exécution. La période de préparation doit précéder la période d'exécution pour chaque tâche et exige un certain type de ressources.

Ce type de problème peut être rencontré aussi en informatique dans un environnement multiprocesseurs ou dans un réseau d'ordinateurs en présence de serveurs [71]. Le ser-

veur qui est une ressource se charge de l'allocation des processus aux ordinateurs ce qui correspond à la phase de préparation et le traitement des processus sur les processeurs correspond à la phase d'exécution.

2.4 Etat de l'art

Le but de cette section est de présenter les différents travaux qui se rapprochent de notre configuration du problème par un de ses aspects, comme les temps de préparation ou bien les ressources renouvelables.

Dans le cas d'un atelier à machine parallèles identiques, chaque tâche est constituée d'une seule opération qui peut être exécutée sur n'importe qu'elle machine. Ce type de problème a attiré l'attention de beaucoup de chercheurs en ordonnancement par son intérêt pratique.

Sans prendre en considération les contraintes de préparation, le problème de minimisation du makespan $P||C_{max}$ est largement étudié. Ce dernier est NP-difficile car le problème restreint $P2||C_{max}$ est NP-difficile [43]. Plusieurs heuristiques ont été développées pour sa résolution, parmi ces dernières, on cite les algorithmes de liste où une liste de priorité est donnée pour les tâches et à chaque étape la première machine disponible est sélectionnée pour traiter la première tâche libre de cette liste. L'un de ces simples et efficaces algorithmes est l'algorithme LPT (Longest Processing Time), qui range les tâches dans un ordre décroissant des durées de traitement. Sa performance au pire cas a été évaluée par Graham [38] égal à $\frac{C_{max}(LPT)}{C^*_{max}} \leq \frac{4}{3} - \frac{1}{3m}$. Avec C^*_{max} la valeur de la solution optimal, $C_{max}(LPT)$ la solution obtenue avec l'algorithme LPT et m le nombre de machines.

Toutefois, si on veut avoir de meilleures garanties de performance, d'autres algorithmes d'approximation peuvent être utilisés, comme par exemple $MULTIFIT$ introduit par Coffman et al. [23] ou l'algorithme proposé par Hochbaum et Shmoys [41].

Une approche par programmation dynamique a été construite en utilisant l'idée présentée par Rothkopf [65] qui permet de résoudre le problème $P||C_{max}$ en $O(nC^m)$, où C représente une borne supérieure pour le C^*_{max}.

Une autre méthode exacte a été proposée par Mokotoff [59] basée sur la résolution d'un programme linéaire en variables entières et bivalentes.

D'autres variantes du problème classique $P||C_{max}$ ont été également étudiées en intégrant d'autres contraintes. Ici, nous mentionnons seulement les études les plus pertinentes en ce

qui concerne le présent travail.

Les problèmes d'ordonnancement avec contraintes de ressources ont été largement étudiés dans la littérature, pour plus de détail, le lecteur peut se référer à [6, 10, 12, 16, 29, 28, 34, 45, 44].

Les problèmes d'ordonnancement sous contraintes de ressources pour l'affectation des tâches ont reçu peu d'attention dans la littérature, Abdelkhodaee et Wirth [1] ont étudié le problème d'ordonnancement sur deux machines parallèles identiques avec ressource supplémentaire qui est un serveur. Chaque tâche nécessite deux opérations, la première est l'opération de setup traitée par le serveur, alors que la deuxième opération est exécutée automatiquement par l'une des machines parallèles (sans serveur). L'objectif est de minimiser le makespan. Les auteurs ont prouvé que ce problème est NP-difficile au sens fort. Ils ont proposé une formulation mathématique en nombres entiers, ont présenté deux sous problèmes polynomiaux, et ont aussi proposé deux heuristiques avec des expérimentations numériques pour le problème général. L'opération de setup correspond à la phase de préparation dans notre problème, les auteurs ont relaxé la contrainte que la phase d'exécution d'une tâche doit être effectuée immédiatement après la phase de préparation.

Abdekhodaee et al. [2] ont discuté deux cas particuliers où les temps d'exécution sont identiques et les temps de setup sont aussi identiques. Ils ont montré que ces deux problème sont NP-difficile au sens faible et ont proposé des bornes inférieures ainsi que des heuristiques constructives.

Suite aux travaux cités en [3], Gan et al. [31] ont développé deux formulations de programmation linéaire mixte en nombres entiers et deux variantes of a branch-and-price scheme.

Koulamas [48] a traité le problème d'ordonnancement à deux machines parallèles Semi-automatiques pour minimiser le temps mort résultant de l'indisponibilité du robot avec la condition que ces deux machines partagent le même serveur (robot) pour la préparation des tâches, il a démontré que ce dernier est NP-difficile au sens fort, aussi il a développé une procédure de réduction pour le transformer en un problème de taille plus petite.

Kravchenko et Werner [50] ont considéré le problème d'ordonnancement de m machines parallèles en présence d'un seul serveur dont l'objectif est de minimiser le makespan ; ils ont présenté un algorithme pseudo-polynomial pour le cas de deux machines où les temps de préparation sont unitaires et ont prouvé que le problème avec un nombre arbitraire de machines et les temps de préparation sont égaux à 1 est NP-difficile, ils ont proposé des heuristiques pour sa résolution.

Dans [71] les mêmes auteurs ont étudié le problème d'ordonnancement où avant le début de l'exécution d'une tâche, un temps de setup est nécessaire et doit être effectué par un ensemble de serveurs. Ils ont proposé un algorithme pseudo-polynomial pour le problème avec des temps de setup unitaires, m machines et $m-1$ serveurs et ont montré que le problème avec un nombre fixe de machines et de serveurs pour minimiser le retard maximum est NP-difficile au sens faible. Aussi, ils ont généralisé des algorithmes pour les problèmes d'ordonnancement à machines parallèles avec des temps de traitement constants au problème correspondant avec serveurs lorsque $s_i = s$ et ils ont donné l'analyse du plus mauvais cas des deux heuristiques de liste.

Hasani et al. [40] ont considéré le problème d'ordonnancement sur deux machines identiques avec un seul serveur pour minimiser le makespan. Avant le traitement, chaque tâche doit être chargée sur une machine et prend un temps donné de setup qui doit être effectué par le serveur. Ils ont proposé une formulation de programmation linéaire mixte en nombres entiers en utilisant une simple idée qui consiste à une décomposition possible de l'ordonnancement en un ensemble de blocs et ils ont comparé la performance de ce modèle avec des heuristiques proposées dans [31]. Les expérimentations numériques ont montré que ce modèle performe mieux que les heuristiques.

Hall et al. [39] ont traité le problème d'ordonnancement non préemptif sur machines parallèles avec un seul serveur, sous plusieurs hypothèses des temps de setup et d'exécution, ils ont considéré plusieurs objectifs à optimiser et ont étudié la complexité de chaque problème.

Su [69] a considéré un problème d'ordonnancement en ligne sur deux machines identiques avec un seul serveur pour minimiser le makespan. Il a présenté un algorithme LPT en ligne pour le problème général, aussi pour le cas particulier où tous les temps de setup sont égaux à 1.

Sahney [66] a étudié le problème à deux machines parallèles avec switching time (c'est le temps nécessaire à l'ouvrier pour aller d'une machine à une autre) et la présence d'un seul serveur, pour minimiser le temps de séjour moyen ($\overline{C}$), et il a proposé une heuristique pour le résoudre.

Zouba et al. [73] ont traité le problème d'ordonnancement non préemptif sur deux machines parallèles identiques en présence d'un seul opérateur pour minimiser le makespan. Ce problème consiste à déterminer des intervalles de temps en utilisant différentes affectations de l'opérateur pour ordonnancer les tâches sur les deux machines, ils ont présenté quelques propriétés caractérisant l'existence d'une solution optimale où les machines terminent le traitement des tâches au même instant.

Une ligne de production en forme de U avec des ouvriers multi-tâches a été étudiée par Nakade and Nishiwaki in [60], ils ont proposé un algorithme pour déterminer une affectation minimale des ouvriers aux machines afin de minimiser le temps global de traitement.

Lorsque les ressources sont nécessaires durant l'exécution des tâches, plusieures recherches sont considérées dans la littérature, en particulier pour problèmes d'ordonnancement avec machines parallèles, [33, 16, 12].

Garey and Johnson ont [33] montré que le problème $P2|res\cdot\cdot, p_i = p|C_{max}$ est résolu polynomialement en $O(n^{5/2})$, alors que le problème $P2|res1\cdot\cdot, p_i = 1|C_{max}$ est résolu en $O(nlogn)$.

Remy [64] a considéré le problème d'ordonnancement connu sous le nom de ressources limitées. Dans ce problème, à chaque tâche est assignée une longueur et un poids, et à chaque machine est assignée une capacité. L'objectif est de minimiser le temps d'exécution total ou la somme des temps d'achèvement. Il a montré que le problème peut être approximé avec un facteur de 2 (makespan) et 14.85 (somme des temps d'achèvement) pour m arbitraire.

Le problème d'ordonnancement sous contraintes de ressources avec date d'échue est étudié comme un problème dynamique en temps continu par Kogan [47]. L'objectif est de minimiser l'inventaire, carnet de commandes et des coûts de production. L'auteur a développé un algorithme polynomial pour le résoudre.

Blazewicz and Ecker [13] ont montré que le problème $P|res\, sor, p_i = 1|C_{max}$ (où le nombre de types de ressources, les limites de ressources et les besoins en ressources sont fixés par les entiers positifs s, o, r, respectivement) est résolu en $O(n)$, même pour un nombre arbitraire de ressources, où les temps de traitement des tâches appartiennent à un nombre fixe de classes L, ou les tâches de même classe ont le même traitement et les besoins en ressources. Blazewicz et al. [15] ont montré que le problème $Pm|res\, sor|C_{max}$ est résolu en $O(n^{L(m+1)})$, lorsque le nombre de machines m est fixé.

Blazewicz et al. [14] ont montré que le problème $P2|res\cdot 11, p_i = 1, r_j|C_{max}$ est NP-difficile au sens fort et que ce dernier est un cas particulier du problème $P2, |res^t\cdot 11, s_i = 1, p_i = p, r_i|C_{max}$ en posant $p = 0$ et en remplaçant $s_i = 1$ par $p_i = 1$, donc le problème $P2, |res^t\cdot 11, s_i = 1, p_i = p, r_i|C_{max}$ est NP-difficile au sens fort.

Kovalyov and Shafransky [49] ont étudié le problème d'ordonnancement d'un ensemble de tâches avec des temps unitaires sur m machines uniformes pour minimiser la date d'achèvement, où chaque tâche nécessite une unité d'une seule ressource supplémentaire

et a un temps de traitement interruptible. Le nombre total des ressources disponibles est donné. Ils ont montré que l'approche proposée dans [16] pour résoudre ce problème est incorrecte. Ils ont présenté un algorithme en $O(mlogm)$ pour le problème d'ordonnancement sans temps mort et un algorithme en temps linéaire pour le problème avec machines identiques.

Dans [51] nous avons étudié le problème d'ordonnancement sur machines identiques avec temps de préparation et la présence de k ouvriers spécialisés, où chaque tâche nécessite pour son traitement un temps de préparation effectué par un ouvrier spécialisé et un temps d'exécution, ce problème est noté $Pm|res^t1 \cdot 1|C_{max}$. Nous citons dans ce qui suit les principaux résultats présentés pour ce problème.

Nous avons modélisé le problème sous forme d'un programme linéaire en nombres entiers. Nous avons considéré les variables bivalentes X_{ijt} définies comme suit :

$X_{ijt}=\begin{cases} 1 & \text{si la tâche } T_i \text{ débute sa préparation sur la machine } M_j \text{ à l'instant t,} \\ 0 & \text{sinon.} \end{cases}$

$t = \overline{0, H-1}$ avec H un temps limite qu'on peut estimer à la borne supérieure.

Ainsi, Le modèle mathématique développé s'écrit :

Min μ

$$\begin{cases} \sum_{j=1}^{m} \sum_{t=0}^{H-1} X_{ijt} = 1 & i = \overline{1,n} & (1) \\ \sum_{y=t}^{t+s_i+p_i-1} X_{ijy} \leq 1 & j = \overline{1,m};\ i = \overline{1,n};\ t = \overline{0,H-1} & (2) \\ \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{y=max\{0,t-s_i+1\}}^{t} X_{ijy} \leq k & t = \overline{0,H-1} & (3) \\ (\sum_{j=1}^{m} \sum_{t=0}^{H-1} tX_{ijt}) + s_i + p_i \leq \mu & i = \overline{1,n} & (4) \\ X_{ijt} \in \{0,1\} & j = \overline{1,m};\ i = \overline{1,n};\ t = \overline{0,H-1} & (5) \\ \mu \in IR & & (6) \end{cases}$$

La première contrainte assure que l'on assigne une tâche à une machine et une seule. La deuxième contrainte indique que dans l'intervalle du temps $[t, t + s_i + p_i - 1]$ on ne peut traiter qu'au plus une seule tâche. La troisième contrainte signifie qu'on ne peut pas faire

plus de k préparations à un instant t donné. Et la dernière contrainte indique que la date de fin de traitement de chaque tâche doit être inférieure ou égale à μ.

Le modèle mathématique associé à $(nmH+1)$ variables et $(2n+(mnH)+H)$ contraintes.

Compte tenu de la complexité d'un tel modèle, sa résolution à l'aide d'un logiciel commercial ne semble possible que pour des problèmes de petite taille. Pour remédier à ce manque d'efficacité, nous avons proposé une approche de résolution heuristique.

Nous avons proposé une borne inférieure pour le C_{max} :

Proposition 1 (Labbi [51]). $\overline{BI} = \max\{\lceil \frac{1}{m}\sum_{i=1}^{n}(p_i+s_i)\rceil, \max_{i=1}^{n}\{p_i+s_i\}\}$ *est une borne inférieure.*

Si toutes les tâches sont traitées sur la même machine, on aura comme borne supérieure $\overline{S}$, tel que : $\overline{S} = \sum_{i=1}^{n}(p_i + s_i)$.

Nous avons traité un sous problème polynomial du problème général où les temps de préparation et d'exécution des tâches sont constants égaux à s et p respectivement. La valeur $\overline{M'}$ constitue une borne inférieure pour la solution optimale.

Algorithme A

début

- $\overline{M'}=\begin{cases} \frac{n(p+s)}{2} & \text{si } n \text{ est pair et } k=2\,; \\ \frac{n(p+s)}{2}+S & \text{si } n \text{ est pair et } k=1 \text{ avec } s\leq p\,; \\ n.s+p & \text{si } n \text{ est pair et } k=1 \text{ avec } s>p\,; \\ \frac{(n+1)(p+s)}{2} & \text{si } n \text{ est impair et } k=2\,; \\ \frac{(n+1)(p+s)}{2} & \text{si } n \text{ est impair et k} = 1 \text{ avec } s\leq p\,; \\ n.s+p & \text{si } n \text{ est impair et } k=1 \text{ avec } s>p\,; \end{cases}$
- $t := 0\,;\ i := 1\,;\ j := 1\,;\ l := 1\,;$
- **répéter**
  si $(t + p_i + s_i) \leq \overline{M'}$ **alors**
  - Affecter la préparation de la tâche T_i à l'ouvrier l et la traiter sur la machine M_j à l'instant t ;
  - $t := t + p_i + s_i\,;\ i := i + 1$

  sinon **si** $k = 1$ **alors** $t := s\,;\ j := j + 1$
  sinon $t := 0\,;\ l := l + 1\,;\ j := j + 1\,;$
  fsi ;
  fsi ;
  jusqu'à $i = n\,;$

fin.

Théorème 1 (Labbi [51]). *L'algorithme A résout le problème* $P2|res^t1k1, s_i = s, p_i = p|C_{max}$ *pour* $k \leq 2$ *en* $O(n)$.

Des algorithmes approchés ont été également développé pour résoudre ce problème. Ils se basent sur des règles de priorité LPT_{p_i}, SPT_{p_i}, LPT_{s_i}, SPT_{s_i}, $LPT_{s_i+p_i}$, $SPT_{s_i+p_i}$. Le principe de ces heuristiques est le suivant : la première phase consiste à ranger les tâches selon une règle de priorité, ensuite, faire appel à la procédure $A1$ dont le principe est qu'à l'instant $t = 0$, on suppose que tous les ouvriers spécialisés et toutes les machines sont disponibles, tant que l'ensemble des tâches n'est pas encore vide, on sélectionne une tâche T_i de cet ensemble et on l'affecte au premier ouvrier libre pour la traiter sur la première machine disponible à l'instant $t = 0$. Ensuite, on élimine cette tâche de l'ensemble des tâches, la préparation et l'exécution de la suivante commence à $t := \max\{\min\{O_l\}, \min\{C_j\}\}$.

Procédure A1

Début
$t := 0$;
$O_l := 0$; $//O_l$ représente la date de disponibilité de l'ouvrier l
$C_j := 0$;$//C_j$ représente la date de disponibilité de la machine j
Tantque $T \neq \emptyset$
  Faire
    Prendre la première tâche T_i de la liste et l'affecter au premier ouvrier libre (soit l) pour la traiter sur la première machine (soit M_j) disponible à l'instant t ;
    $O_l := t + s_i$; $C_j := O_l + p_i$; $T := T \setminus \{T_i\}$; $t := \max\{\min\{O_l\}, \min\{C_j\}\}$;
  Fait ;
Fin.

D'après les expérimentations numériques réalisées sur des instances générées aléatoirement selon la loi uniforme, nous avons conclu que l'heuristique basée sur le rangement des tâches selon la règle $LPT_{s_i+p_i}$ est assez performante et le rapport de performance $\frac{4}{3} - \frac{1}{3m}$ de Graham pour $P2//C_{max}$ (règle LPT) reste valable pour le problème $Pm|res^t1 \cdot 1|C_{max}$ pour $k \geq m$ (règle $LPT_{s_i+p_i}$).

Chapitre 3

Etude du problème avec différents types de ressources

Le début de ce chapitre est consacré à la définition du problème traité ainsi que les notations nécessaire à sa définition. Un exemple illustratif est aussi donné. Nous étudions ensuite la NP-complétude de deux cas spécifiques où dans le premier cas les temps de préparation n'ont que trois valeurs possibles et pour le second cas, les temps de préparation ne peuvent prendre que deux valeurs possibles et les tâches ont des dates de disponibilité qui peuvent prendre que deux valeurs. Nous traitons aussi deux sous problèmes avec des temps d'exécution identiques.

3.1 Définition du problème

Dans ce chapitre, nous considérons le problème avec un nombre arbitraire de types de ressources. Ce dernier peut être décrit comme suit : nous disposons de n tâches $T_1, ..., T_n$ à exécuter sur m machines parallèles identiques $M_1, ..., M_m$. Un nombre arbitraire de types de ressources $R = \{R_1, R_2, ..., R_k\}$ est disponible pour la phase de préparation où chacun est disponible en une unité $q_l = 1, \forall l = 1, 2..., k$. Le traitement de chaque tâche T_i nécessite un temps de préparation s_i, un temps d'exécution p_i et un sous ensemble de ressources représenté par le vecteur $R(T_i) = [R_1(T_i), ..., R_k(T_i)]$ pour la préparation. Durant cette phase, chaque ressource prépare au plus une tâche à la fois et chaque tâche ne peut être traitée que par une seule machine à la fois. L'objectif est de trouver un ordonnancement qui minimise la durée totale de traitement.

Exemple 3. *Considérons huit tâches indépendantes à exécuter sur trois machines identiques avec quatre types de ressources. Les temps de préparation, les temps d'exécution ainsi que le besoin en ressources pour chaque tâches sont donnés par la Table 3.1.*

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7
s_i	1	2	3	3	2	1	2
p_i	1	1	3	4	3	2	1
$R(T_i)$	[1,0,0,1]	[1,1,1,1]	[0,1,1,0]	[0,0,0,1]	[0,1,0,1]	[1,1,0, 0]	[1,0,0,0]

TAB. 3.1 – Caractéristiques des tâches.

Un ordonnacement réalisable est donné par la Figure 3.1 :

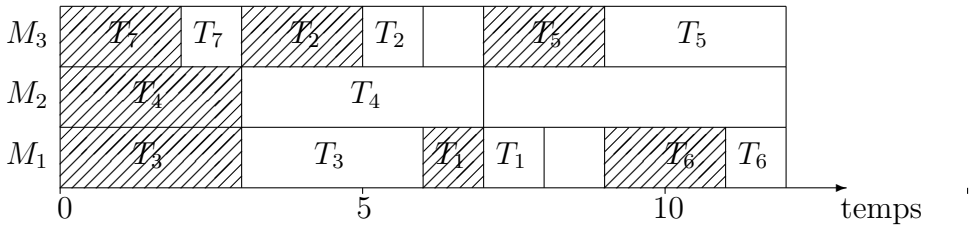


FIG. 3.1 – Solution réalisable de l'exemple 3 avec $C_{max} = 18$.

3.2 Résultats de complexité

Il est trivial de voir que le problème $P2||C_{max}$, qui est NP-difficile au sens faible, est un cas particulier des problèmes $P2|res^t \cdot 11, s_i, p_i = 1|C_{max}$ et $P2|res^t \cdot 11, s_i = 1, p_i|C_{max}$ en considérant que toutes les demandes en ressources sont égales à zéro et les nouveaux temps de traitement sont $p_i^{'} = s_i + p_i$. Ainsi, les problèmes $P2|res^t \cdot 11, s_i, p_i = 1|C_{max}$ et $P2|res^t \cdot 11, s_i = 1, p_i|C_{max}$ sont NP-difficiles au moins dans le sens faible.

Dans cette section, nous considérons deux cas particuliers du problème $P2|res^t \cdot 11, s_i, p_i|$ C_{max}, dans le premier cas, les temps de préparation des tâches ne peuvent prendre que trois valeurs possibles et les temps d'exécution de toutes les tâches sont égaux à 1, ce problème est noté $P2|res^t \cdot 11, s_i \in \{1, 3, 5\}, p_i = 1|C_{max}$, dans le second cas, les temps de préparation des tâches ne peuvent prendre que deux valeurs possibles, les temps d'exécution de toutes les tâches sont égaux à 1, les tâche ont des dates de disponibilité avec deux valeurs possibles, ce problème est noté $P2|res^t \cdot 11, s_i \in \{1, 3\}, p_i = 1, r_i \in \{0, r\}|C_{max}$. Nous montrons que ces deux problèmes sont NP-difficiles au sens fort en utilisant une réduction du problème NP-complet 3-Dimensional Matching (3-DM) [32] à nos problèmes.

Le problème 3-DM est énoncé comme suit. Soit $X = \{x_1, x_2, \ldots, x_q\}$, $Y = \{y_1, y_2, \ldots, y_q\}$ et $Z = \{z_1, z_2, \ldots, z_q\}$ des ensembles disjoints, chacun de cardinalité q, et soit M un sous ensemble de $X \times Y \times Z$, où M est constitué de triplets (x, y, z) tel que $x \in X$, $y \in Y$, and $z \in Z$. M contient-il un sous ensemble $M' \subseteq M$ tel que $|M'| = q$ et les triplets de M' sont deux à deux disjoints, i.e., pour deux triplets distincts $(x_1, y_1, z_1) \in M'$ et $(x_2, y_2, z_2) \in M$, nous avons $x_1 \neq x_2$, $y_1 \neq y_2$, et $z_1 \neq z_2$?

3.2.1 Problem $P2|res^t \cdot 11, s_i \in \{1, 3, 5\}, p_i = 1|C_{max}$

Théorème 2. *[57] Le problème $P2|res^t k11, s_i \in \{1, 3, 5\}, p_i = 1|C_{max}$ est NP-difficile au sens fort pour $k \geq 7$.*

Preuve. Étant donné une instance arbitraire du problème $3 - DM$, nous considérons l'instance suivante de notre problème d'ordonnancement. L'ensemble des tâches est divisé en six sous-ensembles T_M, T_Y, T_Z, T_D, T_V et T_U.

- Les $M-$Tâches, T_M, sont en correspondance avec les éléments de l'ensemble M de $3 - DM$ de sorte que chaque élément $(x, y, z) \in M$, correspond à une tâche de T_M. Soit $M_i = \{(x, y, z) \in M / x = x_i\}$ avec $\bigcup_{i=1}^{q} M_i = M$. Soit $T_{M_i} = \{Tm_{i_1}, Tm_{i_2}, \ldots, Tm_{i_{|M_i|}}\}$ est le sous ensemble de T_M correspondant à M_i.

- Les $Y-$Tâches, $T_Y = \{Ty_1, Ty_2, \ldots, Ty_q\}$, sont en correspondance avec les éléments de l'ensemble Y.

- Les $Z-$Tâches, $T_Z = \{Tz_1, Tz_2, \ldots, Tz_q\}$, sont en correspondance avec les éléments de l'ensemble Z.

- Les $D-$Tâches, $T_D = T_{D_1} \cup T_{D_2} \cup \ldots \cup T_{D_q}$ avec $T_{D_i} = \{Td_{i1}, Td_{i2}, \ldots, Td_{i_{|M_i|-1}}\}$, $|T_{D_i}| = |M_i| - 1$ et $|T_D| = |M| - q$.

- Les $V-$Tâches, $T_V = T_{V_1} \cup T_{V_2} \cup \ldots \cup T_{V_q}$ avec $T_{V_i} = \{Tv_{i1}, Tv_{i2}, \ldots, Tv_{i_{|M_i|-1}}\}$, $|T_{V_i}| = |M_i| - 1$ et $|T_V| = |M| - q$.

- Les $U-$Tâches, $T_U = T_{U_1} \cup T_{U_2} \cup \ldots \cup T_{U_q}$ avec $T_{U_i} = \{Tu_{i1}, Tu_{i2}, \ldots, Tu_{i_{|M_i|-1}}\}$, $|T_{U_i}| = |M_i| - 1$ et $|T_U| = |M| - q$.

Le nombre total de tâches est égale à $4|M| - q$. Les temps de préparation des tâches sont

égaux à 3 pour les sous-ensembles T_M et T_V, égaux à 1 pour T_Y, T_Z et T_U et égaux à 5 pour T_D. Les temps d'exécution sont égaux à 1. Le temps de traitement total des tâches est $16|M| - 8q$.

Nous considérons sept types de ressources renouvelables, R_i, $i = 1, \ldots, 7$, dont chacun est disponible en une unité. Les besoins des tâches en ressources sont détaillés dans la Table 3.2. Les ressources R_5, R_6 et R_7 marquées par des étoiles dans la Table 3.2 signifient que seules des tâches spécifiques ont besoin de ces ressources. Le détail des demandes de ces ressources est le suivant.

– Ressource R_5 est requise par les tâches Ty_i et Tm_{jl}, $(Ty_i, Tm_{jl}) \in T_Y \times T_M$, tel que y_i n'appartient pas à un triple de M_j.

– Ressource R_6 est requise par les tâches Tz_i et Tm_{jl}, $(Tz_i, Tm_{jl}) \in T_Z \times T_M$, tel que z_i n'appartient pas à un triple de M_j.

– Ressource R_7 est requise par les tâches Td_{ij} et $Tm_{i'j'}$, $(Td_{ij}, Tm_{i'j'}) \in T_{D_i} \times T_{M_{i'}}$ avec $i \neq i'$.

Type de tâches	demandes de types de ressources
$M-$Tâches	R_2, R_4, R_5^*, R_6^*, R_7^*
$Y-$Tâches	R_1, R_3, R_5^*
$Z-$Tâches	R_1, R_3, R_6^*
$D-$Tâches	R_1, R_7^*
$V-$Tâches	R_2, R_3
$U-$Tâches	R_1, R_4

TAB. 3.2 – Demandes en ressources des tâches.

L'exemple de la Figure 3.2 montre la relation entre les tâches. Une arête existe entre deux tâches T_i et T_j si elles ne nécessitent pas une même ressource. Par exemple, chaque tâche de T_{M_i} (correspondant au triplet (x_i, y_b, z_c)) est reliée aux tâches de type T_Y, T_Z et T_{D_i}, ainsi elle peut être traitée en parallèle qu'avec les tâches $Ty_b \in T_Y$ et $Tz_c \in T_Z$ ou avec les tâches de T_{D_i}.

Nous montrons que $3-DM$ a une solution si et seulement si le problème d'ordonnancement a une solution avec un makespan égal à $8|M| - 4q$.

Supposons que le problème 3-DM a une solution. Alors, on construit un ordonnancement S comme suit. Soit $T_{M'}$ un sous ensemble de tâches de T_M correspondant à M'. On ordonnance chaque tâche de $T_{M'}$ en parallèle avec leurs tâches correspondantes de T_Y et

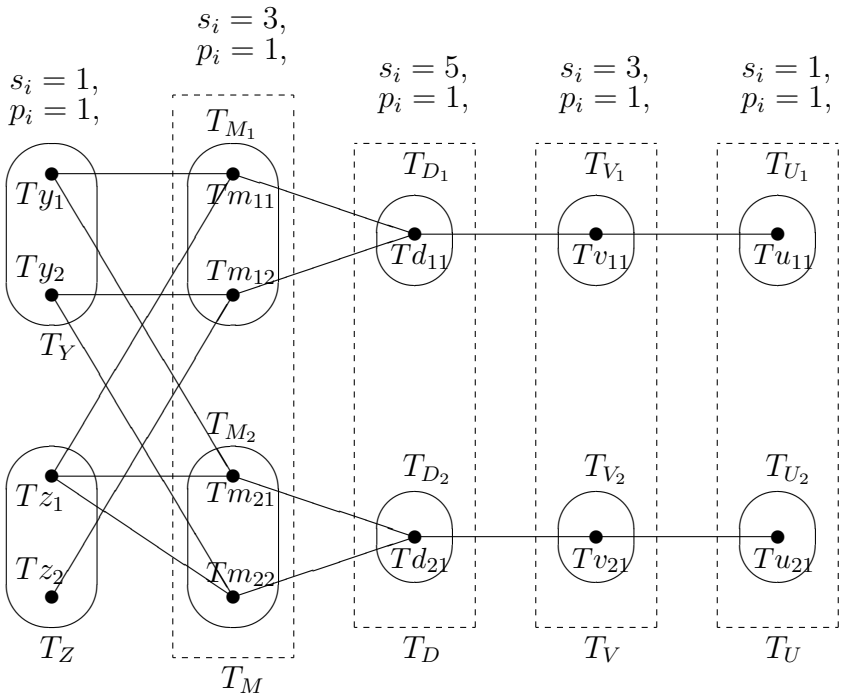


FIG. 3.2 – Exemple d'une instance du problème d'ordonnancement. $X = \{x_1, x_2\}$, $Y = \{y_1, y_2\}$, $Z = \{z_1, z_2\}$ $M = \{(x_1, y_1, z_1), (x_1, y_2, z_2), (x_2, y_1, z_1), (x_2, y_2, z_1)\}$, $M_1 = \{(x_1, y_1, z_1), (x_1, y_2, z_2)\}$, $M_2 = \{(x_2, y_1, z_1), (x_2, y_2, z_1)\}$. Toutes les tâches reliées peuvent être traitées en parallèle car elles ne nécessitent pas les mêmes ressources.

T_Z, sur les deux machines comme le montre la Figure 3.3. Le temps de traitement des tâches de $T_{M'}$ est égal à $4q$. L'ordonnancement S est complété par l'ordonnancement des tâches restantes de T_M comme suit : les tâches Tm_{ij} et $Td_{ij'}$ sont traitées en parallèle sur les deux machines, la tâche $Tv_{ij'}$ est traitée immédiatement après Tm_{ij} sur la même machine. La tâche $Tu_{ij'}$ est traitée immédiatement après $Td_{ij'}$ sur la même machine et en parallèle avec le traitement de la tâche $Tv_{ij'}$ comme le montre la Figure 3.4. Le temps total de fin de traitement est égal à $8|M| - 4q$.

(La partie hachurée correspond au temps de préparation)

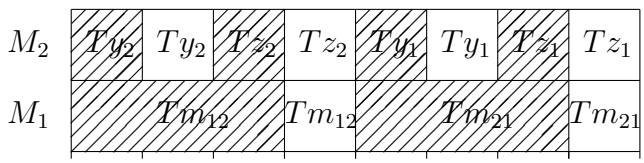


FIG. 3.3 – Ordonnancement des tâches de $T_{M'}$ ($M' = \{(x_1, y_2, z_2), (x_2, y_1, z_1)\}$) correspondant à l'exemple de la Figure 3.2, avec leurs tâches correspondantes.

Inversement, supposons qu'il existe une solution réalisable S pour le problème d'ordonnancement avec un makespan inférieur ou égal à $8|M| - 4q$. Donc, le temps total de traitement des tâches est égal à $16|M| - 8q$, alors dans l'ordonnancement S il n'y a pas

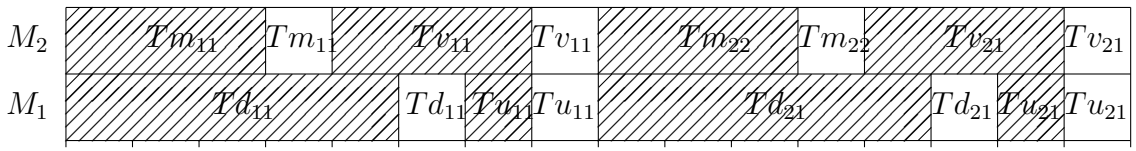


FIG. 3.4 – Ordonnancement des tâches restantes de T_M ($M \setminus M' = \{(x_1, y_1, z_1), (x_2, y_2, z_1)\}$) de l'exemple de la Figure 3.2, avec leurs tâches correspondantes des ensembles T_D, T_V and T_U.

de temps mort sur les deux machines, et $C_{max}(S) = 8|M| - 4q$. De plus, compte tenu des ressources requises par les tâches et afin d'éviter les temps mort sur les deux machines, les tâches de l'ensemble T_D ne peuvent être traitées en parallèle avec les tâches de l'ensemble T_U, car chacun de ces ensembles nécessite la même ressource R_1. Aussi, les tâches de l'ensemble T_M ne peuvent être ordonnancées en parallèle ni avec les tâches de l'ensemble T_V ni avec les tâches de l'ensemble T_U, car l'ensemble T_M partage la même ressources R_2 avec l'ensemble T_V et la même ressources R_4 avec l'ensemble T_U, mais elles peuvent être traitées en parallèle avec les tâches de l'ensemble T_D. Ainsi, les tâches des ensembles T_D, T_M, T_V et T_U, doivent être ordonnancées comme suit : une tâche Tm_{ij} de T_M est traitée en parallèle avec une tâche $Td_{ij'}$ de T_D et la tâche $Tv_{ij'}$ of T_V est ordonnancée juste après la tâche Tm_{ij} sur la même machine, et finalement la tâche $Tu_{ij'}$ de T_U est ordonnancée juste après la tâche $Td_{ij'}$ sur la même machine et en parallèle avec la tâche $Jv_{ij'}$. Par conséquent, les ($|M_i - 1|$) quadruplés sont ordonnancés comme indiqué sur la Figure 3.4 avec une durée de traitement égale à $8(|M| - q)$. Le temps restant $Tps = C_{max} - 8(|M| - q) = 8|M| - 4q - (8(|M| - q)) = 4q$ est occupé dans l'ordonnancement S par les q tâches restantes de l'ensemble T_M et les $2q$ tâches des ensembles T_Y et T_Z. Par construction de l'instance du problème d'ordonnancement, il s'en suit que ces $3q$ tâches doivent être traitées comme suit : les q tâches restantes de l'ensemble T_M sont traitées séquentiellement sur la première machine par exemple et les $2q$ tâches des ensembles T_Y and T_Z sont traitées sur la deuxième machine en parallèle avec ces q tâches restantes de T_M(voir Figure 3.3), et ceci correspond à un sous ensemble $M' \subseteq M$ avec q éléments non identiques de $T_M \cup T_Y \cup T_Z$. Il s'en suit alors que le problème $3 - DM$. ■

Remarque 1. *Notons que le problème de décision associé à* $P2|res^t k11, s_i \in \{1, 3, 5\}, p_i = 1|C_{max}$ *où* $k \geq 7$ *n'appartient pas à la classe NP. En effet, sa taille en entrée est une constante car on peut spécifier le nombre de tâches pour chaque combinaison de types de ressources et de temps de préparation et le nombre de ces combinaisons est une constante. D'autre part, une solution peut être décrite en O(n), qui n'est pas constante.*

Remarque 2. *Dans la preuve du Théorème 2, le temps de préparation de chaque tâche prend une valeur dans l'ensemble* $\{1, 3, 5\}$. *Il est facile de montrer que le résultat du Théorème 2 reste valable quand les temps de préparation des tâches prennent leurs valeurs*

dans l'ensemble $\{\alpha, 1+2\alpha, 2+3\alpha\}$*, avec* $\alpha \geq 0$*. Lorsque* $\alpha = 0$*, l'ensemble des valeurs des temps de préparation est* $\{0,1,2\}$*, le problème d'ordonnancement reste NP-difficile au sens fort si on considère que les tâches avec des temps de préparation nuls ont besoin de ressources, i.e., le temps de préparation est petit et peut être négligé* $s_i = 0$*, autrement, lorsque les temps de préparation des tâches prennent que deux valeurs* $\{1,2\}$ *le problème d'ordonnancement reste ouvert.*

Corollaire 1. *Le problème* $P2|res^t k11, s_i \in \{\alpha, 1+2\alpha, 2+3\alpha\}, p_i = 1|C_{max}$ *est NP-difficile au sens fort pour* $k \geq 7$*.*

Dans ce qui suit, nous montrons que le problème d'ordonnancement avec deux valeurs distinctes des temps de préparation et les temps d'exécution sont unitaires est NP-difficile au sens fort même si les dates de disponibilité ne prennent que deux valeurs possibles. Nous utilisons encore une fois la réduction de $3-DM$ à notre problème d'ordonnancement.

3.2.2 Problème $P2|res^t k11, s_i \in \{1,3\}, p_i = 1, r_i \in \{0,r\}|C_{max}$

Théorème 3. *[57] Le problème* $P2|res^t k11, s_i \in \{1,3\}, p_i = 1, r_i \in \{0,r\}|C_{max}$ *est NP-difficile au sens fort pour* $k \geq 5$*.*

Preuve. Étant donné une instance arbitraire de $3-DM$, soit à considérer l'instance suivante de notre problème d'ordonnancement. L'ensemble des tâches T est divisé en quatre sous-ensembles T_M, T_Y, T_Z et T_D, où T_M, T_Y, T_Z et T_D sont définis comme dans le Théorème 2. Le nombre total des tâches est égal à $2|M|+q$.

Les temps d'exécution sont égaux à 1, les temps de préparation et les dates de disponibilité des tâches sont donnés dans la Table 3.3.

Type-Tâches	s_i	r_i
T_M	3	0
T_Y	1	0
T_Z	1	0
T_D	3	4q

TAB. 3.3 – Instance du problème d'ordonnancement.

Nous considérons cinq types de ressources renouvelables, R_i, $i = 1, \ldots, 5$, dont chacun est disponible en une unité. Les besoins des tâches en ressources sont détaillés dans la Table 3.4. Les ressources R_3, R_4 et R_5 marquées par des étoiles dans la Table 3.4 signifient que

seules des tâches spécifiques ont besoin de ces ressources. Le détail des demandes de ces ressources est le suivant.

– Ressource R_3 est requise par les tâches Ty_i et Tm_{jl}, $(Ty_i, Tm_{jl}) \in T_Y \times T_M$, tel que y_i n'appartient pas à un triple de M_j.

– Ressource R_4 est requise par les tâches Tz_i et Tm_{jl}, $(Tz_i, Tm_{jl}) \in T_Z \times T_M$, tel que z_i n'appartient pas à un triple de M_j.

– Ressource R_5 est requise par les tâches Td_{ij} et $Tm_{i'j'}$, $(Td_{ij}, Tm_{i'j'}) \in T_{D_i} \times J_{M_{i'}}$ avec $i \neq i'$.

Type de tâches	demandes de types de ressources
$M-$Tâches	R_2, R_3^*, R_4^*, R_5^*
$Y-$Tâches	R_1, R_3^*
$Z-$Tâches	R_1, R_4^*
$D-$Tâches	R_1, R_5^*

TAB. 3.4 – Demandes en ressources des tâches.

L'exemple de la Figure 3.5 montre la relation entre les tâches. Une arête existe entre deux tâches T_i et T_j si elles ne partagent pas une même ressource. Par exemple, chaque tâche de T_{M_i} (qui correspond au triplet (x_i, y_b, z_c)) est reliée aux tâches de type T_Y, T_Z et T_D, ainsi elle peut être ordonnancée en parallèle seulement avec les tâches $Ty_b \in T_Y$, $Tz_c \in T_Z$ et T_{D_i}.

Nous montrons que $3-DM$ a une solution si et seulement si le problème d'ordonnancement a une solution avec un makespan inférieure ou égal à $4|M|$.

Supposons que le problème 3-DM a une solution, i.e., il existe un sous ensemble $M' \subseteq M$ tel que $|M'| = q$ et les triplets de $|M'|$ sont deux à deux disjoints. Alors, on construit un ordonnancement S comme suit. Soit $T_{M'}$ un sous ensemble de T_M correspondant à M'. On ordonnance les tâches de $T_{M'}$ en parallèle avec leurs tâches correspondantes de T_Y et T_Z, sur les deux machines, entre les instants 0 et $4q$. L'ordonnancement S est complété par l'ordonnancement de chaque tâche restante de T_M en parallèle avec une tâche de J_D à partir de $t = 4q$, comme le montre la Figure 3.6. La date de fin de l'ordonnancement S est égale à $4|M|$.

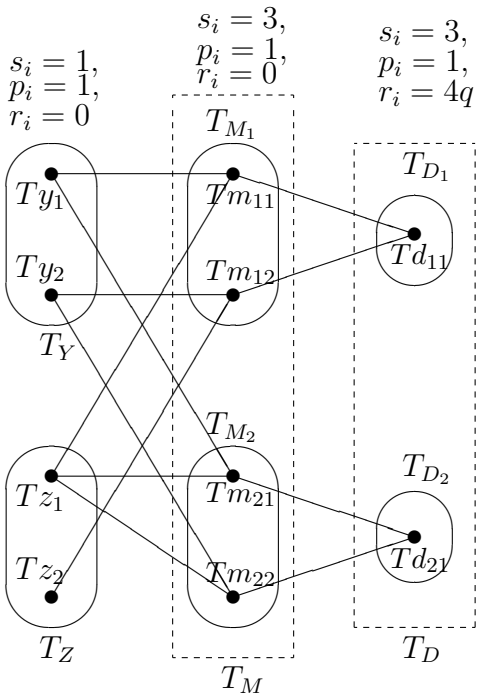


FIG. 3.5 – Exemple d'une instance du problème d'ordonnancement. $X = \{x_1, x_2\}$, $Y = \{y_1, y_2\}$, $Z = \{z_1, z_2\}$ $M = \{(x_1, y_1, z_1), (x_1, y_2, z_2), (x_2, y_1, z_1), (x_2, y_2, z_1)\}$, $M_1 = \{(x_1, y_1, z_1), (x_1, y_2, z_2)\}$, $M_2 = \{(x_2, y_1, z_1), (x_2, y_2, z_1)\}$. Toutes les tâches reliées peuvent être traitées en parallèle car elles ne nécessitent pas les mêmes ressources.

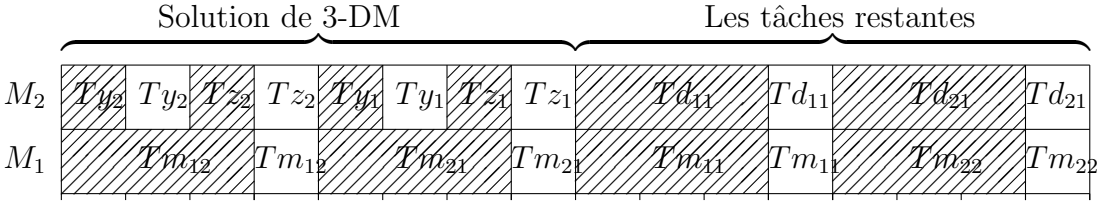


FIG. 3.6 – L'ordonnancement S correspondant à une solution du problème 3-DM de la Figure. 3.5.

Inversement, supposons maintenant qu'il existe une solution réalisable S avec durée inférieure ou égale à $4|M|$. Donc le temps de traitement total des tâches est égal à $8|M|$, alors il n'y pas de temps mort sur les deux machines, et $C_{max}(S) = 4|M|$. En outre, en ce qui concerne les besoins en ressources des tâches et afin d'éviter les temps mort sur les deux machines, nous avons les propriétés suivantes. (i) Lest tâches de T_M ne peuvent pas être traitées deux à deux en parallèle, car chacune nécessite une même ressource, ainsi ces tâches sont ordonnancées séquentiellement dans l'intervalle du temps $[0, 4|M|]$; on suppose que ces tâches sont ordonnancées sur la première machines M_1.(ii) Les tâches de l'ensemble T_D sont disponible à $t = 4q$, et leurs temps de traitement est égal à $4|M| - 4q$, et que ces tâches ne partagent pas de ressources avec les tâches de l'ensemble T_M, ainsi le seul ordonnancement possible des tâches de T_D, est de les traiter sur la deuxième ma-

chine dans l'intervalle du temps $[4q, 4|M|]$. (iii) Les tâches restantes de l'ensemble T_Y et T_Z sont ordonnancées sur la deuxième machine dans l'intervalle du temps $[0, 4q]$ (comme indiqué sur la Figure 3.6), ceci correspond à un sous-ensemble $M' \subseteq M$ avec q triplets non identiques de $T_M \cup T_Y \cup T_Z$. Il s'en suit que le problème $3 - DM$ a une solution. ■

Les Remarques 1 et 2 restent valables pour le problème $P2|res^t k11, s_i \in \{1, 3\}, p_i = 1, r_i \in \{0, r\}|C_{max}$.

Corollaire 2. *Le problème* $P2|res^t k11, s_i \in \{\alpha, 1 + 2\alpha\}, p_i = 1, r_i \in \{0, r\}|C_{max}$ *est NP-difficile au sens fort pour* $k \geq 5$.

3.3 Cas particuliers

Dans cette section nous discutons de deux cas particuliers avec des temps d'exécution identiques.

3.3.1 Problème $Pm|res^t 111, s_i \geq 1, p_i = 1|C_{max}$

Pour le problème $Pm|res^t 111, s_i \geq 1, p_i = 1|C_{max}$, une seule ressource est disponible pour la préparation de toutes les tâches. Cela nous permet de proposer l'algorithme polynomial suivant.

Algorithme List-A
INPUT : $s_1, s_2, s_3..., s_n$
OUTPUT : $t_1, t_2, t_3..., t_n$, C_{max} (dates de début et le makespan)
Initialisation. $t_1 := 0$;
Traitement. Pour $i := 2$ à n Faire $t_i := t_{i-1} + s_{i-1}$;
Solution optimale. $C_{max} := t_n + s_n + 1$

Théorème 4. *[57] L'Algorithme List−A construit une solution optimale pour le problème* $Pm|res^t 111, s_i, p_i = 1|C_{max}$ *en* $O(n)$.

Preuve. Nous avons une seule ressource disponible en une unité, donc la préparation simultanée de deux tâches sur les machines n'est pas possible. Et comme les temps d'exécution de toutes les tâches sont égaux à 1, alors la préparation des tâches se fera en alternée sur les machines. Ainsi, le makespan est égal à $\sum_{i=1}^{n} s_i + 1$ (le temps total de préparation plus 1) et cela coïncide avec la borne inférieure $LB2$ du problème (voir Proposition 3). ■

Exemple 4. *Soit à traiter 7 tâches T_1,..., T_7 sur deux machines identiques avec un seul type de ressources disponible en une unité. Les temps de préparation des tâches sont donnés par la Table 3.5.*

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7
s_i	2	2	4	1	2	3	2

TAB. 3.5 – Temps de préparation des tâches.

L'exécution de l'algorithme donne : $C_{max} = t_7 + s_7 + 1 = 14 + 2 + 1 = 17$.

Un ordonnancement optimal est représenté par la Figure 3.7 :

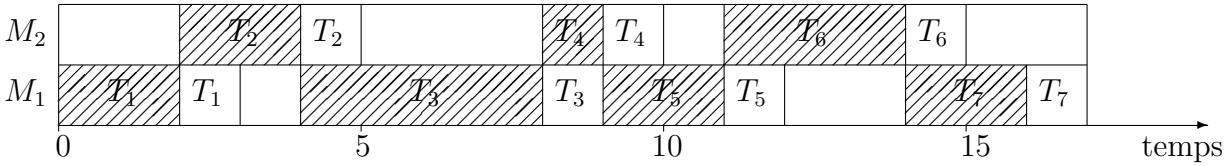


FIG. 3.7 – Solution optimale fournie par l'Algorithme Liste-A.

3.3.2 Problème $P2|res^t \cdot 11, s_i = s, p_i = p|C_{max}$

Dans ce cas toutes les tâches ont une même durée de préparation s, et une même durée d'exécution p. Pour une instance donnée du problème $P2|res^t \cdot 11, s_i = s, p_i = p|C_{max}$, nous construisons un graphe $G = (T, E)$ où T est l'ensemble de sommets qui correspondent aux tâches qui doivent être ordonnancées, et une paire de sommets (T_i, T_j) est dans E si et seulement si les besoins en ressources par les tâches correspondantes à ces sommets ne coïncident pas, c'est-à-dire, $d_\gamma(T_i) + d_\gamma(T_j) \leq 1$, $\forall \gamma = 1, \ldots, k$ ($d_\gamma(T_i)$ décrit la demande de la tâche T_i en ressource R_γ, et $d_\gamma(T_i) = 1$ si la tâche T_i nécessite la ressource R_γ, 0 sinon).

Soit M un couplage maximum dans le graphe G. Il est évident que si M est un couplage parfait alors une solution optimale pour le problème $P2|res^t \cdot 11, s_i = s, p_i = p|C_{max}$ peut être trouvée en ordonnançant les tâches couplées en parallèle sur les deux machines, et dans n'importe quel ordre à partir de l'instant 0, voir la Figure 3.8.

Supposons que M n'est pas un couplage parfait, et que l'ensemble Q correspond aux tâches non couplées dans le graphe G. Soit h la cardinalité de l'ensemble Q. Nous distinguons deux cas :

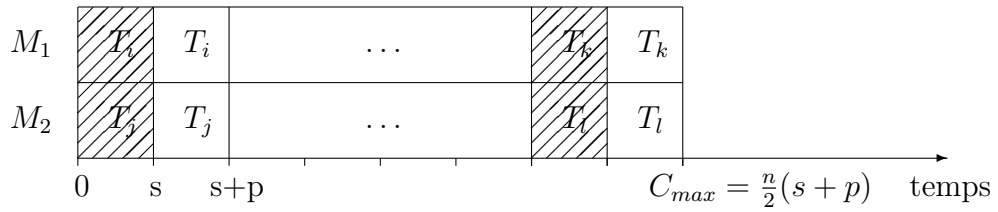


FIG. 3.8 – Ordonnancement optimal si M est un couplage parfait.

1. $s \leq p$. Soit T_i la première tâche ordonnancée de l'ensemble Q dans la solution optimale, et supposons qu'elle soit traitée sur la machine M_1. Puisque les deux machines traitent les tâches de M, donc il n'y a pas de temps mort sur ces deux dernières avant le traitement de la tâche T_i. En outre, au cours de s unités du temps de préparation de T_i sur la machine M_1, la machine M_2 est libre, alors $\lfloor \frac{n}{2} \rfloor (s+p) + s$ est une borne inférieure pour la solution optimale. Considérons une solution réalisable S dans laquelle les tâches couplées sont ordonnancées sur les deux machines à partir de $t = 0$, et S est complétée par les tâches de l'ensemble Q qui sont ordonnancées alternativement sur les deux machines M_1 et M_2 le plus tôt possible. Si h est un nombre pair, il est facile de vérifier que la solution S a une durée totale de traitement égale à $\frac{n}{2}(s+p) + s$, sinon la durée totale de traitement de S est égale à $(\lfloor \frac{n}{2} \rfloor + 1)(s+p)$ qui est aussi une borne inférieure pour le problème $P2|res^t \cdot 11, s_i = s, p_i = p|C_{max}$, donc S est une solution optimale, voir la Figure 3.9.

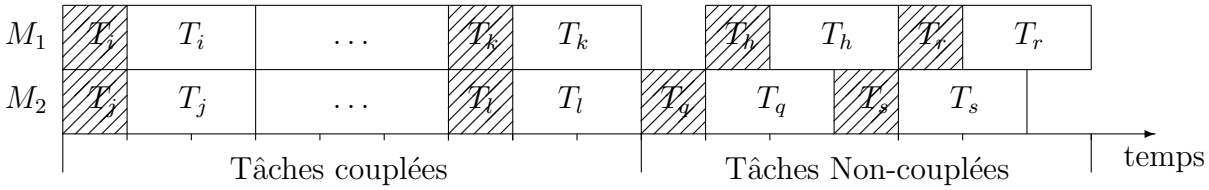


FIG. 3.9 – Ordonnancement optimal quand $s \leq p$ et $Q \neq \emptyset$.

2. $s > p$. Nous commençons par déterminer une borne inférieure pour la solution optimale, ensuite nous montrons qu'une solution réalisable existe avec une valeur du makespan égale à cette borne inférieure. Considérons le problème relaxé dans lequel tous les temps d'exécution des tâches de l'ensemble Q sont égaux à zéro, sauf pour la tâche qui sera traitée à la dernière position. Il est clair que la solution optimale du problème relaxé est une borne inférieure pour le problème $P2|res^t \cdot 11, s_i = s, p_i = p|C_{max}$. La solution optimale du problème relaxé est obtenue par le traitement des tâches couplées en parallèle sur les deux machines et le traitement des tâches de l'ensemble Q en alterné sur les deux machines. La valeur du makespan de cette solution optimale est égale à $|M|(s+p)+hs+p$ qui est une borne inférieure pour le problème $P2|res^t \cdot 11, s_i = s, p_i = p|C_{max}$. Une solution réalisable est obtenue comme suit. Ordonnancer les tâches couplées en parallèle

sur les deux machines, suivies des tâches de l'ensemble Q, dans lequel à chaque instant l'une des deux machines est disponible, ordonnancer sur cette machine les tâches de l'ensemble Q qui n'ont pas encore été traitées. Il est claire que la date de fin de traitement de cette solution est égale à $|M|(s+p)+hs+p$, donc S est la solution optimale, voir Figure 3.10.

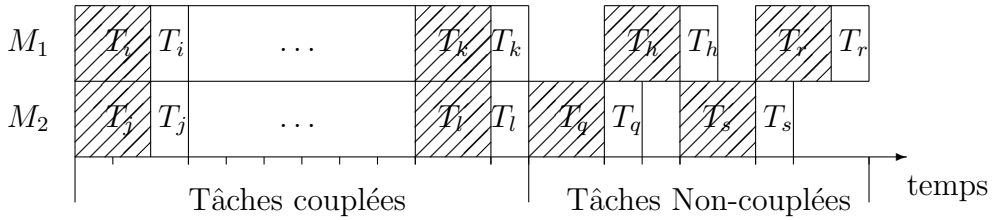


FIG. 3.10 – Ordonnancement optimal quand $s > p$ et $Q \neq \emptyset$.

	Algorithme - MM
1.	Construire le graphe $G = (T, E)$.
2.	Trouver un couplage maximum M dans G. Soit Q l'ensemble des tâches non couplées et soit h la cardinalité de Q.
3.	Les tâches couplées sont traitées en parallèle sur les deux machines.
4.	Si $Q \neq \emptyset$, ordonnancer les tâches de l'ensemble Q alternativement sur les deux machines, le plus tôt possible.
5.	$C_{max}=\begin{cases} \frac{n}{2}(s+p) & \text{si } Q=\emptyset \\ (\lfloor \frac{n}{2} \rfloor +1)(s+p) & \text{si } Q\neq\emptyset,\ h \text{ est impair et } s\leq p \\ \frac{n}{2}(s+p)+s & \text{si } Q\neq\emptyset,\ h \text{ est pair et } s\leq p \\ \lvert M\rvert(s+p)+h.s+p & \text{si } Q\neq\emptyset \text{ et } s>p \end{cases}$

Théorème 5. *[57] L'Algorithme − MM résout le problème* $P2|res^t\cdot 11, s_i = s, p_i = p|C_{max}$ *en* $O(n^{2.5})$.

Preuve. L'algorithme présenté ci-dessus est basé sur le calcul d'un couplage maximum dans le graphe G. Si un couplage parfait existe dans G, alors une solution optimale pour le problème $P2|res^t\cdot 11, s_i = s, p_i = p|C_{max}$ est construite par le traitement des tâches couplées de M dans un ordre arbitraire, et la valeur du makespan est égale à $\frac{n}{2}(s+p)$. Autrement, tout dépend de la valeur du temps de préparation s, du temps d'exécution p, et de la valeur de h. Nous exposons dans les cas (1.), (2.), des bornes inférieures pour $P2|res^t\cdot 11, s_i = s, p_i = p|C_{max}$, alors on propose une solution réalisable avec une durée totale de l'ordonnancement égale à une de ces bornes inférieures. La complexité de cette approche est clairement dominée par le calcul de couplage maximum en $O(n^{2.5})$ [42]. Ainsi, le résultat du Théorème est établi. ■

Exemple 5. *Considérons une instance du problème* $P2|res^t\cdot 11, s_i = s, p_i = p|C_{max}$ *avec deux machines et 8 tâches. Les temps de préparation sont égaux à 1 et les durées d'exé-*

cution sont égaux à 2, le nombre de types de ressources est égal à 2. Les demandes en ressources pour chaque tâche sont données dans la Table 3.6.

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
$R(T_i)$	[1, 0]	[0, 1]	[1, 0]	[1, 1]	[1, 0]	[1, 1]	[1, 1]	[0, 1]

TAB. 3.6 – Demandes en types de ressources.

L'exécution de l'algorithme donne :

1. *On construit le graphe $G = (T, E)$ représenté par la Figure. 3.11.*

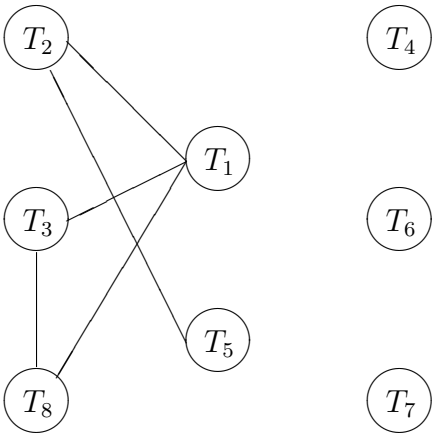


FIG. 3.11 – Graphe $G = (T, E)$ correspondant à l'exemple 5.

2. *Le couplage maximum : $M = \{(T_1; T_2), (T_3; T_8)\}$.*

3. *L'ensemble Q contient les tâches T_4, T_5, T_6 et T_7, $h = 4$*

4. *$Q \neq \emptyset$, h est pair et $s \leq p$ donc $C_{max} = \frac{n}{2}(s + p) + s = 13$.*

5. *L'ordonnancement optimal est donné par la Figure. 3.12.*

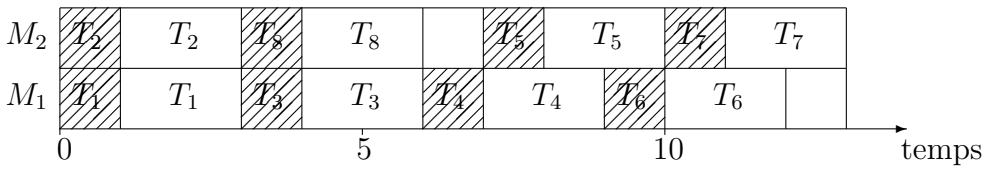


FIG. 3.12 – Solution optimale fournie par l'Algorithme-MM.

Dans la section suivante, nous considérons le problème général $P|res^t \cdots |C_{max}$, dont nous présentons deux bornes inférieures.

3.4 Bornes inférieures

Les bornes inférieures sont utiles pour l'analyse comparative des solutions heuristiques. La première borne inférieure proposée ici est basée sur la borne inférieure du problème classique d'ordonnancement des machines parallèles.

Proposition 2. *[57]* $LB1 = \max\{\lceil \frac{1}{m} \sum_{i=1}^{n}(p_i + s_i)\rceil, \max_{i=1}^{n}\{p_i + s_i\}\}$ *est une borne inférieure pour le problème* $Pm|res^t \cdots |C_{max}$.

La seconde borne inférieure exploite les ressources nécessaires par les tâches. Soit S un sous ensemble de T tel que $\forall(T_i, T_j) \in S$, il existe un ressource R_γ avec $d_\gamma(T_i)+d_\gamma(T_j) = 2$ et soit $W(S) = \sum_{T_i \in S} s_i$.

Proposition 3. *[57]* $LB2 = W(S) + \min_{T_i \in T}\{p_i\}$ *est une borne inférieure pour le problème* $Pm|res^t \cdots |C_{max}$.

En effet, les tâches de l'ensemble S ne peuvent être préparées en parallèle sur les machines, $W(S)$ ajouté à la date de fin de traitement de la dernière tâche traitée forment une borne inférieure pour le C_{max}.

Pour avoir une bonne valeur de $LB2$, on doit maximiser $W(S)$. Il s'agit donc de trouver un stable de poids maximum dans un graphe (comme définie dans la Section 3.3.2), qui est un problème NP-difficile au sens fort. Trois heuristiques de type glouton $GWMIN$, $GWMAX$, et $GWMIN2$ ont été proposées dans [67], pour déterminer un stable de poids maximum.

En combinant les deux bornes inférieures $LB1$ et $LB2$, on obtient une borne inférieure globale pour le C_{max} : $LB = \max\{LB1, LB2\}$.

Chapitre 4

Approche de résolution pour le problème $Pm|res^t \cdot 11|C_{max}$

Dans ce chapitre, nous présentons les heuristiques développées pour résoudre le problème $P2|res^t \cdot 11|C_{max}$, ensuite nous présentons un ensemble d'expérimentations numériques pour évaluer leurs performances. Nous développons par la suite des métaheuristiques dont nous évaluons leurs performances par rapport à la borne inférieure.

4.1 Approche heuristique

Le problème d'ordonnancement étudié est NP-difficile et il n'existe pas une méthode exacte capable de le résoudre efficacement pour des instances de grande taille, et pour cette raison nous avons opté pour une approche heuristique pour sa résolution. Pour cela, nous avons élaboré des heuristiques dont nous testons les performances par rapport à la borne inférieure LB. Nous proposons deux types d'heuristiques qui reposent sur l'utilisation de différentes listes de tri pour chacune d'elle. Avant de passer à leurs descriptions, on définit les règles proposées.

Les six premières listes sont basées sur le rangement des tâches selon l'ordre croissant ou décroissant des temps de préparation et d'exécution. Les six autres listes sont basées sur le graphe $G = (T, E)$ défini précédemment (voir Paragraphe 3.3.2).

Définition 1. *On appelle nombre de compatibilité d'une tâche T_i, le nombre de tâches qui peuvent être préparées en parallèle avec elle i.e., $d_\gamma(T_i) + d_\gamma(T_j) \leq 1$, $\forall \gamma = 1, \ldots, k$, $\forall i, j = 1, \ldots, n$.*

- LET : Les tâches sont rangées dans l'ordre décroissant des temps d'exécution.

- LPT : Les tâches sont rangées dans l'ordre décroissant des temps de préparation.

- $LPET$: Les tâches sont rangées dans l'ordre décroissant de la somme de leurs temps de préparation et d'exécution.

- SET : Les tâches sont rangées dans l'ordre croissant des temps d'exécution.

- SPT : Les tâches sont rangées dans l'ordre croissant des temps de préparation.

- $SPET$: Les tâches sont rangées dans l'ordre croissant de la somme de leurs temps de préparation et d'exécution.

- SCN : Ordre croissant des nombres de compatibilité des tâches.

- LCN : Ordre décroissant des nombres de compatibilité des tâches.

- $SCN1$: T_i est la tâche ayant le plus petit nombre de compatibilité.
  T_l $(l = 2, \ldots, n)$ est la tâche ayant le plus petit nombre de compatibilité dans l'ensemble des tâches $T \backslash \{T_1, ..., T_{l-1}\}$.

- $LCN1$: T_i est la tâche ayant le plus grand nombre de compatibilité.
  Tl $(l = 2, \ldots, n)$ est la tâche ayant le plus grand nombre de compatibilité dans l'ensemble des tâches $T \backslash \{T_1, ..., T_{l-1}\}$.

- $SCN2$: T_i est la tâche ayant le plus petit nombre de compatibilité.
  T_l $(l = 2, \ldots, n)$ est la tâche ayant le plus petit nombre de compatibilité dans l'ensemble des tâches $\{T_1, ..., T_{l-1}\}$.

- $LCN2$: T_i est la tâche ayant le plus grand nombre de compatibilité.
  T_l $(l = 2, \ldots, n)$ est la tâche ayant le plus grand nombre de compatibilité dans l'ensemble des tâches $\{T_1, ..., T_{l-1}\}$.

Heuristique $HS - r$

Cette heuristique est exécutée en deux étapes. La première étape commence par le rangement des tâches selon un des ordres cité précédemment et la seconde étape utilise la sélection Sérielle [18]. Le principe de la sélection Sérielle est le suivant : la tâche de plus haute priorité est sélectionnée et ordonnancée sur la première machine libre dès que possible en respectant les contraintes de ressources.

Sélection Sérielle

Début
Tantque $T \neq \emptyset$
Faire - Choisir une tâche $T_i \in T$ de plus haute priorité ;
- Ordonnancer T_i le plus tôt possible en respectant les contraintes de ressources ;
- $T := T \setminus \{T_i\}$;
Fait
Fin.

Heuristique $HP - r$

Cette heuristique débute par la sélection des tâches selon un ordre donné r décrit précédemment, ensuite appliquer la sélection Parallèle [18]. Le principe de la Sélection Parallèle est le suivant : à chaque instant t où une des machines est disponible, traiter parmi les tâches prêtes (celles qui utilisent moins d'unités que la quantité disponible d'une ressource), la tâche de plus haute priorité.

Sélection Parallèle

Début
$t := 0$;
Tantque $T \neq \emptyset$
Faire Soit $\overline{U} \subseteq T$ l'ensemble des tâches disponibles à l'instant t
Si $\overline{U} \neq \emptyset$
Alors - Choisir une tâche T_i de plus haute priorité ;
- $t_i = t$;
- $T := T \setminus \{T_i\}$;
Sinon déterminer le plus petit instant t où une tâche devient disponible.
FinSi
Fait
Fin.

Les étapes de ces heuristiques sont donc :

1. Trier les tâches selon un ordre r.
2. Appliquer la sélection Sérielle (sélection Parallèle).

La complexité de ces heuristiques est de $O(nlogn)$, en effet, la première étape est une opération de tri de complexité $O(nlogn)$ et la deuxième étape est une application de la sélection Sérielle (sélection Parallèle) qui est de complexité $O(n)$.

Exemple 6. *Soit le problème avec deux machines, sept tâches et trois types de ressources. Les paramètres liés aux tâches sont donnés par la Table 4.1.*

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7
s_i	3	1	2	1	2	3	1
p_i	1	2	2	1	3	1	2
$R(T_i)$	[1, 0, 0]	[0, 1, 0]	[1, 0, 1]	[0, 1, 1]	[1, 1, 1]	[0, 0, 1]	[0, 1, 0]

TAB. 4.1 – Paramètres liés aux tâches.

Le graphe associé à l'exemple 6 est représenté par la Figure 4.1.

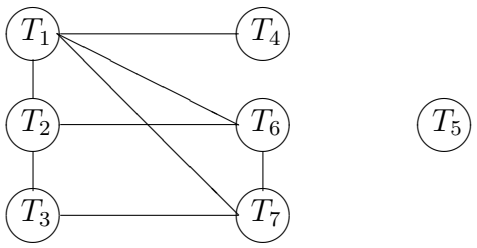


FIG. 4.1 – Graphe $G = (T, E)$ correspondant à l'exemple 6.

En appliquant les règles proposées on obtient les listes suivantes :

$$
\begin{array}{ll}
LET = \{T_5, T_2, T_3, T_7, T_1, T_4, T_6\}, & LPT = \{T_1, T_6, T_3, T_5, T_2, T_4, T_7\}. \\
LPET = \{T_5, T_1, T_3, T_6, T_2, T_7, T_4\}, & SET = \{T_1, T_4, T_6, T_2, T_3, T_7, T_5\}. \\
SPT = \{T_2, T_4, T_7, T_3, T_5, T_1, T_6\}, & SPET = \{T_4, T_2, T_7, T_1, T_3, T_6, T_5\}. \\
SCN = \{T_5, T_4, T_3, T_2, T_6, T_7, T_1\}, & LCN = \{T_1, T_2, T_6, T_7, T_3, T_4, T_5\}. \\
SCN1 = \{T_5, T_4, T_3, T_2, T_1, T_6, T_7\}, & LCN1 = \{T_1, T_2, T_7, T_3, T_4, T_5, T_6\}. \\
SCN2 = \{T_5, T_4, T_2, T_7, T_3, T_6, T_1\}, & LCN2 = \{T_1, T_2, T_6, T_7, T_3, T_4, T_5\}.
\end{array}
$$

Les Figures suivantes 4.2, 4.3, 4.4, 4.5 représentent quelques solutions obtenues par les heuristiques $HS-r$ et $HP-r$.

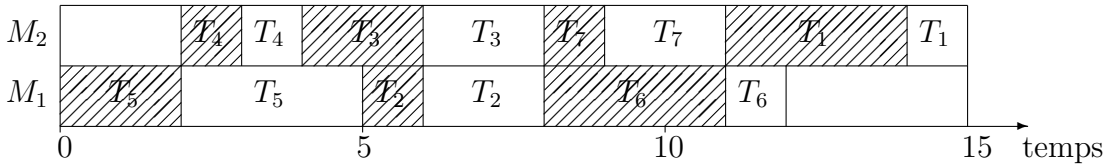


FIG. 4.2 – Solution obtenue par l'heuristique $HS-SCN$.

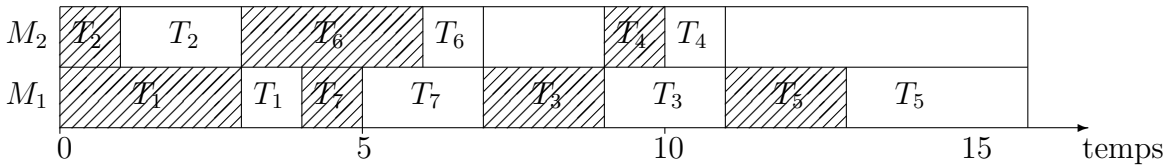


FIG. 4.3 – Solution obtenue par l'heuristique $HS-LCN$.

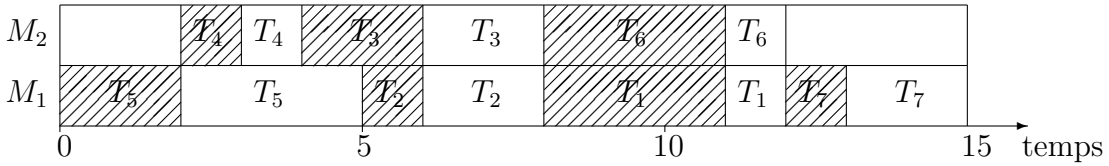


FIG. 4.4 – Solution obtenue par l'heuristique $HP-SCN1$.

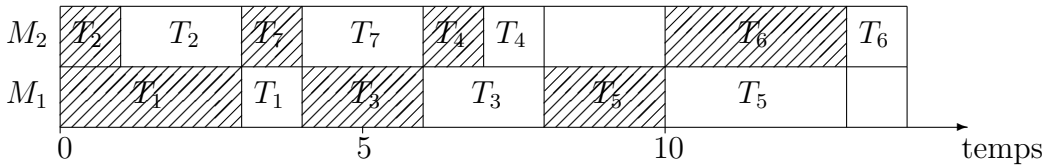


FIG. 4.5 – Solution obtenue par l'heuristique $HP-LCN1$.

4.2 Expérimentations numériques

Pour évaluer l'efficacité des heuristiques, nous avons effectué des expériences numériques. Pour cela, nous avons considéré trois types d'instances distinctes. Le premier type est caractérisé par des petits temps de préparation et de grands temps d'exécution, dans ce cas les temps de préparation et d'exécution sont générés aléatoirement comme suit : (i) $s_i \in [1,5]$, $p_i \in [5,10]$, (ii) $s_i \in [1,10]$, $p_i \in [10,50]$, (iii) $s_i \in [10,50]$, $p_i \in [50,100]$. Dans le second type, nous générons des grands temps de préparation et de petits temps

d'exécution. Les temps de préparation et d'exécution des tâches sont générés de façon aléatoire suivant une loi uniforme comme suit : (i) $p_i \in [1,5]$, $s_i \in [5,10]$, (ii) $p_i \in [1,10]$, $s_i \in [10,50]$, (iii) $p_i \in [10,50]$, $s_i \in [50,100]$. Pour le troisième type, les temps de préparation et d'exécution sont générés dans le même intervalle comme suit : (i) $s_i, p_i \in [1,10]$, (ii) $s_i, p_i \in [10,50]$, (iii) $s_i, p_i \in [50,100]$. Nous avons généré 100 instances aléatoires pour chaque problème avec différents paramètres. Nous avons traité des problèmes avec 2, 5 et 10 machines de tailles $n = 10$, $n = 50$, $n = 100$, $n = 500$ et $n = 1000$. Pour chaque valeur de n, nous avons testé différentes valeurs de k (le nombre de types de ressources) où k prend ses valeurs dans $\{1,2,5,7\}$. Pour chaque instance, les demandes en ressources par les tâches sont générées aléatoirement.

La performance des heuristiques basées sur les différentes règles développées dans la section précédente ont été comparé avec la borne inférieure. Pour chaque instance résolue, nous reportons le nombre de fois où une heuristique H fournit de meilleures solutions par rapport aux autres noté $\#Best$, et le nombre de fois où elle coïncide avec la borne inférieure noté $\#LB$. La performance d'une heuristique donnée est mesurée par l'écart relatif entre la valeur de la solution obtenue C_{max} et la borne inférieure LB ($Dev = \frac{C_{max}(H)-LB}{LB}$). Pour chaque type d'instances nous présentons les déviations moyenne (Dev_{moy}) et maximale (Dev_{max}). Pour chaque Table nous présentons les performances des meilleures heuristiques.

Une analyse de ces expérimentations numériques nous permet de conclure que les heuristiques suivantes $HP-SPT$, $HP-LCN$, $HP-LCN1$, $HP-SCN2$, $HP-LCN2$, $HS-SET$, $HS-SPT$, $HS-SCN$, $HS-LCN$, $HS-SCN1$, $HS-LCN1$ n'ont pas donnés de bons résultats et que les heuristiques $HP-LET$, $HP-LPT$, $HP-LPET$, $HP-SET$, $HP-SPET$, $HP-SCN$, $HP-SCN1$, $HS-LET$, $HS-LPT$, $HS-LPET$, $HS-SPET$, $HS-SCN2$, $HS-LCN2$ donnent de meilleures solutions et leur efficacité dépend du :
- nombre de machines m.
- nombre de tâches n.
- nombre de type de ressources k.
- la valeur du temps d'exécution par rapport à la valeur du temps de préparation.
- les intervalles où appartiennent les temps d'exécution et de préparation.

D'après les expérimentations numériques concernant les problèmes à deux machines, nous avons observé que les heuristiques basées sur la sélection Parallèle performent mieux que celles basées sur la sélection Sérielle. Alors, nous avons décidé de ne présenter que les heuristiques basées sur la sélection Parallèle, pour différentes valeurs de types de ressources. Pour ces expérimentations, pour chacune des Tables A.1, A.2, A.3, nous ne présentons

que les résultats des heuristiques dominantes.

Quand les temps d'exécution prennent des valeurs plus grand que les temps de préparation (voir la Table A.1), les meilleures solutions sont partagées entre les six heuristiques $HP-SPET$, $HP-LPET$, $HP-LET$, $HP-LPT$, $HP-SCN$ et $HP-SCN1$. En effet, pour le premier rangement des temps de préparation et d'exécution, on peut observer que l'heuristique $HP-LPET$ fournit des solutions avec une moyenne significativement meilleure que celles des autres heuristiques pour toutes les instances à $n \leq 100$ et les instances à $n \geq 500$ avec $k = 1, 2$. Pour $n = 1000$, $k = 1$ ($k = 2$), l'heuristique $HP-LPET$ est meilleure dans 84% (83%) des cas dont 59% (49%) sont optimales. Pour les autres instances, ce sont les heuristiques $HP-LPT$, $HP-SCN$ et $HP-SCN1$ qui fournissent de bons résultats. Concernant le deuxième et le troisième rangement, l'heuristique $HP-LPET$ génère de bonnes solutions pour les instances de petite taille et pour les autres cas se sont les heuristiques $HP-LPT$, $HP-SCN$ et $HP-SCN1$ qui semblent meilleures avec une supériorité de l'heuristique $HP-SCN1$ pour la résolution des instances à $n \geq 500$ et $k = 7$.

A la lecture de la Table A.2, on constate la performance de l'heuristique $HP-LPET$ par rapport aux autres heuristiques pour les instances à 10 tâches et les instances à $n = 50$ avec $k = 1$ pour les trois rangements des temps de préparation et d'exécution. Pour la résolution des instances à $n \leq 100$ avec $k = 2$, c'est l'heuristique $HP-SCN1$ qui est meilleure et pour $k = 5, 7$, c'est l'heuristique $HP-SCN$ qui trouve les meilleures solutions. Pour $n \geq 500$, les meilleures solutions sont partagées entre les heuristiques $HP-SCN$ et $HP-SCN1$ avec des déviations maximale et moyenne inférieures à 13% et 8% respectivement.

Quand les temps de préparation et d'exécution prennent leurs valeurs dans un même intervalle (voir Table A.3), les meilleures solutions sont partagées entre les cinq heuristiques $HP-SPET$, $HP-LPET$, $HP-LPT$, $HP-SCN$ et $HP-SCN1$. En effet, l'heuristique $HP-LPET$ est efficace pour la résolution de toutes les instances à 10 tâches, ainsi que pour les instances à $n \geq 50$ et $k = 1$ pour le premier rangement des temps de préparation et d'exécution. Pour $n = 1000$ et $k = 1$, la solution optimale est trouvée par l'heuristique $HP-LPET$ dans 56% des cas sur 76%. On remarque aussi que l'heuristique $HP-LPT$ génère de bons résultats pour les instances à $n \geq 50$ et $k \leq 2$ pour le premier et le deuxième rangement. Pour le troisième rangement, l'heuristiques $HP-SPET$ donne de bonnes solutions pour les instances à $n \geq 50$ et $k = 5, 7$ avec une déviation moyenne inférieure à $0,4\%$ pour $n = 1000$. Pour les instances avec $k = 1, 2$, les meilleures solutions sont trouvées par les heuristiques $HP-SCN$ et $HP-SCN1$.

D'après les expérimentations numériques effectuées pour les problèmes à cinq machines, nous avons remarqué l'apparition de quelques heuristiques basées sur la sélection Sérielle pour le second et le troisième type d'instances. Les Tables suivantes reportent ces résultats.

A la lecture des Tables A.4, A.5 et A.6, il apparaît clairement que pour les instances de petites tailles se sont les heuristiques $HP-LPET$, $HP-LPT$ et $HP-LET$ qui génèrent de bons résultats. Nous observons aussi que l'heuristique $HP-LPT$ fournit de bonnes solutions si on a qu'un seul type de ressources disponible pour la préparation de toutes les tâches. L'heuristique $HP-SPET$ génère de bons résultats pour les instances à $n \geq 500$ et $k \geq 2$ pour le deuxième rangement des temps de préparation et d'exécution. Les heuristiques $HP-SCN$ et $HP-SCN1$ sont généralement efficaces pour la résolution de la plus part des instances.

D'après les Tables A.7, A.8 et A.9, les meilleures solutions trouvées pour la résolution des instances à $k = 1$ sont partagées entre les heuristiques $HS-LPET$ et $HS-LET$ avec une supériorité de l'heuristique $HS-LET$ quand $n \leq 100$ et une supériorité de l'heuristique $HS-LPET$ quand $n \geq 500$. L'heuristique $HP-SET$ fournit de bons résultats pour les instances à $n \geq 100$ et $k = 2$ (pour le premier et le troisième rangement), elle est même meilleure à 100% des cas pour 1000 tâches avec une déviation moyenne inférieure à $2,7\%$. Pour les autres cas, les meilleures solutions sont données par les heuristiques $HP-SCN$ et $HP-SCN1$ avec des déviations moyennes ne dépassant pas $4,2\%$ et $4,7\%$ respectivement.

Quand les temps de préparaion et d'exécution prennent leurs valeurs dans un même intervalle (voir les Tables A.10, A.11, A.12), les meilleures solutions trouvées pour toutes les instances avec un seul type de ressources sont données par les heuristiques basées sur la sélection Sérielle $HS-SPET$, $HS-LPET$ et $HS-LET$ avec une supériorité de $HS-LPET$. L'heuristique $HP-LET$ est efficace pour la résolution des instances de petite taille. On peut observer que l'heuristique $HP-SET$ donne de bonnes solutions pour les instances à $n \geq 500$ et $k = 2$ pour le premier et le deuxième rangement des temps de préparation et d'exécution. Pour les cas restants, les meilleures solutions sont trouvées soit par l'heuristique $HP-SCN$, soit par l'heuristique $HP-SCN1$.

4.3 Approche métaheuristique

Cette section se concentre sur la présentation des approches métaheuristiques proposées pour l'amélioration des heuristiques. Les métaheuristiques sont des heuristiques stochastiques itératives qui progressent vers un optimum local en se comportant comme des algorithmes de recherche, ces méthodes sont généralement hybridées avec une recherche locale. Elles sont souvent inspirées par des analogies avec des phénomènes de la nature, la physique (Recuit simulé), la biologie (algorithmes évolutionnaires) et l'éthologie (colonies de fourmis). Parmi ces méthodes, nous allons adopter deux d'entre elles : le recuit simulé et l'algorithme génétique.

4.3.1 Recuit simulé

Le recuit simulé est une métaheuristique inspirée d'un processus utilisé en métallurgie. Ce processus alterne des cycles de refroidissement lent et de réchauffage (recuit) qui tendent à minimiser l'énergie du matériau. Elle est aujourd'hui utilisée en optimisation pour trouver les extrémaux d'une fonction. Elle a été mise au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983 [46], et indépendamment par V. Cerny en 1985 [19].

Le recuit simulé s'appuie sur l'algorithme de Metropolis, qui permet de décrire le comportement d'un système en équilibre thermodynamique à une certaine température T, partant d'une configuration donnée (solution initiale). Par analogie avec le processus physique, la fonction objectif à minimiser deviendra l'énergie E du système.

Les étapes du recuit simulé

Le recuit simulé commence par une solution initiale qui peut être prise au hasard dans l'espace des solutions possibles. A chaque itération de l'algorithme une modification élémentaire de la solution est effectuée. Cette modification entraine une variation ΔC de la fonction objectif. Si cette variation a pour effet de diminuer la fonction objectif (ou énergie), elle est appliquée à la solution courante. Sinon, elle est acceptée avec une probabilité $e^{\frac{(-\Delta C)}{T}}$. On itère ensuite selon ce procédé en gardant la température constante. Lorsque le système a atteint un équilibre thermodynamique, on abaisse la température du système avant d'effectuer une nouvelle série de transformations. On parle alors de paliers de température. Si la température a atteint un seuil assez bas ou que le système devient

figé, l'algorithme s'arrête. La température joue un rôle important. A haute température, le système est libre de se déplacer dans l'espace des solutions (exp(-DE/T) proche de 1) en choisissant des solutions ne minimisant pas forcément l'énergie du système. A basse température, les modifications baissant l'énergie du système sont choisies, mais d'autres peuvent être acceptées, empêchant ainsi l'algorithme de tomber dans un minimum local.

Adaptation du recuit simulé au problème $Pm|res^t \cdot 11|C_{max}$

– Solution initiale : La solution initiale que nous avons considéré est représentée par la liste de priorité de la meilleure heuristique.

– Voisinage : Le mécanisme proposé pour générer des voisins d'une solution est le suivant : sans perte de généralité, supposons que le $C_{max} = maximum(C_j)$ $(j = 1..., m)$, est sur la machine M_1 et le $minimum(C_j)$ est sur la machine M_2, nous calculons l'écart $\Delta C = C_1 - C_2$. S'il existe une tâche T_i qui est traitée sur la machine M_1 avec un temps de traitement inférieur à ΔC, on la déplace à la dernière position sur la machine M_2 si elle respecte les contraintes de ressources. Sinon, on permute entre deux tâches quelconques dans la liste de priorité décrivant la solution en cours, on note le voisinage d'une liste l l'ensemble des listes $N(l)$.

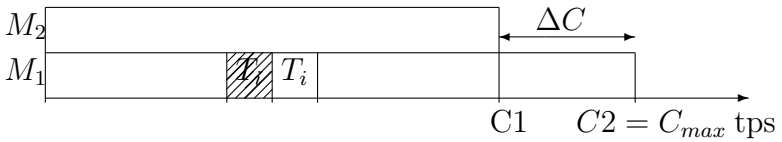


FIG. 4.6 – Voisinage.

– Fonction d'évaluation : L'objectif du notre problème est de minimiser la date de fin de traitement de l'ensemble des tâches qui correspond à la minimisation de l'énergie du système. A toute solution du problème, nous appliquons les sélections Sérielle et Parallèle et nous choisissons la meilleure des deux.

Le schéma général de la méthode est donné par l'algorithme suivant :

Algorithme de recuit simulé pour le problème $Pm|res^t \cdot 11|C_{max}$

1. Initialiser l la liste de départ.
2. Initialiser une température maximale T et une température minimale $Tmin$.
3. **Tantque** $(T > Tmin)$ **faire**
4. - Choisir un voisin $l' \in N(l)$
5. - Calculer ΔC : $\Delta C = C(l') - C(l)$
6. **Si** $\Delta C < 0$ **then** $l' = l$
7. **Sinon**
8. - Générer aléatoirement une probabilité r
9. **Si** $r \leq \exp \frac{-\Delta C}{T}$ **then** $l' = l$
10. **Finsi**
11. **Finsi**
12. $T = T * (1 - alpha)$
13. **Fin**

4.3.2 Algorithme génétique

Les algorithmes génétiques font partie de la famille des algorithmes évolutionnaires. Ils s'inspirent de l'évolution biologique des espèces et basées sur une imitation des phénomènes d'adaptation des êtres vivants. Avec ce type de méthodes, il ne s'agit pas de trouver une solution analytique exacte mais de trouver une bonne solution satisfaisante dans un temps de calcul raisonnable. La première description du processus des algorithmes génétiques a été donnée par Holland en 1975 [30], puis Goldberg [35] les a utilisés pour résoudre des problèmes concrets d'optimisation.

Le but de ces algorithmes génétiques est d'optimiser une fonction prédéfinie, appelée fonction objectif, ou fitness ; ils travaillent sur un ensemble de solutions candidates, appelé *population* d'individus ou chromosomes. Ces derniers sont constitués d'un ensemble d'éléments, appelés *gènes*, qui peuvent prendre plusieurs valeurs, appelées *allèles*. Un chromosome est une représentation ou un codage d'une solution du problème donné. Une première population est choisie soit aléatoirement, soit par des heuristiques ou par des méthodes spécifiques au problème, soit encore par mélange de solutions aléatoires et heuristiques. Cette population doit être suffisamment diversifiée pour que l'algorithme ne reste pas bloqué dans un optimum local. C'est ce qui se produit lorsque trop d'individus sont semblables. Les algorithmes génétiques génèrent de nouveaux individus, de telle sorte qu'ils soient plus performants que leurs prédécesseurs. Le processus d'amélioration des individus s'effectue par utilisation d'opérateurs génétiques, qui sont la sélection, le croisement et la mutation.

Pour mettre en oeuvre un algorithme génétique, il est nécessaire de disposer :

– Une représentation génétique du problème, c'est-à-dire un codage de solutions utilisé sous la forme de chromosomes.

– Un mécanisme de génération de la population initiale.

– Une fonction qui permet d'évaluer l'adaptation d'un chromosome à son environnement, ce qui offre la possibilité de comparer des individus.

– Un mode de sélection des chromosomes à reproduire.

– Opérateurs de croisement et de mutation permettant de diversifier la population au cours des générations.

Les étapes de l'algorithme génétique

L'algorithme génétique commence par une génération d'une population initiale de P_{size} individus, pour lesquels nous calculons leurs fitness et nous sélectionnons les individus par une méthode de sélection. Ces individus seront manipulés par un opérateur de croisement qui les choisit selon une probabilité P_{crois}. Leurs résultats peuvent être mutés par un opérateur de mutation avec une probabilité de mutation P_{mut}. Les phases de sélection et de recombinaison (croisement et mutation) permettent de générer une nouvelle population d'individus, qui ont de bonnes chances d'être plus forts que ceux de la génération précédente. Les individus issus de la phase de recombinaison seront insérés par une méthode d'insertion dans la nouvelle population, dont nous évaluons la valeur de la fonction objectif de chacun de ses individus. De génération en génération, la force des individus de la population augmente et un test d'arrêt sera effectué pour décider quand arrêter l'algorithme. La Figure 4.7 présente un schéma de fonctionnement général de l'algorithme génétique.

Adaptation de l'algorithme génétique au problème $Pm|res^t \cdot 11|C_{max}$

– Codage : Le premier pas dans l'implantation des algorithmes génétiques est de créer une population d'individus initiaux. Le codage que nous avons retenu est de longueur égale au nombre total de tâches à réaliser. L'individu est alors représenté par une séquence de tâches. La Figure 4.8 présente le codage d'une solution quelconque. La tâche numéro 5 sera lancé en production la première, suivi de la tâche numéro 7... etc. et enfin la tâche numéro 6, placé dans le dernier chromosome du code, sera lancée la dernière.

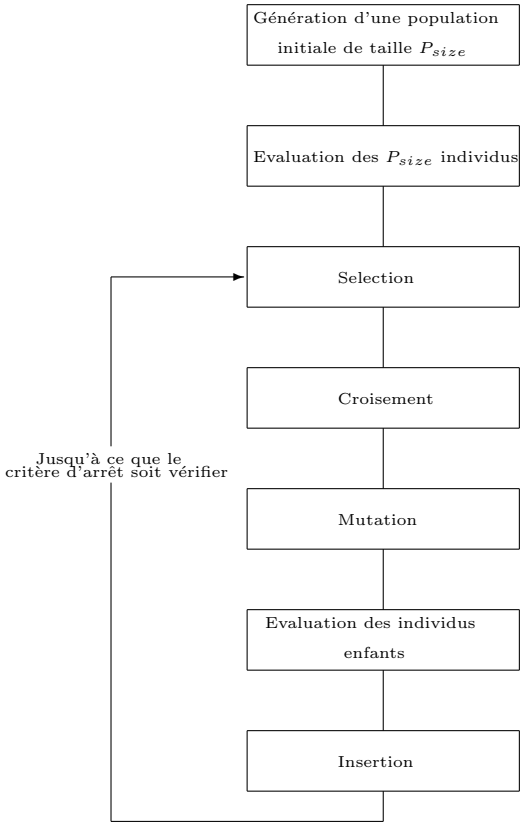


FIG. 4.7 – Fonctionnement général de l'algorithme génétique.

5	7	4	2	1	8	3	6

FIG. 4.8 – Codage d'une solution.

– Population initiale : la population initiale que nous avons considéré est composée de séquences de tâches utilisant les règles de priorités définies dans la section précédente et de séquences de tâches générées aléatoirement. Le problème principal dans cette étape est le choix de la taille de la population. Si la taille de la population est trop grande, le temps de calcul augmente et demande un espace mémoire important. Par contre, une population de taille très petite, la solution obtenue n'est pas satisfaisante. Il faut donc trouver le bon compromis. Pour cela, nous avons testé le paramètre T_{int} qui représente la taille de notre population avec les valeurs $\{10, 20, 50, 100, 200\}$ et d'après les résultats obtenus, la bonne valeur est $T_{int} = 50$.

– La procédure de sélection : la sélection permet d'identifier les individus susceptibles d'être croisés dans une population. La procédure de sélection que nous avons utilisée est la sélection aléatoire, cette sélection se fait aléatoirement, uniformément et sans

intervention de la valeur de la fonction objectif, donc chaque individu a une probabilité uniforme $\frac{1}{T_{int}}$ d'être sélectionné.

- Le croisement : le croisement permet d'enrichir la population en manipulant les composantes des chromosomes. Un croisement est envisagé avec deux parents et génère un ou deux enfants. Les individus survivants à la phase de sélection vont subir le croisement avec une probabilité P_{crois}, dans notre algorithme génétique, nous utilisons le croisement en 2-points. Ce dernier s'effectuera de la manière suivante :

  - Choisir deux individus de la population actuelle comme parents pour générer deux enfants.

  - Générer aléatoirement deux positions $k_1, k_2 \in [1, ..., n]$ avec $k_1 < k_2$.

  - Générer aléatoirement une probabilité de croisement α.

  - Si $\alpha \leq P_{crois}$: ce croisement conserve dans l'enfant i la zone interne du parent j (zone comprise entre k_1 et k_2)et pour l'enfant j la zone interne du parent i. Ensuite, compléter les cases vides restantes de l'enfant i par les éléments du parent i et les cases vides restantes de l'enfant j par les éléments du parent j qui ne provoquent pas de doublon.

  - Si $\alpha > P_{crois}$: copier le parent i à l'enfant i et le parent j à l'enfant j.

- La mutation : les individus issus du croisement vont ensuite subir un processus de mutation avec une probabilité P_{mut}. La mutation nous garantit que l'algorithme génétique sera susceptible d'atteindre la plupart des points du domaine réalisable. L'opérateur de mutation utilisé dans notre cas est l'opérateur d'échange réciproque qui permet de sélectionner au hasard deux gènes et de les échanger. Si β, généré aléatoirement, appartient à l'intervalle [0, P_{mut}] nous appliquons l'opérateur de mutation sur l'enfant i pour avoir un enfant muté $imut$, et si $\beta > P_{mut}$ nous appliquons l'opérateur de mutation sur l'enfant j pour avoir un enfant muté $jmut$.

- Fonction d'évaluation : pour chaque chromosome, qui est une permutation de n tâches, nous appliquons les sélections Sérielle et Parallèle définies précédemment, ensuite nous choisissons la meilleure des deux solutions trouvées.

- Insertion : les individus qui sont générés aléatoirement et les enfants mutés sont triés selon leur fonction d'évaluation dans un ordre croissant. Seule la moitié supérieure de la population, correspondant aux meilleurs individus, est sélectionnée. Il est à noter que

la taille de la population reste fixe égale à T_{int} de génération en génération.

– Critère d'arrêt : l'algorithme génétique s'arrête après un nombre fixé de génération noté $Itermax$.

Finalement, les étapes de notre algorithme sont données par l'algorithme suivant :

Algorithm génétique pour le problème $Pm|res^t \cdot 11|C_{max}$

1. Initialiser : T_{int}, $Itermax$, P_{mut}, P_{croi}, δ.
2. Générer la population initiale de taille T_{int}.
3. **Répéter**
4. i=0
5. **Tant que** $\delta < \frac{T_{int}}{2}$ **faire**
6. Sélectionner aléatoirement deux parents de la population.
7. Croisement des deux parents pour obtenir deux enfants par une probabilité P_{croi}.
8. Muter les deux enfants par une probabilité P_{mut} pour obtenir un enfant muté.
9. $\delta = \delta + 1$.
10. **Fin Tant que.**
11. Evaluer tous les chromosomes par la fonction d'évaluation.
12. Ranger les parents et les enfants dans l'ordre croissant selon leur fonction d'évaluation.
13. Supprimer les T_{int} chromosomes faibles et enregistrer les T_{int} meilleures chromosomes selon leur fonction d'évaluation.
14. i=i+1.
15. **Jusqu'à** $i = Itermax$.

L'un des problèmes rencontrés lors de l'application des algorithmes génétiques est la perte de diversité très rapide de la population. De manière à pallier ce problème, nous avons conçu un algorithme génétique hybride.

4.3.3 Algorithme génétique hybride

Les algorithmes génétiques possèdent une connaissance globale à travers leurs populations et explorent l'espace de recherche tandis que le recuit simulé détient une connaissance locale et exploite le voisinage. Dans cette partie, nous proposons un algorithme génétique hybride séquentielle.

Les étapes de l'algorithme génétique hybride

Notre approche de résolution se fait en trois étapes.

1. La première est une application directe de l'algorithme génétique et à la fin de cette dernière, une population d'individus qui satisfait notre critère est obtenue.

2. Dans une deuxième étape, on a recours au recuit simulé. Dans ce cas, le meilleur individu de la population finale obtenu dans la première étape est choisi comme solution initiale.

3. A la fin de la deuxième étape, on obtient une solution qui va être améliorée dans la troisième étape par l'application de l'un des algorithmes de liste basés sur les rangements des tâches définis ci-dessous.

   - CJ : A l'étape i, si la tâche $T_{j'}$ qui est à la position $i+1$ est compatible avec la tâche T_j qui est à la position i on la laisse à cette position, sinon on cherche une tâche compatible avec cette dernière et on la permute avec la tâche $T_{j'}$.

   - CJI : On prend l'inverse du rangement CJ.

   - ICJ : A l'étape i, si la tâche $T_{j'}$ qui est à la position $i+1$ est incompatible avec la tâche T_j qui est à la position i on la laisse à cette position, sinon on cherche une tâche incompatible avec cette dernière et on la permute avec la tâche $J_{j'}$.

   - $ICJI$: On prend l'inverse du rangement ICJ.

   - ER : On sélectionne deux tâches au hasard et on les permute.

Pour le choix des meilleures listes, nous avons mené une étude comparative entre eux en leurs appliquant les sélections Sérielle et Parallèle. Nous avons généré des instances aléatoires du problème pour les valeurs de m et n appartenant aux ensembles $\{2, 5, 10\}$, $\{50, 100, 500, 1000\}$, respectivement. Pour chaque valeur de n, nous avons testé 100 instances. Les temps de préparation et d'exécution des tâches sont tirés suivant la loi uniforme dans les intervalles [1, 10], [10, 50] et [50, 100]. Nous avons comparé ces heuristiques les unes par rapport aux autres en calculant le nombre de fois où une heuristique fournit de meilleures solutions.

Les résultats des expérimentations numériques sont donnés dans la Table 4.2.

n	$\|R\|$	$s_i, p_i \in [1,10]$					$s_i, p_i \in [10,50]$					$s_i, p_i \in [50,100]$				
		CJ	**CJI**	**ICJ**	**ICJI**	**ER**	**CJ**	**CJI**	**ICJ**	**ICJI**	**ER**	**CJ**	**CJI**	**ICJ**	**ICJI**	**ER**
m=2																
50	1	12	59	34	0	20	13	46	30	0	17	7	53	37	0	4
	2	3	50	52	0	19	5	48	43	0	10	4	39	45	0	12
	5	2	58	24	11	18	4	58	17	7	14	15	44	20	10	12
	7	12	61	13	10	16	10	56	9	17	12	18	33	17	18	16
100	1	14	50	46	0	22	9	46	39	0	15	8	46	42	0	8
	2	1	49	51	0	9	2	54	40	0	9	2	56	40	0	5
	5	0	89	10	0	5	0	81	15	0	5	7	60	20	2	12
	7	0	88	7	5	5	3	72	15	7	5	22	40	14	12	13
500	1	2	50	52	0	24	4	41	42	0	18	2	47	46	0	6
	2	1	47	46	0	22	0	50	47	0	9	0	55	43	0	2
	5	0	99	4	0	0	0	97	3	0	0	0	94	6	0	0
	7	0	100	0	0	0	0	99	1	0	0	0	96	3	0	1
1000	1	7	46	47	0	28	3	47	40	0	15	1	38	53	0	8
	2	0	56	46	0	10	0	50	45	0	7	0	50	49	0	2
	5	0	95	9	0	0	0	99	1	0	0	0	99	1	0	0
	7	0	100	0	0	0	0	100	0	0	0	0	100	0	0	0
m=5																
50	1	21	44	34	0	17	28	31	20	0	21	23	32	29	0	16
	2	19	43	28	2	26	23	39	21	1	24	14	52	21	2	14
	5	4	68	17	13	12	7	62	16	7	15	4	71	15	9	8
	7	3	57	25	15	22	9	57	17	16	11	4	64	16	10	13
100	1	36	25	21	0	22	31	29	21	0	22	18	28	27	0	28
	2	16	52	20	0	22	18	45	21	0	19	7	65	14	0	15
	5	0	80	11	4	9	2	80	9	3	6	0	78	10	8	5
	7	2	85	8	4	5	1	82	4	5	10	1	86	7	4	5
500	1	20	31	22	0	30	17	37	36	0	11	25	26	25	0	25
	2	14	58	20	0	17	8	67	12	0	13	11	66	14	0	9
	5	0	97	2	0	1	0	99	1	0	0	0	98	0	0	2
	7	0	100	0	0	0	0	99	0	0	1	0	100	0	0	0
1000	1	19	41	24	0	20	21	31	26	0	22	22	25	27	0	26
	2	7	61	18	0	18	3	66	14	0	17	7	65	14	0	14
	5	0	98	2	0	0	0	98	1	0	1	0	99	1	0	0
	7	0	99	1	0	0	0	100	0	0	0	0	100	0	0	0
m=10																
50	1	38	45	29	0	30	27	36	23	0	21	29	40	24	0	23
	2	27	48	35	7	35	27	39	34	5	25	30	48	29	6	28
	5	7	51	25	21	19	10	42	25	20	20	6	63	20	6	13
	7	13	56	23	19	23	5	63	16	11	18	8	55	18	15	13
100	1	27	41	25	0	23	27	33	16	0	28	27	31	28	0	17
	2	23	45	30	0	22	21	38	21	0	25	20	39	21	2	30
	5	3	65	18	11	13	5	58	18	11	13	4	67	9	11	12
	7	1	78	11	10	10	1	78	9	9	9	0	78	10	6	8
500	1	33	30	25	0	28	28	27	24	0	23	18	22	27	0	33
	2	16	54	21	0	23	19	42	22	0	17	16	43	22	0	19
	5	0	90	8	1	1	0	76	11	4	10	3	66	18	4	9
	7	0	99	0	0	1	0	97	1	0	2	0	95	1	0	4
1000	1	29	27	20	0	31	27	24	24	0	27	21	32	25	0	22
	2	20	50	18	0	19	17	46	26	0	17	14	47	19	0	20
	5	0	96	3	0	1	0	71	14	0	15	3	62	15	5	15
	7	0	100	0	0	0	0	96	1	0	3	0	96	3	0	1

TAB. 4.2 – Comparaison entre les algorithmes de liste.

A partir de la Table 4.2, nous constatons que l'heuristique basée sur la liste CJI donne de meilleurs résultats pour toutes les instances testées. Nous remarquons qu'elle est meilleure à 100% pour la plupart des instances à 500 et 1000 tâches avec 5 et 7 types de ressources. L'heuristique basée sur la liste ICJ fournit des résultats satisfaisants pour les instances à 50 et 100 tâches avec 1 et 2 types de ressources.

4.4 Expérimentations numériques

Dans la section précédente, nous avons discuté des approches métaheuristiques que nous avons adapté pour la résolution du problème $Pm|res^t \cdot 11|C_{max}$. Les métaheuristiques disposent de mécanismes particuliers permettant d'éviter d'être rapidement piégées par les minimas locaux. Ces méthodes se montrent donc le plus souvent plus puissantes que les heuristiques. Cependant, le bon réglage des différents paramètres est l'étape cruciale pour que la métaheuristique retourne de bons résultats. Pour le recuit simulé, nous avons fixé ses paramètres de la manière suivante : Température initiale $T = 100$ (nous avons testé le paramètre T avec plusieurs valeurs 10, 50, 100 et 500, et d'après les résultats obtenus, la bonne valeur est $T = 100$ car nous n'avons remarqué aucune amélioration de la solution pour les autres valeurs). Température minimale $T_{min} = 10^{-4}$, donc l'algorithme s'arrête quand $T < 10^{-4}$ et pour diminuer la température nous avons choisi $\alpha = 0,3$. Le nombre de paliers effectués vaut donc 40. Pour les paramètres de l'algorithme génétique et après plusieurs expérimentations nous avons pu les fixer comme suit : Une taille de la population égale à 50, une probabilité de croisement $P_{crois} = 0.8$, une probabilité de mutation $P_{mut} = 0.1$ et en fin l'algorithme s'arrête après 100 itérations.

L'objectif de cette section est de comparer les performances de l'algorithme génétique (AG), l'algorithme génétique hybride (AGH) et le recuit simulé (RS). Nous avons généré des problèmes tests possédant différentes caractéristiques, dans un premier temps nous avons fixé le nombre de machines à $m = 2$ et 5, et pour chaque nombre de machines nous avons fait varier le nombre de tâches ($n = 50, 100, 500$) et pour chaque nombre de tâches nous avons fait varier le nombre de types de ressources ($k = 1, 2, 5, 7$). Les temps de préparation et d'exécution sont générés selon une loi uniforme dans les intervalles suivants : [1, 10],[10, 50] et [50, 100]. Les résultats de ces méthodes ont été comparés avec la borne inférieure LB. Nous avons calculé le nombre de fois où une métaheuristique fournit de meilleures solutions par rapport aux autres et le nombre de fois où la solution trouvée par une métaheuristique est égale à la borne inférieure. Aussi, nous avons calculé les déviations moyenne et maximale. Les Tables 4.3 et 4.4 illustrent ces résultats.

n	$\|R\|$		$s_i, p_i \in [1,10]$			$s_i, p_i \in [10,50]$			$s_i, p_i \in [50,100]$		
			AG	**AGH**	**RS**	**AG**	**AGH**	**RS**	**AG**	**AGH**	**RS**
50	1	#Best	100	86	79	100	79	14	98	92	5
		#LB	100	86	79	100	79	14	95	90	5
		$\%Dev_{max}$	0	1,06	6,36	0	0,59	5,31	0,4	0,32	4,0
		$\%Dev_{moy}$	0	0,06	0,29	0	0,03	0,77	0,008	0,01	1,28
	2	#Best	100	74	54	96	88	4	94	83	1
		#LB	100	74	54	96	88	4	88	78	1
		$\%Dev_{max}$	0	4,16	14,23	0,13	0,32	7,58	0,71	0,6	4,39
		$\%Dev_{moy}$	0	0,21	0,93	0,003	0,01	1,56	0,02	0,03	1,82
	5	#Best	97	53	2	81	71	0	61	72	0
		#LB	95	52	2	50	51	0	36	37	0
		$\%Dev_{max}$	0,37	5,0	15,38	1,67	2,16	9,58	0,9	1,05	8,32
		$\%Dev_{moy}$	0,01	0,74	4,33	0,14	0,21	4,41	0,17	0,14	3,47
	7	#Best	90	62	0	54	59	0	62	44	0
		#LB	52	38	0	13	9	0	1	2	0
		$\%Dev_{max}$	2,43	10,76	18,05	2,15	1,91	12,05	1,63	1,45	8,73
		$\%Dev_{moy}$	0,28	1,07	6,44	0,52	0,49	6,57	0,5	0,56	5,21
100	1	#Best	100	84	84	100	74	16	90	97	1
		#LB	100	84	84	100	74	16	90	97	1
		$\%Dev_{max}$	0	0,35	0,36	0	0,13	4,46	0,17	0,09	1,98
		$\%Dev_{moy}$	0	0,03	0,14	0	0,01	0,48	0,003	0,003	0,7
	2	#Best	100	67	49	99	84	5	87	92	0
		#LB	100	67	49	99	84	5	81	86	0
		$\%Dev_{max}$	0	0,53	6,66	0,03	0,41	4,61	0,22	0,15	2,67
		$\%Dev_{moy}$	0	0,07	0,64	$3 * 10^{-6}$	0,02	0,97	0,01	0,01	1,05
	5	#Best	88	64	0	79	81	0	55	70	0
		#LB	83	58	0	64	63	0	23	29	0
		$\%Dev_{max}$	0,37	3,11	8,08	1,41	0,72	5,64	0,76	0,66	5,29
		$\%Dev_{moy}$	0,04	0,44	3,12	0,09	0,08	2,67	0,14	0,13	2,55
	7	#Best	80	60	0	55	59	0	51	53	0
		#LB	41	29	0	11	8	0	0	0	0
		$\%Dev_{max}$	1,88	5,28	9,57	2,43	2,9	9,39	1,47	1,34	6,43
		$\%Dev_{moy}$	0,35	0,62	5,11	0,51	0,51	4,67	0,45	0,44	3,93
500	1	#Best	100	90	87	100	85	12	90	100	3
		#LB	100	90	87	100	85	12	90	100	3
		$\%Dev_{max}$	0	0,03	0,58	0	0,04	1,51	0,03	0	0,75
		$\%Dev_{moy}$	0	0,003	0,01	0	0,002	0,16	0,001	0	0,16
	2	#Best	100	75	62	100	86	1	100	96	0
		#LB	100	75	62	100	86	1	99	96	0
		$\%Dev_{max}$	0	0,07	1,92	0	0,09	1,73	0,005	0,04	0,86
		$\%Dev_{moy}$	0	0,009	0,15	0	0,003	0,25	$5*10^{-7}$	$6*10^{-6}$	0,25
	5	#Best	100	80	0	97	98	0	64	79	0
		#LB	100	80	0	97	98	0	44	51	0
		$\%Dev_{max}$	0	0,33	4,03	0,1	0,01	2,63	0,19	0,14	1,55
		$\%Dev_{moy}$	0	0,03	1,22	0,001	$2*10^{-6}$	0,68	0,031	0,026	0,94
	7	#Best	84	89	0	65	70	0	53	55	0
		#LB	71	71	0	39	39	0	7	8	0
		$\%Dev_{max}$	0,25	0,25	5,08	0,58	0,8	2,98	0,48	0,41	2,79
		$\%Dev_{moy}$	0,02	0,02	1,46	0,08	0,08	1,73	0,16	0,16	1,78

TAB. 4.3 – Etude comparative entre les métaheuristiques pour $m = 2$.

Sur l'ensemble des tests effectués, l'algorithme génétique se montre efficace dans la résolution de la majorité des problèmes. Il est capable de résoudre à l'optimalité tous les problèmes à 50, 100 et 500 tâches avec 1, 2 et 5 types de ressources quand les temps de préparation et d'exécution prennent leurs valeurs dans l'intervalle [1, 10]. Nous remarquons aussi que pour les instances à $n = 100$ et 500 tâches, l'algorithme génétique et l'algorithme génétique hybride sont proches en terme de performance. Toutefois, l'algorithme génétique hybride devient meilleur pour les instances à 5 et 7 types de ressources quand les temps de préparation et d'exécution prennent leurs valeurs dans les intervalles [10, 50] et [50, 100]. Par ailleurs, le recuit simulé est le moins performant des trois.

Pour une meilleure lisibilité, les Figures 4.11, 4.12 transcrivent les résultats trouvés par l'algorithme génétique et l'algorithme génétique hybride pour $s_i, p_i \in [10, 50]$.

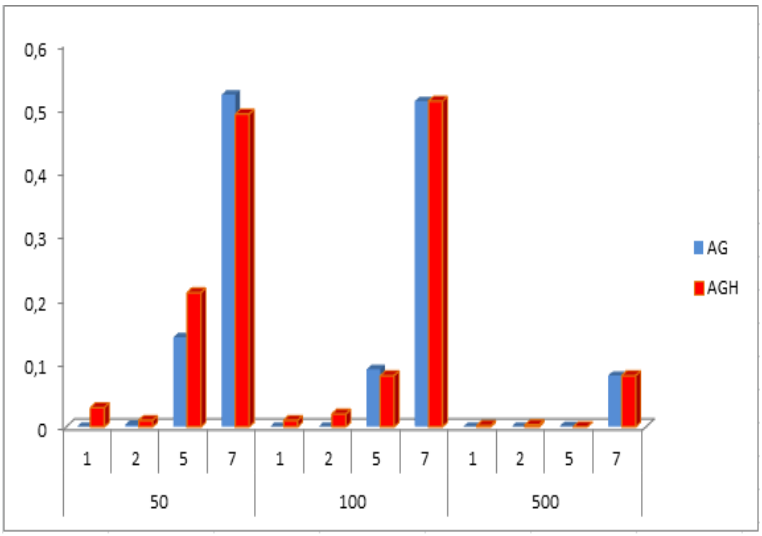


FIG. 4.9 – Comportement de l'AG et l'AGH par rapport à la borne inférieure, $m = 2$.

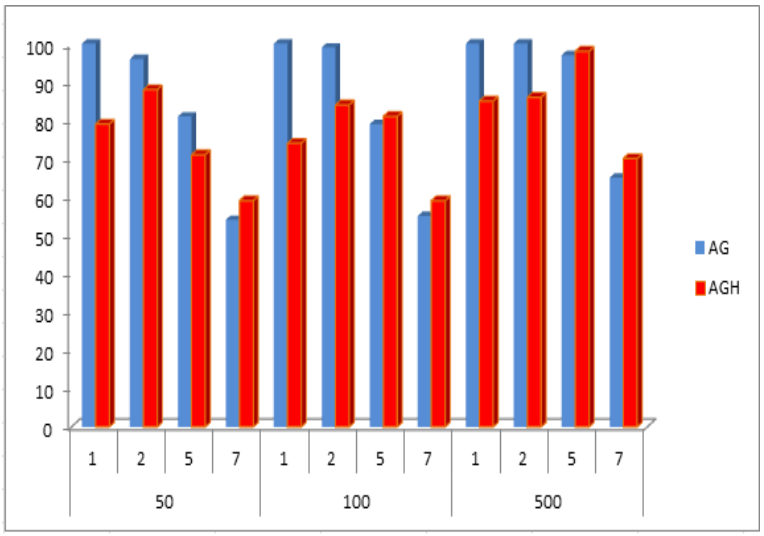


FIG. 4.10 – Comportement de l'AG et l'AGH par rapport à la meilleure solution, $m = 2$.

n	$\|R\|$		$s_i, p_i \in [1,10]$			$s_i, p_i \in [10,50]$			$s_i, p_i \in [50,100]$		
			AG	**AGH**	**RS**	**AG**	**AGH**	**RS**	**AG**	**AGH**	**RS**
50	1	#Best	97	10	0	97	4	0	100	0	0
		#LB	75	4	0	51	0	0	62	0	0
		$\%Dev_{max}$	3,63	15,78	30,55	4,34	18,91	28,54	4,98	20,41	33,51
		$\%Dev_{moy}$	0,39	6,96	9,53	0,32	9,62	10,71	0,35	10,52	14,67
	2	#Best	90	54	0	81	36	0	91	12	0
		#LB	69	37	0	23	3	0	26	0	0
		$\%Dev_{max}$	10,31	21,42	30,83	3,37	12,85	22,06	7	13,3	29,66
		$\%Dev_{moy}$	0,76	2,43	10,6	0,69	2,8	9,23	0,81	4,34	9,68
	5	#Best	96	62	0	93	97	14	87	95	0
		#LB	45	29	0	42	43	0	23	23	0
		$\%Dev_{max}$	16,1	19,08	34,35	16,71	16,56	28,63	12,03	12,03	31,74
		$\%Dev_{moy}$	2,3	3,85	12,86	1,32	1,33	10,94	2,43	2,41	12,99
	7	#Best	98	57	0	79	91	0	88	95	0
		#LB	49	28	0	29	27	0	23	24	0
		$\%Dev_{max}$	8,72	12,79	29,86	10,71	10,96	25,45	8,8	8,7	30,03
		$\%Dev_{moy}$	1,22	2,52	9,29	1,17	1,16	9,36	1,39	1,4	9,78
100	1	#Best	98	3	0	100	1	0	100	0	0
		#LB	74	0	0	48	0	0	41	0	0
		$\%Dev_{max}$	5,02	15,0	18,72	4,41	20,45	26,79	2,62	18,17	28,66
		$\%Dev_{moy}$	0,4	8,55	9,26	0,04	11,48	11,62	0,12	12,0	14,63
	2	#Best	78	55	0	68	34	0	77	25	0
		#LB	25	10	0	8	0	0	8	0	0
		$\%Dev_{max}$	9,15	8,05	23,25	9,54	10,69	21,76	7,67	10,79	22,88
		$\%Dev_{moy}$	0,8	1,58	9,3	1,83	3,18	9,32	1,86	4,49	10,8
	5	#Best	85	86	0	70	84	0	68	85	0
		#LB	29	31	0	13	16	0	17	17	0
		$\%Dev_{max}$	11,9	11,5	36,8	14,29	14,22	24,08	11,32	11,22	25,49
		$\%Dev_{moy}$	2,75	2,33	12,9	2,35	2,32	13,2	2,02	1,9	13,19
	7	#Best	68	94	0	76	89	0	85	98	0
		#LB	13	14	0	10	11	0	6	7	0
		$\%Dev_{max}$	18,41	10,63	30,32	10,54	10,17	23,7	8,54	8,51	24,65
		$\%Dev_{moy}$	2,85	2,46	12,48	1,77	1,77	12,35	1,87	1,86	12,45
500	1	#Best	100	0	0	100	0	0	100	0	0
		#LB	18	0	0	6	0	0	0	0	0
		$\%Dev_{max}$	3,15	14,18	13,76	5,6	18,55	17,27	1,57	17,09	21,0
		$\%Dev_{moy}$	0,59	10,7	10,2	0,36	12,9	12,21	0,27	14,23	16,13
	2	#Best	43	60	0	50	50	0	70	30	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	4,52	5,19	10,39	8,39	8,33	14,59	6,82	7,47	13,56
		$\%Dev_{moy}$	3,15	3,08	7,8	4,74	4,73	9,01	4,32	5,21	11,02
	5	#Best	45	74	0	50	75	0	50	53	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	8,02	7,33	21,42	8,23	8,68	20,9	4,67	4,65	19,6
		$\%Dev_{moy}$	3,34	3,21	13,29	2,42	2,33	14,36	2,006	1,9	14,15
	7	#Best	45	60	0	34	72	0	40	74	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	8,23	8,1	21,88	8,47	7,7	22,31	7,03	6,3	23,2
		$\%Dev_{moy}$	3,74	3,61	15,63	3,13	3,0	11,53	2,73	2,52	15,9

TAB. 4.4 – Etude comparative entre les métaheuristiques, $m = 5$.

Plus le nombre de machines augmente, plus l'écart moyen entre les méthodes et la borne inférieure augmente. Pour $m = 5$ machines, l'algorithme génétique apparaît relativement plus performant dans le cas où on a 1 et 2 types de ressources. La déviation moyenne est inférieure à $0,6\%$ pour $k = 1$ et inférieure à 4% pour $k = 2$. En effet, pour $k = 5, 7$, l'algorithme génétique hybride est en mesure de fournir des solutions meilleures que l'algorithme génétique. Par exemple, pour $n = 500$ et $k = 5$ il donne une solution réalisable dans 74 cas et une déviation moyenne inférieure à $3,3\%$ pour le premier intervalle et il trouve une solution réalisable dans 60 cas pour $k = 7$ et une déviation moyenne ne dépassant pas $3,7\%$.

Les Figures suivantes décrivent les résultats concernant la performance de l'AG et l'AGH par rapport à la borne inférieure et par rapport à la meilleure solution, $s_i, p_i \in [10, 50]$.

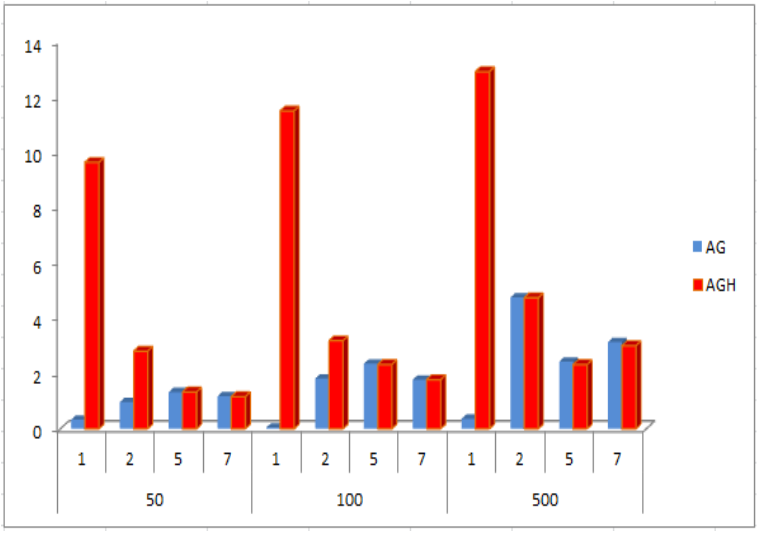


FIG. 4.11 – Comportement de l'AG et l'AGH par rapport à la borne inférieure, $m = 5$.

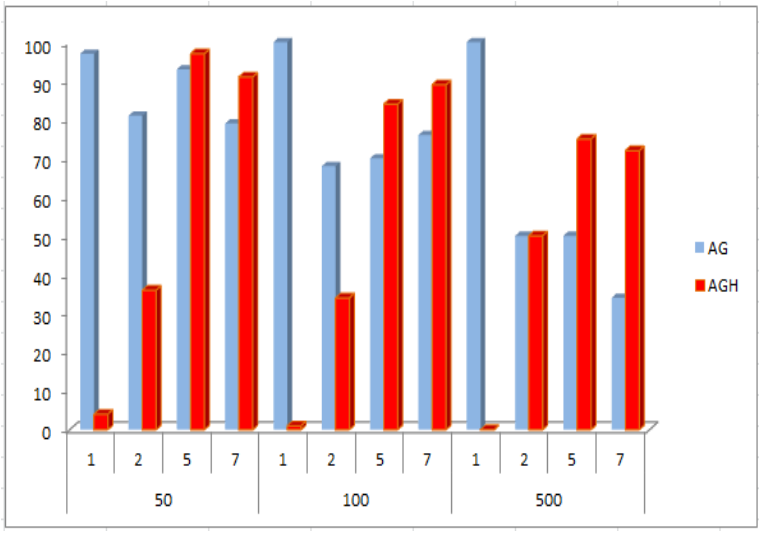


FIG. 4.12 – Comportement de l'AG et l'AGH par rapport à la meilleure solution, $m = 5$.

On remarque que plus le nombre de tâches et le nombre de types de ressources augmentent, le nombre de solutions trouvées par l'AG décroît contrairement à l'AGH.

Chapitre 5

Etude du problème à 2 machines et un seul type de ressources

Dans ce chapitre, nous nous intéressons à l'étude d'un sous-problème du problème général sur le critère de la durée totale d'ordonnancement. Nous proposons une méthode exacte à base de modélisation mathématique pour résoudre le problème. Nous étudions des cas particuliers de ce sous-problème et nous proposons des bornes inférieures. De plus, nous développons un ensemble d'heuristiques pour sa résolution avec des expérimentations numériques.

5.1 Définition et notations

Le problème d'ordonnancement avec la présence d'un seul type de ressources est défini comme suit : Soit un ensemble de tâches $T_1, T_2, ..., T_n$ qui doit être réalisé sur deux machines identiques M_1, M_2. Nous supposons qu'un seul type de ressources R est disponible en q unités. Chaque tâche nécessite pour son traitement un temps de préparation s_i, un temps d'exécution p_i et b_i unités de la ressource R ($b_i \leq q$) pour sa préparation. L'objectif est de minimiser le makespan. Le problème considéré est noté par $P2|res^t1\cdot\cdot|C_{max}$.

Pour illustrer ce problème, nous proposons l'exemple suivant :

Exemple 7. *Soit à exécuter un ensemble de huit tâches T_1, T_2,..., T_8 sur deux machines M_1 et M_2 avec $q = 4$. Les temps de préparation, les temps d'exécution des tâches et le nombre d'unité de la ressource R nécessaire pour la préparation de chaque tâche sont donnés par la Table 5.1.*

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
s_i	2	4	5	3	3	1	4	1
p_i	1	5	3	4	1	4	1	1
b_i	1	2	2	3	4	2	3	1

TAB. 5.1 – Paramètres liés aux tâches.

Un ordonnancement réalisable est représenté par la Figure 5.1.

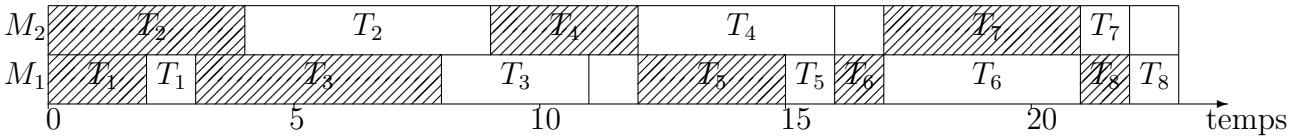


FIG. 5.1 – Solution réalisable de l'exemple 7.

5.2 Modélisation mathématique

Dans cette section, nous présentons une modélisation du problème $P2|res^t1\cdot\cdot|C_{max}$ en un programme linéaire en nombres entiers.

5.2.1 Les variables

Considérons les variables bivalentes suivantes :

- $X_{ij}=\begin{cases} 1 & \text{si la tâche } T_i \text{ est traitée sur la machine } M_j \qquad i=\overline{1,n},\ j=\overline{1,2}\,; \\ 0 & \text{sinon.} \end{cases}$
- $\alpha_{il}=\begin{cases} 1 & \text{si } t_i \leq t_l \qquad \text{pour } i,l=\overline{1,n},\ i\neq l\,; \\ 0 & \text{sinon.} \end{cases}$
- $\beta_{il}=\begin{cases} 1 & \text{si } t_l - t_i - s_i + 1 \leq 0 \qquad \text{pour } i,l=\overline{1,n},\ i\neq l\,; \\ 0 & \text{sinon.} \end{cases}$
- t_i : date de début de préparation de la tâche T_i $(i=\overline{1,n})$;

et soit $M=\sum_{i=1}^{n}(p_i+s_i)$: une borne supérieure.

5.2.2 Les contraintes

(1)- Cette contrainte assure que l'on assigne une tâche à une machine et une seule.

$$\sum_{j=1}^{2} X_{ij} = 1 \qquad \text{pour } i = \overline{1,n}.$$

(2)- Le chevauchement de deux tâches sur la même machine est une configuration interdite.

Si $X_{ij} = 1$ et $X_{lj} = 1$ alors les deux tâches T_i et T_l sont affectées à la même machine M_j, et T_i commence avant T_l, donc $t_i + s_i + p_i \leq t_l$.

Les variables α_{il} sont décrites par les deux types de contraintes suivantes :

$$\alpha_{il} + \alpha_{li} = 1 \text{ pour } i, l = \overline{1,n},\ i < l$$

$$t_l - t_i + 1 \leq M\alpha_{il} \qquad \text{pour } i, l = \overline{1,n},\ i \neq l\,;$$

Si $\alpha_{il} = 1$ alors $\alpha_{li} = 0$ donc $t_i - t_l + 1 \leq M\alpha_{li} = 0$ et $t_i \leq t_l$. Et si $\alpha_{il} = 0$ alors $t_l - t_i + 1 \leq M\alpha_{il} = 0$ donc $t_l \leq t_i$.

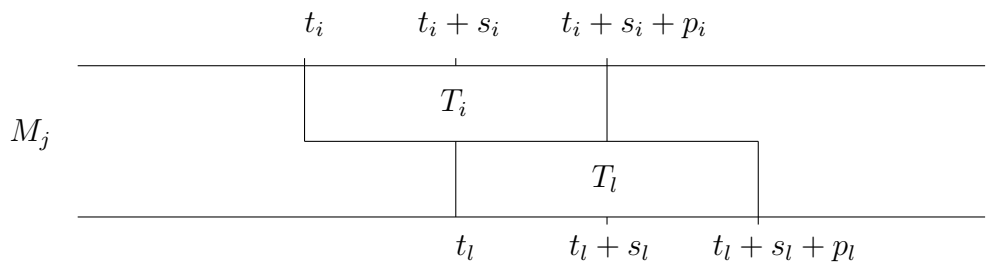


FIG. 5.2 – Chevauchement de deux tâches sur la même machine est interdit.

Les contraintes de non chevauchement sont donc :

$$t_i + p_i + s_i - t_l \leq M(1 - \alpha_{il} + 2 - X_{ij} - X_{lj}) \qquad \text{pour } i, l = \overline{1,n},\ i \neq l,\ j = \overline{1,2}\,;$$

Si les deux tâches T_i et T_l sont affectées à la même machine M_j, et T_i commence avant T_l, on aura $X_{ij} = 1$ et $X_{lj} = 1$ et forcement $\alpha_{il} = 1$ à cause des contraintes $t_l - t_i + 1 \leq M\alpha_{il}$, alors $t_i + s_i + p_i - t_l \leq 0 \Longrightarrow t_i + s_i + p_i \leq t_l$. C'est à dire que la tâche T_i termine avant le début de la tâche T_l.

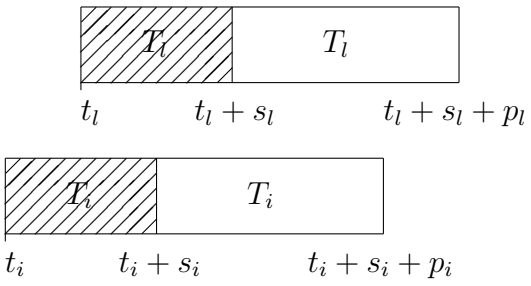


FIG. 5.3 – Le cas : $t_i \leq t_l < t_i + s_i$.

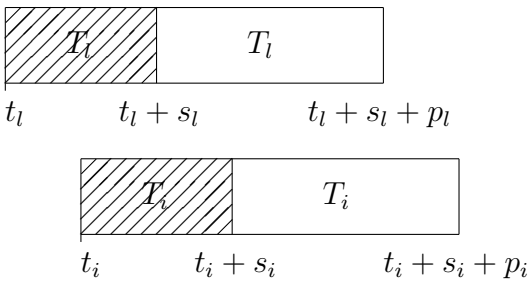


FIG. 5.4 – Le cas : $t_l \leq t_i \leq t_l + s_l$.

(3)- Contraintes de capacité de préparation : une paire de tâches (T_i, T_l) est préparée sur les deux machines en parallèle si et seulement si $b_i + b_l \leq q$. On distingue deux cas présentés par les Figures 5.3, 5.4 :

On a : $(t_i \leq t_l < t_i + s_i)$ ou $(t_l \leq t_i \leq t_l + s_l) \Longrightarrow b_i + b_l \leq q$

Nous avons aussi $(t_i \leq t_l < t_i + s_i) \Longrightarrow (b_i + b_l \leq q)$ si et seulement si :

$$(t_i - t_l \leq 0) \text{ et } (t_l - t_i - s_i < 0) \Longrightarrow b_i + b_l \leq q$$
$$(t_i - t_l - 1 < 0) \text{ et } (t_l - t_i - s_i < 0) \Longrightarrow b_i + b_l \leq q$$

On aura donc :

$$t_l - t_i + 1 \leq M\alpha_{il}$$
$$\text{et}$$
$$t_i + s_i - t_l \leq M\beta_{il}$$
$$\text{et}$$
$$b_i + b_l - q \leq M(2 - \alpha_{il} - \beta_{il})$$

pour $i, l = \overline{1, n}$, $i \neq l$

(4)- La fonction objectif est alors minimiser le maximum de $\{t_i + s_i + p_i\}$:

$$t_i + s_i + p_i \leq Y \qquad i = \overline{1,n}.$$

Le programme linéaire en nombres entiers s'écrit de la manière suivante :

Min Max $\{t_i + s_i + p_i\}$

$$\begin{cases} \sum_{j=1}^{2} X_{ij} = 1 & i = \overline{1,n} & (1) \\ \alpha_{il} + \alpha_{li} = 1 & i,l = \overline{1,n};\ i < l & (2) \\ t_l - t_i + 1 \leq M\alpha_{il} & i,l = \overline{1,n};\ i \neq l & (3) \\ t_i + p_i + s_i - t_l \leq M(1 - \alpha_{il} + 2 - X_{ij} - X_{lj}) & i,l = \overline{1,n};\ i \neq l;\ j = \overline{1,2} & (4) \\ t_i + s_i - t_l \leq M\beta_{il} & i,l = \overline{1,n};\ i \neq l & (5) \\ b_i + b_l - q \leq M(2 - \alpha_{il} - \beta_{il}) & i,l = \overline{1,n};\ i \neq l & (6) \\ X_{ij} \in \{0,1\} & j = \overline{1,2};\ i = \overline{1,n} & (7) \\ t_i \geq 0 & i = \overline{1,n} & (8) \\ \alpha_{il}, \beta_{il} \in \{0,1\} & i,l = \overline{1,n};\ i \neq l & (9) \\ Y \geq 0 & & (10) \end{cases}$$

5.2.3 Taille du modèle

Le modèle mathématique précédent a :

- $(2n^2 + n + 1)$ variables.
- n contraintes de type (1).
- $\frac{n(n-1)}{2}$ contraintes de type (2).
- $n(n-1)$ contraintes de type (3).
- $2n(n-1)$ contraintes de type (4).
- $n(n-1)$ contraintes de type (5).
- $n(n-1)$ contraintes de type (6).

Donc ce dernier requiert $(2n^2 + n + 1)$ variables et $\frac{10n^2-9n}{2}$ contraintes.

Nous avons testé la qualité et l'efficacité du modèle mathématique, présenté précédemment, en utilisant le solveur CPLEX uniquement sur des instances de petite taille, car à partir de 10 tâches le temps de calcul devient exponentiel. Pour chaque problème nous avons fixé le nombre de machines à $m = 2$, le nombre de tâches à $n = 2, 3, 4, ..., 10$ et les demandes en ressource R par les tâches ont été générées suivant la loi uniforme dans l'intervalle [1, 5]. La table suivante récapitule les résultats obtenus par le solveur CPLEX.

Nombre de tâches	Nombre de variables	Nombre de contraintes	Problèmes résolus	Temps moyen d'exécution
2	15	15	10	0.00
3	28	39	10	< 0.004
4	45	74	10	$0,01$
5	66	120	10	$0,029$
6	91	177	10	0.097
7	120	245	10	0.543
8	153	324	10	4.912
9	190	414	10	36.061
10	231	515	10	274.616

TAB. 5.2 – Résultats obtenus par le solveur CPLEX.

Les résultats expérimentaux montrent que le solveur CPLEX permet de résoudre les problèmes à $n = 2, 3$ tâches en un temps ne dépassant pas 0.004 secondes, pour $n = 4, 5, 6, 7$ un temps ne dépassant pas une seconde, et pour $n = 8$ le temps moyen d'exécution ne dépasse pas 5 secondes.

5.3 Quelques sous problèmes

Bien que le problème $P2|res^t1\cdot\cdot|C_{max}$ est NP-difficile, il existe certains cas particuliers où la solution optimale qui minimise la durée totale de l'ordonnancement est facile à trouver. Nous présentons dans cette partie deux cas particuliers et nous donnons pour chacun, l'algorithme qui détermine la solution optimale.

5.3.1 Problème $P2|res^t1\cdot\cdot, s_i = s, p_i = p|C_{max}$, $q \geq 2$

Dans ce cas, les temps de préparation ainsi que les temps d'exécution des tâches sont identiques égaux à s et p respectivement, et la disponibilité de la ressources R est supérieure ou égale à 2 ($q \geq 2$). Ce problème peut être résolu avec l'Algorithme-MM proposé dans le chapitre précédent.

Théorème 6. *[53] L'Algorithme − MM fournit une solution optimale pour le problème* $P2|res^t1\cdot\cdot, s_i = s, p_i = p|C_{max}$ *avec* $q \geq 2$ *en* $O(n^{2.5})$.

Exemple 8. *Considérons l'ordonnancement de 8 tâches* T_1, T_2,...,T_8 *sur deux machines avec une disponibilité* $q = 3$ *de la ressource R. Les temps de préparation sont égaux à 2 et les durées d'exécution sont égales à 1. Les demandes en ressource R sont données dans la Table 5.3.*

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
b_i	2	2	1	2	2	1	3	2

TAB. 5.3 – Les demandes en ressource R pour chaque tâche.

Le déroulement de l'Algorithme - MM est le suivant :

1. Construction du graphe $G = (T, E)$, voir la Figure. 5.5 :

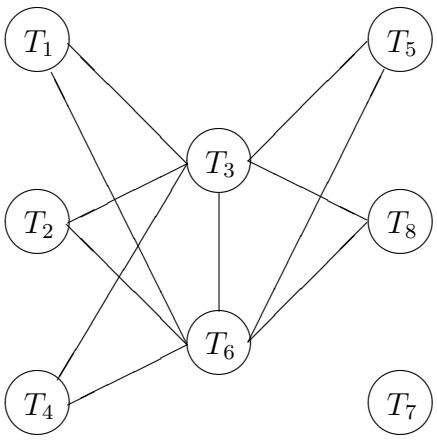


FIG. 5.5 – Graphe $G = (T, E)$ correspondant à l'exemple 8.

2. Un couplage maximum est $M = \{(T_1; T_3), (T_2; T_6)\}$.

3. L'ensemble des tâches non couplées est $Q = \{T_4, T_5, T_7, T_8\}$.

4. $Q \neq \emptyset$ et $s > p$ donc $C_{max} = |M|(s+p) + h.s + p = 15$.

5. L'ordonnancement optimal est représenté par la Figure 5.6.

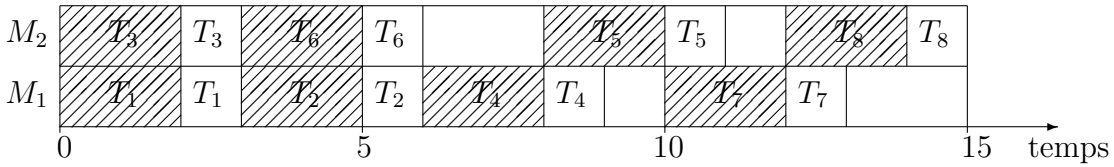


FIG. 5.6 – Solution optimale de l'exemple 8.

5.3.2 Problème $P2|res^t12\cdot, s_i = s, p_i = p|C_{max}$

Lorsque la ressource R est disponible seulement en 2 unités, on peut améliorer la complexité de l'Algorithme - MM en remarquant que les tâches qui ne nécessitent qu'une unité de la ressource R pour leurs préparation forment une clique dans le graphe G. Soit T' l'ensemble de ces tâches et soit Q l'ensemble des tâches qui nécessitent deux unités de la ressource R pour leur préparation et h la cardinalité de l'ensemble Q.

	Algorithme - MMA
1.	Traiter les $\left\lfloor \frac{\lvert T' \rvert}{2} \right\rfloor$ tâches sur M_1 et les $\left\lfloor \frac{\lvert T' \rvert}{2} \right\rfloor$ autres tâches sur M_2 en parallèle.
2.	Traiter les tâches de l'ensemble Q alternativement sur les deux machines dès que possible.
3.	$C_{max} = \begin{cases} \frac{n}{2}(s+p) & \text{si } Q = \emptyset \\ (\lfloor \frac{n}{2} \rfloor + 1)(s+p) & \text{si } Q \neq \emptyset, h \text{ est impair et } s \leq p \\ \frac{n}{2}(s+p) + s & \text{si } Q \neq \emptyset, h \text{ est pair et } s \leq p \\ \lvert M \rvert (s+p) + h.s + p & \text{si } Q \neq \emptyset \text{ et } s > p \end{cases}$

Théorème 7. *[53] L'Algorithme - MMA résout le problème* $P2|res^t12\cdot, s_i = s, p_i = p|C_{max}$ *en* $O(n)$.

Preuve. C'est un cas particulier du Théorème 5. ■

Exemple 9. *Nous disposons de* 7 *tâches à traiter sur deux machines parallèles identiques* M_1 *et* M_2 *avec* $q = 2$*, les temps de préparation ainsi que les temps d'exécution sont égaux à 1. Les demandes en ressource* R *pour chaque tâche sont données par la Table 5.4.*

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7
b_i	1	1	2	2	1	2	1

TAB. 5.4 – Les demandes en ressource R.

L'exécution de l'algorithme donne :

– $T' = \{T_1, T_2, T_5, T_7\}$, $|T'| = 4$.

– $Q = \{T_3, T_4, T_6\}$, $h = 3$.

– $Q \neq \emptyset$, h est impair et $s = p$, donc $C_{max} = (\lfloor \frac{n}{2} \rfloor + 1)(s + p) = 8$.

– L'application de l'Algorithme - MMA fournit la solution représentée par la Figure 5.7.

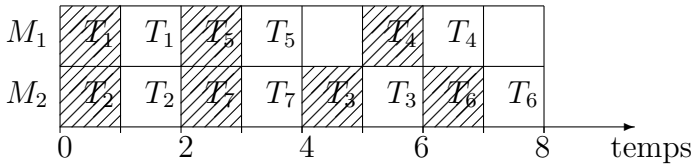


FIG. 5.7 – Solution générée par l'Algorithme - MMA.

5.4 Approche de résolution

Dans cette section, nous présentons tout d'abord les bornes inférieures développées pour le problème $P2|res^t1\cdot\cdot|C_{max}$, ensuite nous exposons les heuristiques pour résoudre ce dernier avec une étude expérimentale.

5.4.1 Bornes inférieures

Nous proposons dans ce qui suit des bornes inférieures pour le problème en se basant sur un graphe de compatibilité.

Graphe de compatibilité

Considérons le graphe $G = (T, E)$ où T est l'ensemble des sommets correspondant aux tâches à ordonnancer et où une paire de sommets (T_i, T_j) est dans E si et seulement si

$b_i + b_j \leq q$. On peut décomposer l'ensemble T en deux sous-ensembles $T = S \cup C$ tels que :

– S (stable) est l'ensemble des tâches qui vérifient la relation $b_i + b_j > q$, $\forall T_i, T_j \in S$.

– C (clique) est l'ensemble des tâches qui vérifient la relation $b_i + b_j \leq q$, $\forall T_i, T_j \in C$.

$G = (S \cup C$,E), qui est un graphe scindé, défini une relation de compatibilité entre les tâches. Deux tâches sont dites compatibles si leurs préparations respectives peuvent s'exécuter simultanément (en parallèle) sur les deux machines.

Pour construire le graphe G, il suffit de prendre :

– $S = \{T_i \ / \ b_i > \frac{q}{2}\}$.

– $C = \{T_i \ / \ b_i \leq \frac{q}{2}\}$.

Remarque 3. *Si* $\exists y < \frac{q}{2} \ / \ \forall T_i \in C, b_i \leq y$ *et* $\exists T_j \in S \ / \ b_j \leq q - y$ *alors* T_j *peut être reliée à toutes les tâches de* C *et donc on peut l'ajouter à la clique* C*.*

Pour augmenter la taille de la clique C, on choisit, parmi toutes les tâches qui vérifient la condition de la Remarque 3, celle qui a le plus grand temps d'exécution. En cas d'égalité, on choisit celle ayant le plus petit temps de préparation.

Remarque 4. *Si* $\exists y > \frac{q}{2} \ / \ \forall T_i \in S, b_i \geq y$ *et* $\exists T_j \in C \ / \ b_j > q - y$ *alors* T_j *peut être ajoutée au stable* S*.*

Pour augmenter la taille du stable S, on choisit, parmi toutes les tâches T_j qui vérifient la condition de la Remarque 4, celle qui a le plus grand temps de préparation. En cas d'égalité, on choisit celle ayant le plus grand temps d'exécution.

Une autre méthode, pour déterminer la clique C et le stable S, consiste à appliquer le résultat du Théorème suivant :

Théorème 8. *[36] Soit* $G = (V, E)$ *un graphe non orienté avec une séquence* $d_1 \geq d_2 \geq ... \geq d_n$ *de degrés et soit* $m = \max\{i | d_i \geq i - 1\}$*. Alors,* G *est un graphe scindé si et seulement si* $\sum_{i=1}^{m} d_i = m(m-1) + \sum_{i=m+1}^{n} d_i$*.*
Si c'est le cas, alors $\omega(G) = m$*.*

Le principe de cette méthode est le suivant :

Ranger les sommets du graphe par ordre décroissant de leurs degrés (soit $d_1 \geq d_2 \geq \ldots \geq d_n$), en cas d'égalité, choisir celui dont la tâche correspondante a le plus grand temps de préparation, ensuite le plus petit temps d'exécution. Puis calculer la taille de la clique $|C| = \max\{i/d_i \geq i-1\}$. La clique est composée des sommets dont le rang est inférieur ou égal à $|C|$ et le stable est formé des sommets dont le rang est supérieur strictement à $|C|$.

Remarque 5. *Si une tâche T_i est reliée à une tâche T_j, alors elle est reliée à toutes les tâches ayant une demande $b_d \leq b_j$.*

Calcul des Bornes

Premièrement, considérons les tâches du stable S (de cardinalité maximale comme indiqué dans la Remarque 4).

Proposition 4. *[53]* $LB1_S = \sum_{T_i \in S} s_i + \min_{T_i \in S}\{p_i\}$ *est une borne inférieure.*

Preuve. Les tâches du stable ne peuvent pas se préparer en parallèle sur les deux machines, donc la somme de leur temps de préparation plus le temps d'exécution de la dernière tâche traitée est une borne inférieure. ■

Proposition 5. *[53]* $LB2_S = \left\lceil \frac{\sum_{T_i \in S}(s_i+p_i) + \min_{T_i \in S}\{s_i\}}{2} \right\rceil$ *est une borne inférieure.*

Preuve. Comme la préparation de deux tâches ne peut se faire en parallèle sur les deux machines, alors l'une de ces deux dernières ne peut commencer son traitement qu'à partir de la date de fin de préparation de la tâche traitée en première position sur l'autre machine. Donc, dans le calcul de la borne inférieure, nous prenons en considération ce temps mort. ■

En combinant les deux bornes inférieures $LB1_S$ et $LB2_S$ on obtient une borne inférieure globale $LB_S = \max\{LB1_S, LB2_S\}$.

Considérons maintenant les tâches de la clique C.

Proposition 6. *[53]* $LB1 = \max\left\{ \left\lceil \frac{\sum_{T_i \in C}(s_i+p_i)}{2} \right\rceil, \max_{T_i \in C}(s_i+p_i) \right\}$ *est une borne inférieure.*

Notons $IT = \max\{0, 2LB_S - \sum_{T_i \in S}(s_i+p_i)\}$ le temps mort laissé par les tâches du stable traitées sur les deux machines.

Donc d'après les résultats précédents, on peut énoncer le Théorème suivant :

Théorème 9. *[53]* $\overline{LB} = LB_S + \max\{0, \lceil \frac{2LB1-IT}{2} \rceil\}$ *est une borne inférieure pour le problème* $P2|res^t1\cdot\cdot|C_{max}$.

Preuve. Pour le calcul de la borne inférieure, nous commençons par remplir le vide (le temps mort), laissé par les tâches du stable S, par les tâches de la clique C. Donc, $LB_S + \max\{0, \lceil \frac{2LB1-IT}{2} \rceil\}$ est une borne inférieure. ■

5.4.2 Approche heuristique

Pour résoudre le problème $P2|res^t1\cdot\cdot|C_{max}$, nous proposons huit méthodes approchées, leurs principe est d'appliquer les heuristiques $HS-r$ et $HP-r$ présentées précédemment avec les règles de priorités suivantes : $SPET$, SET, SPT, $LPET$, LET, LPT, LRR et SRR. Les six premières règles proposées sont définies comme dans le chapitre 4. Les deux dernières sont définies comme suit :

- LRR : Les tâches sont rangées dans l'ordre décroissant des demandes en ressource R.

- SRR : Les tâches sont rangées dans l'ordre croissant des demandes en ressource R.

Exemple 10. *Nous avons* 8 *tâches à ordonnancer sur deux machines avec une disponibilité* $q = 4$ *de la ressource* R. *les temps de préparation, les temps d'exécution ainsi que les demandes en ressource* R *pour chaque tâche sont donnés par la Table 5.5.*

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
s_i	2	1	3	3	1	2	4	1
p_i	3	2	1	3	4	1	2	1
b_i	3	2	1	1	3	4	1	2

TAB. 5.5 – Les demandes en ressource R.

En appliquant les règles définies précédemment on obtient les listes suivantes :

Les diagrammes de Gantt suivants représentent les solutions obtenues en appliquant les heuristiques $HS-r$ et $HP-r$ avec les règles $LPET$ et SET.

$LET = \{T_5, T_1, T_4, T_2, T_7, T_3, T_6, T_8\}$, $LPT = \{T_7, T_3, T_4, T_1, T_6, T_2, T_5, T_8\}$.
$LPET = \{T_4, T_7, T_1, T_5, T_3, T_2, T_6, T_8\}$, $LRR = \{T_6, T_1, T_5, T_2, T_8, T_3, T_4, T_7\}$.
$SET = \{T_3, T_6, T_8, T_2, T_7, T_1, T_4, T_5\}$, $SPT = \{T_2, T_5, T_8, T_1, T_6, T_3, T_4, T_7\}$.
$SPET = \{T_8, T_2, T_6, T_3, T_1, T_5, T_4, T_7\}$, $SRR = \{T_3, T_4, T_7, T_2, T_8, T_1, T_5, T_6\}$.

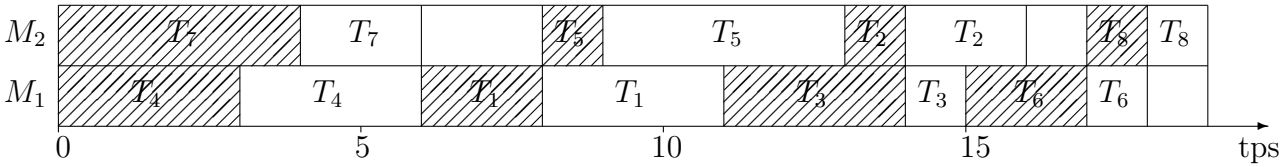


FIG. 5.8 – Solution obtenue par $HS - LPET$ avec $C_{max} = 19$.

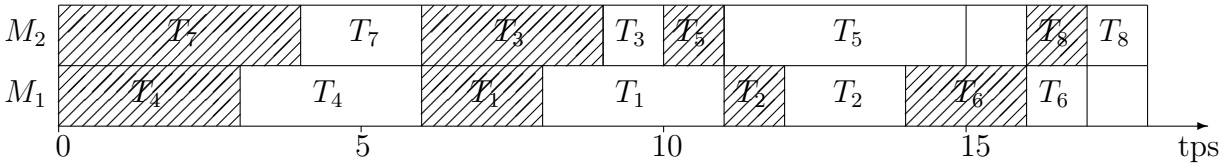


FIG. 5.9 – Solution obtenue par $HP - LPET$ avec $C_{max} = 18$.

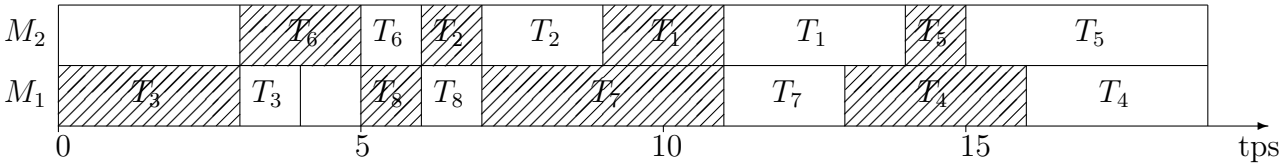


FIG. 5.10 – Solution obtenue par $HS - SET$ avec $C_{max} = 19$.

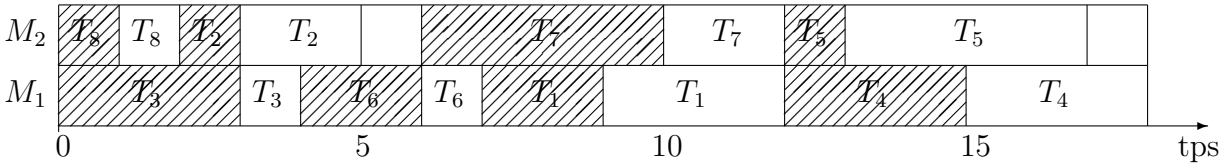


FIG. 5.11 – Solution obtenue par $HP - SET$ avec $C_{max} = 18$.

5.4.3 Expérimentations numériques

Nous avons mené trois types d'instances distinctes (voir Chapitre 4) pour le problème $P2|res^t 1 \cdot \cdot|C_{max}$. Pour chaque type, nous avons généré 100 instances aléatoires avec différentes valeurs du nombre de tâches $n = \{10, 50, 100, 500, 1000\}$. Le nombre d'unités de

la ressource R est fixé à $q = 10$, et les demandes en ressource R par les tâches ont été générées suivant la loi uniforme dans l'intervalle [1, 10].

Afin d'évaluer la performance des heuristiques proposées, nous avons besoin de les comparer à une solution optimale. Comme le problème est NP-difficile, cette comparaison sera faite par rapport à la borne inférieure $\overline{LB}$ de la solution optimale du problème $P2|res^t1\cdot\cdot|C_{max}$. Nous avons calculé le nombre de fois où chacune des heuristiques a fourni la meilleure solution et le nombre de fois où chacune d'elle est égale à la borne inférieure. Nous avons aussi calculé les déviations moyennes (Dev_{moy}) et maximales (Dev_{max}) ainsi que le temps de calcul moyen ($Time$).

Les résultats des expérimentations numériques sont illustrés dans les tables 5.6, 5.7 et 5.8. En effet, les méthodes proposées permettent de résoudre les problèmes jusqu'à $n = 1000$ en un temps qui ne dépasse pas 3 secondes.

n		$s_i, p_i \in [1, 10]$				$s_i, p_i \in [10, 50]$				$s_i, p_i \in [50, 100]$			
		HP-SPET	**HP-LPET**	**HP-LET**	**HP-LPT**	**HP-SPET**	**HP-LPET**	**HP-LET**	**HP-LPT**	**HP-SPET**	**HP-LPET**	**HP-LET**	**HP-LPT**
10	#Best	6	39	30	27	5	30	16	21	5	23	10	20
	#LB	0	4	4	6	0	0	2	1	0	0	1	0
	$\%Dev_{max}$	27,65	24	30	25,45	23,45	19,15	24,02	28,16	16,87	15,41	21,00	20,32
	$\%Dev_{moy}$	11,96	7,63	9,48	9,73	11,72	8,24	9,56	9,25	10,17	8,96	10,14	9,45
	$\%Tps_{moy}$	0	0	0	0	0	0	0	0	0	0	0	0
50	#Best	36	35	4	41	55	19	3	17	69	7	1	8
	#LB	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	9,48	9,15	13,71	8,82	6,48	8,42	11,93	8,94	4,62	6,73	7,66	6,64
	$\%Dev_{moy}$	4,15	4,38	6,74	4,32	3,49	4,22	5,89	4,44	2,47	3,44	4,42	3,93
	$\%Time$	0	0	0	0	0	0	0	0	0	0	0	0
100	#Best	68	18	0	25	86	2	0	11	98	3	0	0
	#LB	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	5,92	7,25	10,30	6,11	4,00	6,43	9,17	5,33	2,76	4,98	6,26	5,62
	$\%Dev_{moy}$	2,46	3,28	5,83	3,01	2,22	3,29	5,06	3,21	1,51	2,59	3,70	3,12
	$\%Time$	0	0	0	0	0	0	0	0	0	0	0	0
500	#Best	100	0	0	0	100	0	0	0	100	0	0	0
	#LB	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	2,36	3,39	6,69	3,71	1,72	3,03	5,29	3,48	0,93	2,69	3,77	3,16
	$\%Dev_{moy}$	1,40	2,07	4,02	2,36	1,12	2,04	3,65	2,42	0,63	1,29	2,59	2,09
	$\%Time$	1	0	0	0	1	1	0	0	1	1	0	0
1000	#Best	100	0	0	0	100	0	0	0	100	0	0	0
	#LB	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	2,12	2,54	5,37	3,31	1,39	2,66	3,99	2,89	0,75	1,58	2,75	2,30
	$\%Dev_{moy}$	1,31	1,73	3,63	2,35	0,95	1,58	3,20	2,23	0,52	0,91	2,22	1,93
	$\%Time$	3	3	2	2	3	3	2	2	3	3	2	2

TAB. 5.6 – Les temps de préparation et d'exécution prennent leurs valeurs dans un même intervalle.

De la Table 5.6, on remarque que les heuristiques basées sur la sélection Parallèle donnent des solutions meilleures que celles basées sur la sélection Sérielle. Ceci peut être expliqué par le fait que la sélection Parallèle minimise les temps mort en leurs affectant les tâches qui ont des demandes en ressources R inférieures ou égales à la quantité totale. Pour les problèmes à 10 tâches, les heuristiques $HP-LPET$, $HP-LET$ et $HP-LPT$ donnent de bonnes solutions et en comparant leurs solutions générées, on constate que l'heuristique $HP-LPET$ est la plus performante. Par ailleurs, plus le nombre de tâches augmente, plus l'heuristique $HP-SPET$ basée sur le tri des tâches dans l'ordre croissant de la somme des temps de préparation et d'exécution performe mieux, et nous remarquons aussi que les heuristiques $HP-LPET$, $HP-LET$ et $HP-LPT$ se dégradent rapidement. La

déviation moyenne de chaque heuristique est donnée par la Figure 5.12 pour le premier type d'instances.

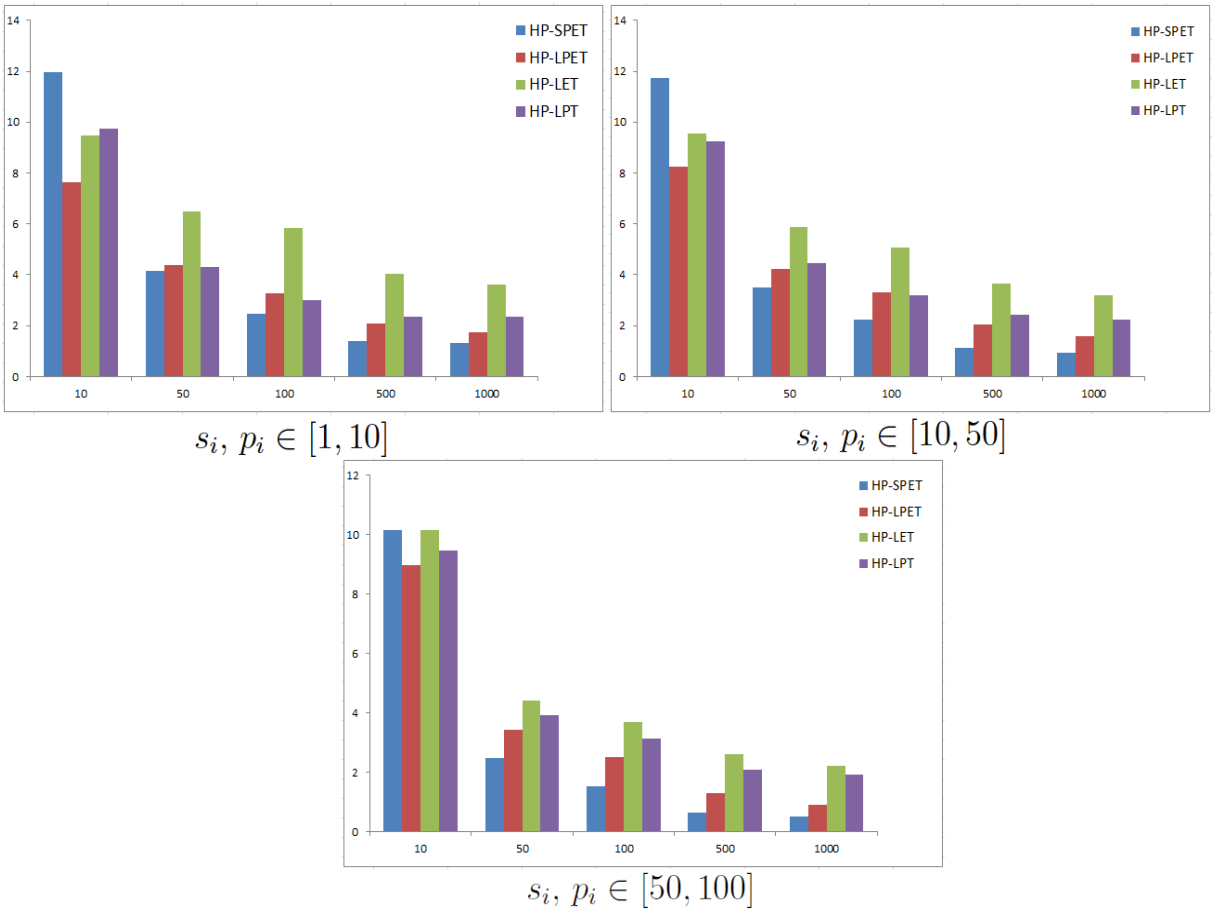
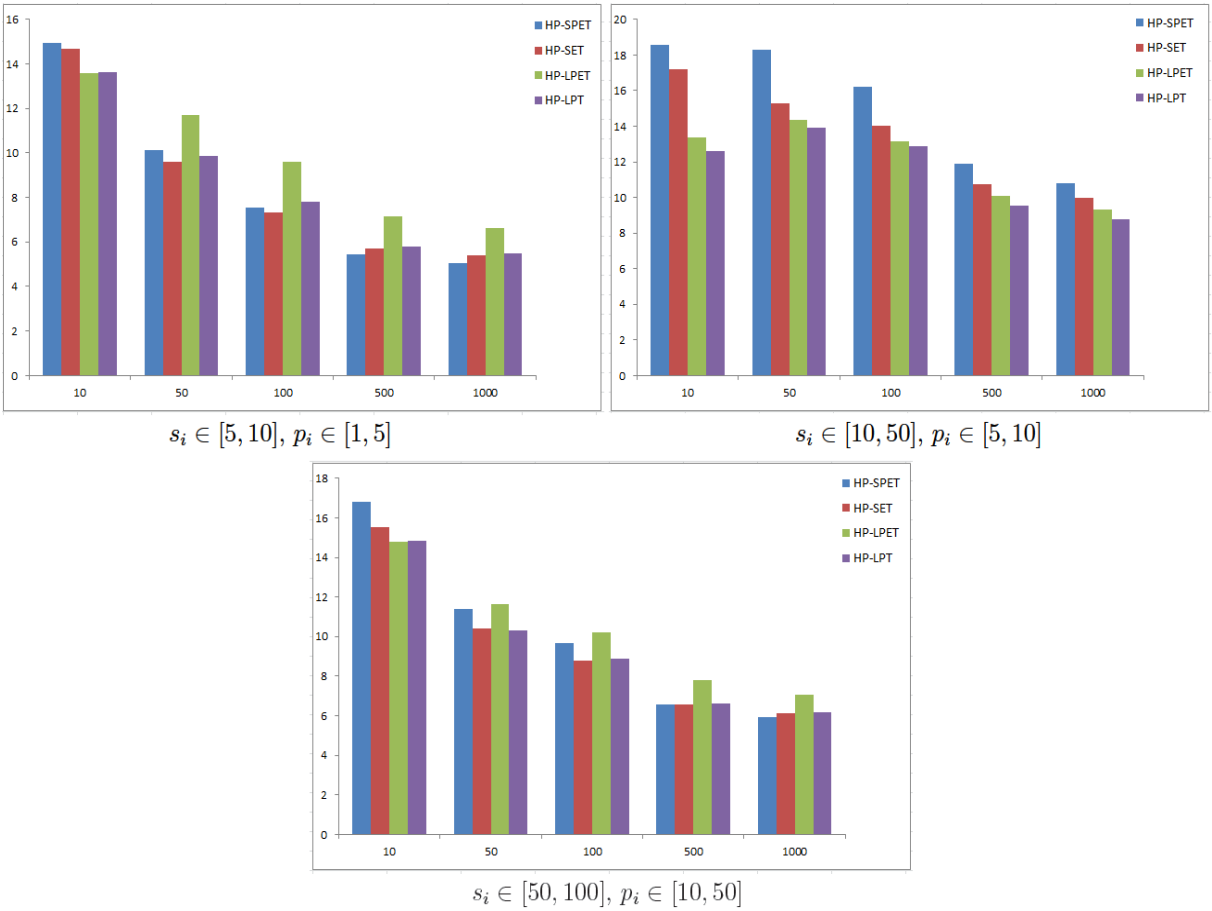


FIG. 5.12 – Déviations moyennes, s_i, p_i prennent leurs valeurs dans un même intervalle.

n		$s_i \in [5,10], p_i \in [1,5]$				$s_i \in [10,50], p_i \in [5,10]$				$s_i \in [50,100], p_i \in [10,50]$			
		HP-SPET	**HP-SET**	**HP-LPET**	**HP-LPT**	**HP-SPET**	**HP-SET**	**HP-LPET**	**HP-LPT**	**HP-SPET**	**HP-SET**	**HP-LPET**	**HP-LPT**
10	#Best	17	17	26	31	7	12	40	40	8	16	25	24
	#LB	0	1	1	8	5	2	0	0	0	0	0	0
	$\%Dev_{max}$	38,88	32,00	30,61	28,57	43,71	38,56	32,02	31,65	39,35	32,95	30,30	29,94
	$\%Dev_{moy}$	14,93	14,68	13,57	13,61	18,54	17,21	13,36	12,99	16,80	15,54	14,87	14,82
	$\%Time$	0	0	0	0	0	0	0	0	0	0	0	0
50	#Best	29	42	7	31	6	27	24	36	18	27	14	35
	#LB	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	20,23	18,65	20,63	8,65	29,13	25,77	24,12	24,76	22,73	19,13	22,18	19,31
	$\%Dev_{moy}$	10,13	9,59	11,72	9,88	18,27	15,28	14,34	13,94	11,37	10,43	11,65	10,31
	$\%Time$	0	0	0	0	0	0	0	0	0	0	0	0
100	#Best	33	43	2	29	10	30	17	39	17	39	4	39
	#LB	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	18,27	17,04	22,17	18,06	28,52	24,52	23,55	22,47	7,55	14,91	15,17	14,48
	$\%Dev_{moy}$	7,54	7,33	9,61	7,82	16,19	14,02	13,15	12,87	9,66	08,88	10,23	08,94
	$\%Time$	0	0	0	0	0	0	0	0	0	0	0	0
500	#Best	60	16	0	26	7	12	15	68	34	34	1	31
	#LB	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	8,30	8,10	10,57	7,98	17,44	15,90	15,02	14,87	10,07	8,97	10,98	8,77
	$\%Dev_{moy}$	5,44	5,70	7,17	5,81	11,87	10,74	10,09	9,66	6,57	6,55	7,79	6,69
	$\%Time$	1	0	1	0	1	0	1	0	1	0	1	0
1000	#Best	81	7	0	15	0	1	7	92	57	21	0	22
	#LB	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	6,76	8,80	9,47	7,43	16,82	15,19	14,25	13,86	8,03	7,50	9,84	7,90
	$\%Dev_{moy}$	5,04	5,40	6,63	5,47	10,78	10,06	9,32	8,88	5,92	6,11	7,08	6,15
	$\%Time$	3	2	3	2	3	2	3	2	3	2	3	2

TAB. 5.7 – Temps de préparation supérieurs aux temps d'exécution.

A la lecture de la Table 5.7, on constate les performances très approchées des heuristiques $HP-LPET$ et $HP-LPT$ pour la résolution des problèmes à 10 tâches. Pour $n \geq 100$, la meilleure solution est partagée entre les heuristiques $HP-SPET$, $HP-SET$ et $HP-LPT$. Quand les temps de préparation et d'exécution prennent leurs valeurs dans $[5,10]$ et $[1,5]$ respectivement, on constate la performance absolue et très nette de l'heuristique $HP-SPET$ pour les problème à 500 et 1000 tâches, par contre quand $s_i \in [10,50], p_i \in [5,10]$ c'est l'heuristique $HP-LPT$ qui fournit de meilleures solutions avec une déviation moyenne inférieure à 9%.



$s_i \in [5,10]$, $p_i \in [1,5]$ $\quad$ $s_i \in [10,50]$, $p_i \in [5,10]$

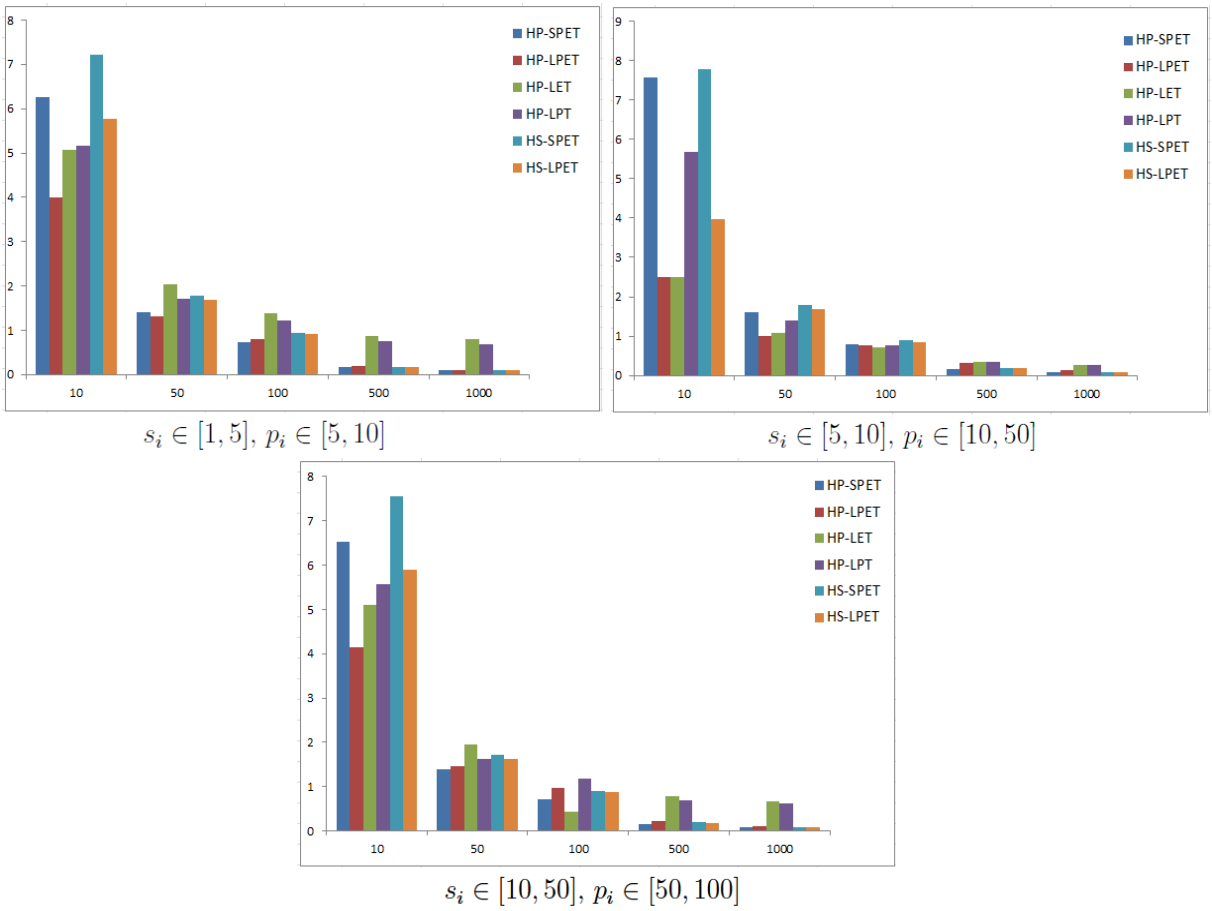
$s_i \in [50,100]$, $p_i \in [10,50]$

FIG. 5.13 – Déviations moyennes, $s_i > p_i$.

De la Table 5.8, l'analyse des performances des heuristiques nous indique que pour les problèmes de petites tailles, les heuristiques $HP-LPET$, $HP-LET$ et $HP-LPT$ fournissent de bons résultats, alors que pour les problèmes de grandes tailles c'est l'heuristique $HP-SPET$ qui est la plus avantageuse, puisque c'est elle qui génère le meilleur résultat pour la plupart des instances. Nous remarquons aussi la forte apparition des heuristiques $HS-SPET$ et $HS-LPET$ pour résoudre les problèmes à 500 et 1000 tâches quand les temps de préparation et d'exécution sont générés aléatoirement dans les intervalles $[1,5]$ et $[5,10]$ respectivement.

n		$s_i \in [1,5], p_i \in [5,10]$					$s_i \in [5,10], p_i \in [10,50]$					$s_i \in [10,50], p_i \in [50,100]$				
		HP-SPET	**HP-LPET**	**HP-LET**	**HP-LPT**	**HS-SPET**	**HS-LPET**	**HP-SPET**	**HP-LPET**	**HP-LET**	**HP-LPT**	**HS-SPET**	**HS-LPET**	**HP-SPET**	**HP-LPET**	**HP-LET**	**HP-LPT**	**HS-SPET**	**HS-LPET**
10	#Best	11	50	30	35	7	21	1	49	45	13	0	9	2	30	20	12	1	5
	#LB	1	18	11	14	1	3	0	4	2	2	0	0	0	1	0	1	0	0
	$\%Dev_{max}$	12,06	9,80	12,5	16,66	12,5	12,5	13,75	6,76	7,51	15,62	14,64	8,75	10,92	9,61	12,54	12,40	12,69	9,47
	$\%Dev_{moy}$	6,52	3,99	5,07	5,16	7,22	5,77	7,57	2,51	2,50	5,68	7,79	3,98	6,53	4,15	5,09	5,57	7,55	5,90
	$\%Time$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	#Best	26	40	18	30	3	9	1	35	21	21	0	1	19	17	9	25	0	3
	#LB	0	0	0	2	0	0	0	1	1	3	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	2,31	2,84	4,74	4,74	2,47	2,37	2,56	2,27	2,16	3,18	2,25	2,21	2,22	2,52	5,28	3,84	2,14	2,13
	$\%Dev_{moy}$	1,40	1,32	2,04	1,72	1,78	1,68	1,61	1,01	1,08	1,41	1,78	1,68	1,40	1,47	1,94	1,62	1,72	1,63
	$\%Time$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100	#Best	59	40	10	19	13	16	5	12	25	31	0	12	60	14	3	14	0	7
	#LB	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	1,21	1,62	3,47	2,86	1,45	1,45	1,11	1,37	1,77	1,99	1,12	1,16	0,94	1,45	2,87	2,68	1,09	1,07
	$\%Dev_{moy}$	0,73	0,81	1,38	1,21	0,93	0,91	0,79	0,77	0,72	0,77	0,90	0,85	0,71	0,97	1,43	1,18	0,90	0,87
	$\%Time$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
500	#Best	90	51	0	0	50	50	73	0	2	5	15	23	99	2	0	0	0	0
	#LB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	0,24	0,47	1,31	1,11	0,24	0,24	0,20	0,52	0,63	0,60	0,23	0,75	0,18	0,42	1,29	1,12	0,21	0,21
	$\%Dev_{moy}$	0,17	0,20	0,87	0,75	0,18	0,18	0,16	0,33	0,36	0,34	0,18	0,18	0,15	0,23	0,79	0,68	0,19	0,18
	$\%Time$	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
1000	#Best	99	77	0	0	76	76	98	0	0	0	18	24	100	0	0	0	0	0
	#LB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$\%Dev_{max}$	0,14	0,14	1,20	1,05	0,14	2,81	0,45	0,51	0,11	0,55	0,54	1,38	0,10	0,19	1,07	0,90	0,11	0,11
	$\%Dev_{moy}$	0,09	0,09	0,80	0,69	0,09	0,094	0,08	0,15	0,27	0,28	0,09	0,09	0,08	0,11	0,66	0,63	0,09	0,09
	$\%Time$	3	3	2	2	3	3	3	3	2	2	3	3	3	3	2	1	3	3

TAB. 5.8 – Temps de préparation inférieurs aux temps d'exécution.



$s_i \in [1,5]$, $p_i \in [5,10]$ $s_i \in [5,10]$, $p_i \in [10,50]$

$s_i \in [10,50]$, $p_i \in [50,100]$

FIG. 5.14 – Déviations moyennes, $s_i < p_i$.

Selon la Figure 5.14, on voit clairement que les déviations moyennes des heuristiques diminuent lorsque la taille des instances augmente. La déviation moyenne de l'heuristique $HS-SPET$ passe de $6,53\%$ pour les instances avec 50 tâches à $0,08\%$ pour les instances avec 1000 tâches quand $s_i \in [10,50]$ et $p_i \in [50,100]$.

Conclusion et perspectives

Le travail présenté dans cette thèse s'intéresse à l'étude du problème d'ordonnancement sur machines parallèles identiques sous contraintes de préparation dont l'objectif est de minimiser la date de fin de traitement des tâches. Nous avons traité deux sous-problèmes du problème général, le premier concerne le cas où le nombre de types de ressources est arbitraire et chaque type est disponible en une unité. Le deuxième problème concerne le cas où un seul type de ressources est disponible et en q unités.

Après avoir présenté rapidement dans le premier chapitre, les problèmes d'ordonnancement avec prise en compte des ressources et des temps de préparation, nous avons présenté dans le deuxième chapitre le problème étudié et nous avons proposé un état de l'art sur l'ordonnancement à machines parallèles identiques en insistant sur les problèmes avec temps de préparation et de ressources. La fin de ce chapitre a été consacrée aux principaux travaux trouvés dans la littérature pour le problème d'ordonnancement avec un seul type de ressources disponible en q unités et où chaque tâche ne nécessite qu'une unité pour sa préparation.

Dans le troisième chapitre, nous avons considéré le problème avec un nombre arbitraire de types de ressources tel que chaque tâche nécessite soit 0 soit 1 unité de chaque type pour la phase de préparation. Au cours de ce chapitre, nous avons étudié deux sous problèmes NP-difficiles au sens fort, deux sous problèmes faciles et des bornes inférieures ont été également proposées. Le quatrième chapitre a été dédié à la résolution de ce problème et ceci par des méthodes approchées basées sur des heuristiques et des métaheuristiques. Nous avons aussi mené différents tests expérimentaux sur des instances générées aléatoirement pour évaluer la performance des heuristiques ainsi que les métaheuristiques.

Dans le dernier chapitre, nous avons étudié le problème où chaque tâche nécessite q unités d'une même ressource pour sa préparation. Pour ce dernier, nous avons proposé une modélisation mathématique en nombres entiers, nous avons traité deux sous problèmes faciles et nous avons élaboré des algorithmes pour les résoudre, aussi des bornes inférieures ont été proposées. Nous avons exposé en fin de chapitre des heuristiques basées sur les

listes de priorité et nous l'avons terminé par une discussion des différents résultats.

La Table 5.9 résume nos principaux résultats.

Problème	Résultats de complexité	Référence
$P2\|res^t k11, s_i \in \{1,3,5\}, p_i = 1\|C_{max}$	NP-difficile au sens fort	[57]
$P2\|res^t k11, s_i \in \{1,3\}, p_i = 1, r_i \in \{0,r\}\|C_{max}$	NP-difficile au sens fort	[57]
$P2\|res^t 111, s_i \geq 1, p_i = 1\|C_{max}$	Algorithme $List - A$ $O(n)$	[57]
$P2\|res^t \cdot 11, s_i = s, p_i = p\|C_{max}$	Algorithme - MM $O(n^{2.5})$	[57]
$P2\|res^t 1 \cdot \cdot, s_i = s, p_i = p\|C_{max}$	Algorithme - MM $O(n^{2.5})$	[53]
$P2\|res^t 12 \cdot, s_i = s, p_i = p\|C_{max}$	Algorithme - MMA $O(n)$	[53]

Tab. 5.9 – Résumé des principaux résultats.

En résumé, quelques problèmes ont été étudiés dans cette thèse, certains sont difficiles, d'autres sont faciles, néanmoins il reste beaucoup de choses à faire, comme améliorer l'algorithme génétique hybride, proposer une méthode exacte et comparer les performances des approches de résolution développées dans cette thèse avec la solution optimale. Etudier d'autres sous problèmes en terme de complexité dans le cas où les temps de préparation ne prennent que deux valeurs possibles et où les temps d'exécution sont égaux à p. Considérer le même problème avec préemption des tâches, soit par rapport aux temps d'exécution ou par rapport aux temps de préparation. Etendre ce travail à d'autres critères d'optimalité, la somme pondérée par exemple (dans le cas préemptible ou non). Une dernière perspective est de considérer le problème à machines parallèles uniformes.

Bibliographie

[1] Abdelkhodaee A.H., Wirth A. Scheduling parallel machines with a single server : some solvable cases and heuristics, Computers & Operations Research, 29(3), 295-315, 2002.

[2] Abdelkhodaee A.H., Wirth A., Gan H.S. Equal processing and equal setup time cases of Scheduling parallel machines with a single server. Computers & Operations Research, 31, 1867-1889, 2004.

[3] Abdelkhodaee A.H., Wirth A., Gan H.S. Scheduling two parallel machines with a single server : the general case. Compputers & Operations Research, 33, 994-1009, 2006.

[4] Allahverdi A., Gupta J.N.D., Aldowaisan T. A review of scheduling research involving setup considerations. OMEGA The International Journal of Management Sciences, 27, 219–239, 1999.

[5] Allahverdi A., Ng C.T., Cheng T.C.E., Kovalyov M.Y. A survey of scheduling problems with setuo times or costs. European Journal of Operational Research, 187, 985–1032, 2008.

[6] Angelelli E., Filippi C. On the complexity of interval scheduling with a resource constraint. Theoretical Computer Science, vol. 412, 3650-3657, 2011.

[7] Bellanger A., Oulamara A. Scheduling hybrid flowshop with parallel batching machines and compatibilities. Computers & Operations Research, 36(6), 1982-1992, 2009.

[8] Bendraouche M., Boudhar M. Scheduling jobs on identical machines with agreement graph. Computers & Operations Research, 39, 382–390, 2012.

[9] Bendraouche M., Boudhar M., Oulamara A. Scheduling : Agreement graph vs resource constraints. European Journal of Operational Research, 240, 355–360, 2015.

[10] Blazewicz J. Selected topics in scheduling theory. Annals of Discrete Mathematics, vol. 31, 1-61, 1987.

[11] Blazewicz J., Barcelo J., Kubiak W. and Rock H. Scheduling tasks on two processors with deadlines and additional resources. European Journal of Operational Research, 26, 364-370, 1986.

[12] Blazewicz J., Cellary W., Slowinski R., Weglarz J. Scheduling under resource constraints : deterministic models. Annals of Operations Research, vol. 7, 1986.

[13] Blazewicz J., Ecker K. A linear time algorithm for restricted bin packing and scheduling problems. Oper. Res. Lett., 2, 80, 1983.

[14] Blazewicz J., Ecker K.H., Pesch E.,Schmidt G. and Weglarz J. Scheduling Computer and Manufacturing Processes. Springer-Verlag, Berlin, 2001.

[15] Blazewicz J., Kubiak W., Szwarcfiter J. Scheduling independent fixed-type tasks. In R. Slowinski and J. Weglarz (eds). Advances in Project Scheduling, Elsevier, Amsterdam, 225, 1989.

[16] Blazewicz J., Lenstra J.K., Rinnooy Kan A.H.G. Scheduling subject to resource constraints : classification and complexity. Discrete Applied Mathematics, vol. 5, 11-24, 1983.

[17] Brucker, P. Scheduling algorithms. Springer-Verlag, Berlin, Germany, 4th edition, 2004.

[18] Brucker P., Knust S. Complex Scheduling. Springer-Verlag Berlin, 2012.

[19] Cerny, V. Thermodynamical approach to the traveling salesman problem : an efficient simulation algorithm. J. of Optimization Theory and Applications, tome 45, n°1, P. 41-51, 1985.

[20] Chaari, T. Un algorithme génétique pour l'ordonnancement robuste : application au problème du flow shop hybride. Thèse, Université de Valenciennes et du Hainaut-Cambrésis, 2010.

[21] Cheng T.C.E., Gupta J.N.D., Wang G. A review of flowshop scheduling research with setup times. Production and Operations Management, 9, 262–282, 2000.

[22] Chrétienne, P. and Carlier, J. Problèmes d'ordonnancement : modélisation, complexité, algorithmes. Masson, 1988.

[23] Coffman, E.G., Garey, M.R. and Johnson, D.S. An application of bin-packing to multi-processor sheduling. SIAM Journal of computing, 17, 1-17, 1978.

[24] Crouhy, M. La gestion informatique de la production industrielle. Editions de l'usine édition, 1983.

[25] De Werra D. Preemptive scheduling, linear programming and networkflows. SIAM Journal on Algebraic and Discrete Methods, 5, 11-20, 1984.

[26] De Werra D. On the two-phase method for preemptive scheduling. European Journal of Operational Research, 37, 227-235, 1988.

[27] Dréo J., Péreowski, A., Siarry, P. et Taillard, E. Métaheuristiques pour l'optimisation difficile. Editions Eyrolles. 40, 2003.

[28] Figielska E. A new heuristic for scheduling the two-stage flowshop with additional resources.Computers and Industrial Engineering, vol. 54, 750-763, 2008.

[29] Figielska E. Heuristic algorithms for preemptive scheduling in a two-stage hybrid flowshop with additional renewable resources at each stage. Computers & Industrial Engineering, vol. 59, 509-519, 2010.

[30] Holland J.H. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, 1975.

[31] Gan HS., Wirth A., Abdelkhodaee A. A branche-and-price algorithm for the general case of scheduling parallel machines with a single server. Computer and Operations Research, 39, 2242-2247, 2012.

[32] Garey M.R., Johnson D.S. Computers and Intractability : A guide to the theory of NP-completeness. New York : Freeman, 1979.

[33] Garey M.R., Johnson D.S. Complexity results for multiprocessor scheduling under resource constraints. SIAM J. Comput. 4, 397, 1974.

[34] Glass C.A., Shafransky Y.M., Strusevich V.A. Scheduling for parallel dedicated machines with aSingle Server. Naval Research Logistics, vol. 47, 304-328, 2000.

[35] Goldberg, D. Genetic algorithm in search, optimisation and machine learning. Addison Wesley. 36, 1989.

[36] Golumbic, M.C. Algorithmic Graph theory and Perfect graph. Academic Press, 1980.

[37] Graham, R.L. Bonds on multiprocessing timing anomalies. SIAM Journal of Applied Mathematics, 17, 263-269, 1969.

[38] Graham R.L, Lawler E.L, Lenstra J.K and Rinnooy Kan A.H.G. Optimization and approximation in detrministic sequencing and scheduling theory : a survey, Ann. Discrete Mathematics, 5, 287-326, 1979.

[39] Hall N., Potts C.N., Sriskandarajah C. Parallel machine scheduling problem with a common server. Discrete Applied Mathematics, 102(3), 223-243, 2000.

[40] Hasani K., Kravchenko S.A, Werner F.Block models for scheduling jobs on two parallel machines with a single server. Computer and Operations Research, 41, 94-97, 2014.

[41] Hochbaum, D.S., Shmoys, D.B. Using dual approximation algorithms for scheduling problems : theorical and practical results. J. Assoc. Comput. Mach, 34, 144-162, 1987.

[42] Kariv O., Even S. An $O(n^{2.5})$ algorithm for maximum matching in general graphs. 16-th Annual Symposium on Foundations of Computer Science, IEEE, 100-112, 1975.

[43] Karp, R. Complexity of computer computations. R.E. Miller, J.W. Thatcher (eds.), Plenum Press, New-York, 1972.

[44] Kellerer H., Strusevich V.A. Scheduling parallel dedicated machines under a single non-sharedresource. European Journal of Operational Research, vol. 147, 345-364, 2003.

[45] Kellerer H., Strusevich V.A. Scheduling problems for parallel dedicated machines under multipleresource constraints. Discrete Applied Mathematics, vol. 133, 45-68, 2004.

[46] Kirkpatrick S., Gelatt C., Vecchi M. Optimization by simulated annealing. Science, tome 220, n°4598, P. 671-680, 1983.

[47] Kogan K. Optimal scheduling of parallel machines with constrained resources. European Journal of Operational Research, 170(3), 771-787, 2006.

[48] Koulamas, CP. Scheduling two parallel semiautomatic machines to minimize machine interference. Computers and Operations Research, 23(10), 945-956, 1996.

[49] Kovalyov M.Y., Shafransky Y.M. Uniform machine scheduling of unit-time jobs subject to resource constraints. Discrete Applied Mathematics, vol. 84, 253-257, 1998.

[50] Kravchenko S.A., Werner F. Parallel machine scheduling problem with a single server. Mathematical and Computer Modelling, 26(12), 1-11, 1997.

[51] Labbi, W. Ordonnancement sur machines parallèles identiques avec temps de préparation. Mémoire de Magister en mathématiques spécialité Recherche Opérationnelle, U.S.T.H.B, 2009.

[52] Labbi, W., Boudhar, M. Ordonnancement sur machines identiques en présence d'ouvriers spécialisés. 7ème Colloque sur l'Optimisation et les Systèmes d'Information (COSI'10), P. 193-202, Avril 18-20, Ouargla, Algerie, 2010.

[53] Labbi, W., Boudhar, M. Scheduling with preparation constraints. The Second International Symposium on Operational Research (ISOR'11), P. 159-160, Mai 30-02 June, Algeries, Algeria, 2011.

[54] Labbi, W., Boudhar, M., Oulamara, A. Algorithme génétique pour le problème d'ordonnancement sous contraintes de préparation. 9^{me} Colloque International sur l'Optimisation et les Systèmes d'information (COSI'12), 12-15 Mai, Tlemcen-Algérie, 2012.

[55] Labbi, W., Boudhar, M., Oulamara, A. Metaheuristics for scheduling on two identical machines with preparation times. The 26^{th} European conference on Operational Research. July 1-4, Rome, Italy, 2013.

[56] Labbi, W., Boudhar, M., Oulamara, A. Scheduling with preparation constraints. The 26^{th} ECCO conference, May 30- June 1, Paris, 2013.

[57] Labbi, W., Boudhar, M., Oulamara, A. Scheduling two identical parallel machines with preparation constraints. International Journal of Production Research, DOI : 10.1080/00207543.2014.978032. Accepted. To appear.

[58] Lopez, P., Roubellat, F. Ordonnancement de la production. Paris, Hermès Sciences Publications, 2001.

[59] Mokotoff, E.G. Scheduling to minimize the makespan on identical parallel machines : An LP-Based algorithm. Univrsity of Alcalé, 2003.

[60] Nakade K., Nishiwaki R. Optimal allocation of heterogeneous workers in a U-shaped production line. Computers and Industrial Engineering, 54, 432-440, 2008.

[61] Oulamara A., Finke G. and Kamgaing Kuiteing A. Flowshop scheduling problem with a batching machine and task compatibilities. Computers & Operations Research, 36(2), 391-401, 2009.

[62] Oulamara A., Rebaine D. and Serairi M. Scheduling the two-machine open shop problem under resource constraints for setting the jobs. Annals of Operations Research, 211, 333-356, 2013.

[63] Pinedo, M.L. Scheduling : Theory, Algorithms, and Systems, 4^{th} edition, Springer, 2010.

[64] Remy J. Resource constrained scheduling on multiple machines. Information Processing Letters, 91(4), 177-182, 2004.

[65] Rothkopf, M.H. Sheduling independent tasks on parallel processors. Management Science, 12, 347-447, 1966.

[66] Sahney V.K. Single server, two machines sequencing with switching time. Operations Research, 20, 24-36, 1972.

[67] Shuichi S., Mitsunori T. and Koichi Y. A note on greedy algorithms for the maximum weighted independent set problem. Discrete Applied Mathematics. 126, 313-322, 2003.

[68] Slowinski R. L'ordonnacement des tâches preemptives sur les processeurs indépendants en présence de ressources supplémentaires. RAIRO Informatique /Computer Science, 15(2), 155-166, 1981.

[69] Su, C. Online LPT algorithms for parallel machines scheduling with a single server. Journal of Combinatorial Optimization, 26, 480-488, 2013.

[70] Talbi, EG. : Metaheuristics from design to implementation, Jonh Wiley and sons, 2009.

[71] Werner F., Kravchenko S.A. Scheduling with Multiple Servers. Automation and Remote Control, vol. 71(10), 2109-2121, 2010.

[72] Yang W.H., Liao C.J. Survey of scheduling research involving setup times. International Journal of Systems Science, 30, 143–155, 1999.

[73] Zouba M., Baptiste P. and Rebaine D. Scheduling identical parallel machines and operators within a period based changing mode. Computers & Operations Research, 36(12), 3231-3239, 2009.

Annexe A

Expérimentations numériques du problème $Pm|res^t \cdot 11|C_{max}$

n	$\|R\|$		$s_i \in [1,5]$, $p_i \in [5,10]$						$s_i \in [1,10]$, $p_i \in [10,50]$						$s_i \in [10,50]$, $p_i \in [50,100]$					
			HP-SPET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1	HP-SPET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1	HP-SPET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1
10	1	#Best	6	66	33	44	25	25	0	41	40	12	8	8	1	42	16	15	16	16
		#LB	4	55	27	36	19	19	0	14	12	3	6	6	0	9	4	3	2	2
		$\%Dev_{max}$	10	8,51	11,11	9,61	12,76	12,76	14,86	7,18	8,38	12,33	14,97	14,97	9,46	6,52	8,3	9,76	12,34	12,34
		$\%Dev_{moy}$	4,6	1,29	3,16	2,55	4,2	4,2	6,24	1,74	1,7	5,56	5,96	5,96	4,68	1,62	2,91	2,75	3,95	3,95
	2	#Best	2	53	27	36	18	19	0	40	40	11	11	10	1	23	13	18	12	13
		#LB	0	31	18	23	11	12	0	13	12	2	5	5	0	5	2	2	3	3
		$\%Dev_{max}$	12,24	8,69	12,24	13,04	14,28	14,28	13,33	6,82	8,29	14,04	14,11	14,11	10,29	8,13	10,3	11,97	13,4	13,4
		$\%Dev_{moy}$	5,9	2,37	4,03	3,82	5,54	5,67	6,86	2,08	2,19	6,13	5,67	5,8	5,57	3,14	3,79	3,9	4,97	5,2
	5	#Best	7	41	20	23	9	10	0	29	26	12	8	8	3	27	9	17	10	8
		#LB	0	12	3	6	1	3	0	1	0	1	2	1	0	0	0	0	1	0
		$\%Dev_{max}$	13,33	11,11	14	16,66	17,64	17,64	13,73	6,89	7,94	17,21	14,74	15,89	10,63	9,95	13,4	14,25	16,91	16,91
		$\%Dev_{moy}$	7,25	4,48	5,96	6,49	7,6	7,7	7,84	3,06	3,36	6,27	6,41	6,41	6,9	4,78	6,24	6,25	7,35	7,74
	7	#Best	12	33	20	17	12	12	0	31	27	10	9	9	8	14	14	11	5	6
		#LB	0	4	4	5	0	0	0	0	2	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	11,76	12,5	15,55	15,68	17,77	20,83	13,12	13,07	11,76	15,3	19,18	19,18	11,78	11,17	13,53	16,02	17,63	17,63
		$\%Dev_{moy}$	7,37	5,6	7,45	7,73	8,81	8,85	8,47	3,92	4,01	7,14	7,8	7,65	7,36	6,23	7,16	8,09	9,01	8,66
50	1	#Best	1	73	15	29	26	26	0	38	29	19	12	12	0	31	12	20	11	11
		#LB	0	57	10	23	20	20	0	19	12	6	7	7	0	9	2	3	3	3
		$\%Dev_{max}$	1,66	1,21	2,82	1,94	2,43	2,43	1,88	0,95	0,89	2,53	2,76	2,76	1,64	1,52	2,55	1,96	2,31	2,31
		$\%Dev_{moy}$	1,04	0,23	0,93	0,68	0,88	0,88	1,29	0,36	0,37	1,01	1,1	1,1	1,11	0,52	0,92	0,82	1,13	1,13
	2	#Best	0	47	17	21	19	23	0	23	22	13	24	24	0	21	7	19	18	16
		#LB	0	24	7	9	10	9	0	3	3	3	2	5	0	1	0	3	0	3
		$\%Dev_{max}$	1,71	1,62	2,72	2,03	3,57	2,81	2,08	1,45	1,89	2,92	2,62	2,65	1,75	1,49	3,05	3,02	3,09	2,50
		$\%Dev_{moy}$	1,17	0,55	1,01	0,82	1,08	1,01	1,47	0,55	0,57	1,13	0,97	0,91	1,17	0,68	1,14	0,91	1,08	1,05
	5	#Best	18	33	11	19	25	25	0	28	16	14	17	13	8	14	8	13	21	17
		#LB	0	2	0	1	9	4	0	0	0	1	1	1	0	0	0	0	0	0
		$\%Dev_{max}$	2,09	3,11	4,58	3,54	4,38	4,33	2,46	1,56	1,93	2,94	3,34	3,14	1,78	2,57	4,32	3,9	3,77	3,66
		$\%Dev_{moy}$	1,31	1,18	1,93	1,57	1,48	1,57	1,56	0,69	0,8	1,36	1,19	1,19	1,34	1,36	1,81	1,64	1,51	1,59
	7	#Best	39	41	8	14	15	15	0	44	24	10	11	11	26	28	9	5	9	7
		#LB	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	2,04	2,82	5,83	6,09	5,57	5,46	2,29	1,69	1,77	3,94	4,13	3,98	2,05	3,41	5,05	5,69	5,26	4,61
		$\%Dev_{moy}$	1,39	1,43	2,55	2,37	2,19	2,21	1,58	0,83	0,97	1,68	1,62	1,57	1,41	1,52	2,27	2,31	2,36	2,32
100	1	#Best	0	84	19	27	14	14	0	34	22	11	11	11	0	39	13	16	12	12
		#LB	0	65	17	18	11	11	0	12	3	3	3	3	0	10	2	0	2	2
		$\%Dev_{max}$	0,81	0,6	2	1	1,24	1,24	0,93	0,49	0,63	1,25	1,47	1,47	0,74	0,61	1,18	0,93	1,23	1,23
		$\%Dev_{moy}$	0,55	0,08	0,45	0,35	0,48	0,48	0,68	0,24	0,28	0,47	0,54	0,54	0,55	0,25	0,54	0,42	0,54	0,54
	2	#Best	0	54	7	35	16	25	0	25	20	15	20	13	0	23	14	17	13	15
		#LB	0	28	4	18	11	14	0	5	5	3	4	2	0	1	0	1	0	2
		$\%Dev_{max}$	0,83	0,59	1,79	1,19	1,57	1,37	0,93	0,66	0,78	1,29	1,39	1,51	0,73	0,83	1,63	0,94	1,23	1,21
		$\%Dev_{moy}$	0,5	0,2	0,57	0,39	0,53	0,47	0,73	0,3	0,33	0,55	0,52	0,54	0,6	0,35	0,57	0,43	0,48	0,51
	5	#Best	15	34	5	21	30	40	0	23	7	13	29	18	7	12	5	17	20	22
		#LB	0	0	0	3	6	5	0	1	0	0	6	1	0	0	0	0	0	2
		$\%Dev_{max}$	1,02	1,37	3,08	2,04	1,95	1,65	1,03	0,91	0,91	1,71	1,4	1,38	0,83	1,49	2,02	1,78	1,69	1,89
		$\%Dev_{moy}$	0,64	0,6	1,14	0,76	0,66	0,58	0,77	0,39	0,49	0,7	0,53	0,56	0,67	0,72	1,09	0,82	0,71	0,71
	7	#Best	36	37	1	13	21	21	0	33	23	10	18	21	46	19	2	3	16	10
		#LB	0	1	0	0	4	2	0	0	0	1	2	3	0	0	0	0	0	0
		$\%Dev_{max}$	1,04	1,6	3,6	2,92	3,59	2,9	1,17	1,27	1,21	2,14	1,6	1,74	1,03	1,73	3,15	2,52	2,66	3,12
		$\%Dev_{moy}$	0,67	0,69	1,68	1,2	1	1,04	0,8	0,46	0,52	0,97	0,71	0,71	0,69	0,96	1,62	1,44	1,04	1,05
500	1	#Best	0	89	12	28	22	22	0	19	17	20	12	12	0	27	7	28	14	14
		#LB	0	66	9	20	12	12	0	6	1	4	1	1	0	2	2	2	0	0
		$\%Dev_{max}$	0,16	0,04	0,2	0,16	0,24	0,24	0,18	0,09	0,16	0,27	0,28	0,28	0,14	0,09	0,24	0,19	0,25	0,25
		$\%Dev_{moy}$	0,1	0,01	0,09	0,06	0,08	0,08	0,14	0,05	0,07	0,1	0,12	0,12	0,11	0,05	0,1	0,07	0,1	0,1
	2	#Best	0	82	10	30	16	17	0	10	17	18	22	24	0	20	5	35	15	20
		#LB	0	43	6	18	12	12	0	0	2	2	8	4	0	0	1	1	2	1
		$\%Dev_{max}$	0,16	0,08	0,4	0,19	0,24	0,24	0,19	0,11	0,19	0,27	0,24	0,25	0,14	0,19	0,42	0,19	0,22	0,23
		$\%Dev_{moy}$	0,12	0,02	0,11	0,07	0,11	0,11	0,14	0,07	0,08	0,09	0,09	0,09	0,12	0,07	0,12	0,07	0,09	0,09
	5	#Best	9	19	0	39	38	42	0	13	3	21	25	28	6	5	0	31	27	27
		#LB	0	0	0	14	11	13	0	0	0	1	3	4	0	0	0	0	2	0
		$\%Dev_{max}$	0,16	0,24	0,88	0,31	0,32	0,32	0,19	0,24	0,31	0,31	0,25	0,27	0,15	0,38	0,59	0,25	0,35	0,31
		$\%Dev_{moy}$	0,12	0,11	0,32	0,09	0,10	0,09	0,15	0,12	0,14	0,12	0,09	0,09	0,13	0,19	0,32	0,11	0,11	0,11
	7	#Best	25	12	0	9	43	52	2	12	2	9	28	40	31	1	0	3	28	43
		#LB	0	0	0	1	9	13	0	0	0	0	3	6	0	0	0	0	0	0
		$\%Dev_{max}$	0,16	0,4	1,26	0,69	0,43	0,47	0,19	0,33	0,43	0,4	0,36	0,30	0,15	0,67	1,06	0,9	0,41	0,37
		$\%Dev_{moy}$	0,13	0,18	0,66	0,28	0,12	0,11	0,15	0,13	0,2	0,17	0,12	0,11	0,13	0,34	0,59	0,28	0,14	0,13
1000	1	#Best	0	84	14	19	28	28	0	16	25	17	19	19	0	25	9	22	19	19
		#LB	0	59	13	14	22	22	0	5	1	4	4	4	0	1	1	1	3	3
		$\%Dev_{max}$	0,08	0,03	0,1	0,1	0,12	0,12	0,09	0,04	0,08	0,14	0,14	0,14	0,06	0,04	0,14	0,09	0,13	0,13
		$\%Dev_{moy}$	0,05	0,008	0,04	0,03	0,04	0,04	0,07	0,03	0,03	0,05	0,05	0,05	0,05	0,02	0,04	0,04	0,05	0,05
	2	#Best	0	83	10	30	17	17	0	8	17	22	22	18	0	16	9	32	10	17
		#LB	0	49	4	16	9	9	0	1	1	2	2	2	0	0	0	2	0	2
		$\%Dev_{max}$	0,08	0,03	0,17	0,1	0,12	0,12	0,09	0,07	0,09	0,13	0,12	0,12	0,07	0,09	0,21	0,1	0,12	0,12
		$\%Dev_{moy}$	0,05	0,01	0,06	0,03	0,05	0,05	0,07	0,03	0,04	0,05	0,05	0,05	0,06	0,03	0,06	0,03	0,05	0,05
	5	#Best	7	26	0	47	34	39	0	8	0	22	31	32	1	1	1	40	24	29
		#LB	0	3	0	13	17	19	0	0	0	0	3	4	0	0	0	1	3	1
		$\%Dev_{max}$	0,08	0,13	0,39	0,1	0,12	0,12	0,09	0,14	0,23	0,14	0,14	0,14	0,07	0,24	0,41	0,1	0,12	0,12
		$\%Dev_{moy}$	0,06	0,04	0,17	0,04	0,04	0,04	0,07	0,07	0,1	0,06	0,04	0,04	0,06	0,1	0,16	0,04	0,05	0,04
	7	#Best	18	12	0	20	42	43	4	12	1	14	24	37	16	0	0	6	32	48
		#LB	0	0	0	2	11	11	0	0	0	0	2	2	0	0	0	0	0	0
		$\%Dev_{max}$	0,08	0,21	0,67	0,29	0,15	0,16	0,09	0,16	0,25	0,22	0,16	0,14	0,08	0,37	0,7	0,34	0,16	0,16
		$\%Dev_{moy}$	0,06	0,09	0,38	0,09	0,05	0,05	0,07	0,08	0,15	0,08	0,06	0,05	0,06	0,2	0,38	0,12	0,06	0,05

TAB. A.1 – Temps d'exécution supérieurs aux temps de préparation, $m = 2$

n	$\|R\|$		$s_i \in [1,5]$, $p_i \in [5,10]$				$s_i \in [1,10]$, $p_i \in [10,50]$				$s_i \in [10,50]$, $p_i \in [50,100]$			
			HP-LPET	**HP-LPT**	**HP-SCN**	**HP-SCN1**	**HP-LPET**	**HP-LPT**	**HP-SCN**	**HP-SCN1**	**HP-LPET**	**HP-LPT**	**HP-SCN**	**HP-SCN1**
10	1	#Best	44	21	26	26	51	49	12	12	47	19	12	12
		#LB	34	15	15	15	13	11	2	2	8	1	0	0
		$\%Dev_{max}$	24,52	29,41	21,81	21,81	25,1	24,39	19,14	19,14	25,91	25,76	21,62	21,62
		$\%Dev_{moy}$	7,08	7,71	6,91	6,91	5,24	5,09	6,82	6,82	7,17	7,03	7,39	7,39
	2	#Best	31	29	26	27	38	39	32	32	25	22	21	16
		#LB	15	6	9	10	1	2	1	1	0	1	0	0
		$\%Dev_{max}$	27,45	27,77	22	22	22,84	24,19	15,2	15,2	26,56	30,92	22,46	22,46
		$\%Dev_{moy}$	10,16	9,78	8,74	8,71	8,14	8,24	7,22	7,3	10,19	10,19	8,77	8,95
	5	#Best	34	30	27	25	38	39	29	28	30	25	19	18
		#LB	5	2	1	1	3	1	3	3	1	0	0	0
		$\%Dev_{max}$	28,3	27,41	27,27	27,27	23,42	36,32	20,76	18,84	28,41	28,51	26,43	26,43
		$\%Dev_{moy}$	12,33	12,15	12,6	12,71	8,17	8	7,3	7,44	14,72	13,95	14,01	14,02
	7	#Best	33	26	17	17	34	25	23	24	30	20	13	13
		#LB	6	2	0	0	8	4	7	7	2	1	0	0
		$\%Dev_{max}$	30,9	32,6	26	26	30,95	30,95	36,3	32,73	30,52	31,34	26,73	26,73
		$\%Dev_{moy}$	11,27	11,5	11,5	11,54	6,82	6,9	6,63	6,51	12,03	12,86	12,38	12,44
50	1	#Best	30	18	28	28	28	27	24	24	31	16	24	24
		#LB	29	2	9	9	12	7	2	2	5	1	1	1
		$\%Dev_{max}$	13,7	10,4	11,6	11,69	14,01	13,4	14,34	14,34	14,9	12,7	15,9	15,9
		$\%Dev_{moy}$	3,78	2,74	2,57	2,57	3,18	3,1	3,32	3,32	3,47	3,17	2,63	2,63
	2	#Best	4	16	34	40	16	25	19	32	12	18	27	49
		#LB	3	1	4	6	1	0	1	1	1	0	0	1
		$\%Dev_{max}$	15,47	12	15,2	14,9	15,72	16,26	15,06	15,28	17,31	14,07	15,58	15,66
		$\%Dev_{moy}$	6,65	4,83	3,91	3,9	6,37	6,11	5,45	5,44	5,6	4,52	3,31	3,33
	5	#Best	1	10	45	38	9	17	46	33	0	19	42	34
		#LB	0	0	0	1	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	20,9	17,92	17,07	16,66	22,24	22,78	19,25	18,39	21,49	21,07	18,66	19,16
		$\%Dev_{moy}$	12,9	10,9	8,7	8,9	13,9	13,61	11,02	11,15	13,56	11,71	9,81	9,93
	7	#Best	0	20	40	31	15	17	46	37	8	16	36	32
		#LB	0	0	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	24,7	23,1	21,27	22,13	23,03	22,71	17,68	18,43	23,95	22,36	21,26	21,57
		$\%Dev_{moy}$	17,7	15,54	13,7	13,9	12,64	12,69	11,16	11,27	16,87	15,44	13,48	13,65
100	1	#Best	21	24	38	38	45	14	31	31	10	24	30	30
		#LB	20	10	13	13	18	3	5	5	6	0	0	1
		$\%Dev_{max}$	13,75	9,85	11,29	11,29	10,77	10,77	13,16	13,16	11,11	9,14	8,54	8,54
		$\%Dev_{moy}$	3,01	1,74	1,34	1,34	1,7	2	2,31	2,31	3,06	2,19	1,6	1,6
	2	#Best	2	11	40	51	27	23	29	25	6	21	27	34
		#LB	2	2	11	7	1	2	0	0	1	0	2	0
		$\%Dev_{max}$	13,12	9,16	10,1	9,4	12,51	12,67	13,96	13,85	12,61	9,52	9,15	9,04
		$\%Dev_{moy}$	4,17	2,68	1,81	1,74	4,01	4,03	4,02	4,07	4	3	2,36	2,28
	5	#Best	0	6	42	43	8	10	49	34	0	6	46	42
		#LB	0	0	0	0	0	0	0	0	0	0	1	0
		$\%Dev_{max}$	18,91	15,03	16,3	15,32	19,82	19,45	16,48	17,88	18,93	15,52	17,21	17,09
		$\%Dev_{moy}$	10,73	8,26	6,16	6,18	11,49	11,55	8,82	9,04	10,33	8,52	6,24	6,34
	7	#Best	0	5	48	41	5	5	53	38	1	5	54	38
		#LB	0	0	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	22,02	19,8	17,87	18,18	22,35	23,97	18,96	19,4	21,97	21,4	17,56	18,51
		$\%Dev_{moy}$	14,71	12,13	9,76	9,86	15,7	15,67	12,41	12,55	15,62	13,96	10,92	11,01
500	1	#Best	5	14	62	62	19	29	38	38	3	9	59	59
		#LB	5	2	10	10	12	2	2	2	2	0	0	0
		$\%Dev_{max}$	5,39	3,15	2,91	2,91	2,03	2,42	6,41	6,41	4,79	2,73	1,81	1,81
		$\%Dev_{moy}$	1,35	0,46	0,17	0,17	0,52	0,48	0,73	0,73	1,17	0,55	0,18	0,18
	2	#Best	2	3	50	75	20	29	28	38	0	10	39	49
		#LB	2	1	10	9	0	0	1	5	0	0	4	0
		$\%Dev_{max}$	4,76	2,62	2,46	2,66	6,51	6,74	10,5	10,34	5,78	3,52	4,67	5,03
		$\%Dev_{moy}$	1,9	0,84	0,4	0,31	1,06	1,01	1,37	1,34	1,93	1,18	0,72	0,66
	5	#Best	0	1	41	59	3	1	48	48	0	2	61	36
		#LB	0	0	1	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	9,6	6,39	5,94	6,60	11,53	10,26	10,22	10,08	8,13	6,27	6,4	6,54
		$\%Dev_{moy}$	5,55	3,47	1,88	1,85	5,81	5,66	3,85	3,86	5,47	4,01	2,33	2,44
	7	#Best	0	0	63	41	0	0	61	39	0	0	59	41
		#LB	0	0	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	12,26	10	6,48	6,66	15,48	15,28	12,67	12,68	12,35	10,66	7,29	7,7
		$\%Dev_{moy}$	8,75	6,23	3,38	3,49	10,86	10,74	7,17	7,3	9,21	7,32	4,44	4,53
1000	1	#Best	5	15	67	67	8	23	57	57	4	8	68	68
		#LB	5	2	12	12	4	1	2	2	4	0	0	0
		$\%Dev_{max}$	4,15	1,75	1,49	1,49	1,16	1,1	3,74	3,74	2,96	1,67	1,44	1,44
		$\%Dev_{moy}$	1	0,24	0,09	0,09	0,3	0,25	0,25	0,25	0,87	0,38	0,11	0,11
	2	#Best	1	3	63	73	8	31	37	31	0	0	47	59
		#LB	1	0	13	14	0	0	6	4	0	0	0	0
		$\%Dev_{max}$	4,35	1,51	1,85	1,36	2,14	1,83	4,08	4,21	3,5	2,1	2,05	2,3
		$\%Dev_{moy}$	1,41	0,52	0,22	0,12	0,61	0,53	0,71	0,65	1,31	0,72	0,22	0,15
	5	#Best	0	1	53	56	3	6	54	37	0	3	49	48
		#LB	0	0	3	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	6,91	4,12	3,76	3,58	6,13	6,23	5,35	5,22	6,3	4,85	4,7	4,44
		$\%Dev_{moy}$	3,94	2,19	0,86	0,86	3,7	3,5	2,44	2,5	4,08	2,81	1,47	1,48
	7	#Best	0	0	59	43	0	0	62	39	0	0	59	42
		#LB	0	0	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	9,62	7,12	5	5,06	11	10,81	8,7	9,1	8,76	7,15	5,93	6,05
		$\%Dev_{moy}$	7,1	4,71	2,3	2,35	7,98	7,84	4,65	4,73	6,92	5,28	2,78	2,83

TAB. A.2 – Temps d'exécution inférieurs aux temps de préparation, $m = 2$.

n	$\|R\|$		$s_i, p_i \in [1,10]$					$s_i \in, p_i \in [10,50]$					$s_i, p_i \in [50,100]$				
			HP-SPET	HP-LPET	HP-LPT	HP-SCN	HP-SCN1	HP-SPET	HP-LPET	HP-LPT	HP-SCN	HP-SCN1	HP-SPET	HP-LPET	HP-LPT	HP-SCN	HP-SCN1
10	1	#Best	1	56	26	21	21	0	42	20	9	9	1	27	19	13	13
		#LB	0	38	9	13	13	0	8	1	2	2	0	5	1	1	1
		$\%Dev_{max}$	30,76	17,91	20,83	20	20	21,63	16,66	18,42	23,27	23,27	18,42	11,36	13,68	16,62	16,62
		$\%Dev_{moy}$	9,48	3,69	5,09	6,25	6,25	8,02	3,76	4,83	6,67	6,67	6,88	4,82	4,89	5,55	5,55
	2	#Best	2	48	34	17	20	0	39	23	9	10	8	19	15	19	20
		#LB	0	23	11	4	7	0	3	0	1	1	0	3	1	0	0
		$\%Dev_{max}$	28,33	18,33	23,52	28,33	26	24,57	22,77	24,22	23,97	23,97	16,59	15,11	15,14	15,67	15,67
		$\%Dev_{moy}$	10,93	4,82	6,09	8,19	8,03	10,3	6,04	7,34	8,43	8,3	8,3	7,33	6,77	7,25	7,29
	5	#Best	5	30	19	7	6	8	26	20	12	9	13	18	13	11	9
		#LB	0	1	3	1	0	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	31,11	27,45	26,92	27,77	28,57	28,3	23,89	31,73	24,83	24,83	17,35	15,38	19,52	19,49	19,49
		$\%Dev_{moy}$	15,83	10,61	12,38	14,33	14,25	13,39	10,43	11,3	12,9	13,08	10,42	9,41	10,47	11,17	11,26
	7	#Best	8	35	23	13	13	3	27	17	7	9	9	25	9	8	9
		#LB	0	0	3	0	1	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	32,6	28,26	28,26	30,95	30,95	29,03	22,06	29,69	33,92	35,68	18,75	17,29	20,05	22,25	22,25
		$\%Dev_{moy}$	16,79	11,97	13,81	16,48	16,56	14,8	11,72	14,04	16,35	16,17	11,63	10,35	12,34	13,06	13,06
50	1	#Best	1	39	47	25	25	1	26	29	20	20	12	19	14	21	21
		#LB	0	28	22	14	14	0	8	4	2	2	0	7	0	1	1
		$\%Dev_{max}$	4,86	5,01	2,22	3,96	3,96	4,27	4,53	2,69	3,8	3,8	2,9	3,85	4,44	6,38	6,38
		$\%Dev_{moy}$	1,98	1,04	0,68	1,14	1,14	1,77	1,13	0,82	1,2	1,2	1,41	1,41	1,41	1,35	1,35
	2	#Best	2	32	46	28	33	1	16	23	27	27	16	6	7	30	21
		#LB	0	14	18	12	12	0	7	1	0	1	0	0	0	0	0
		$\%Dev_{max}$	6,47	8,63	5,6	7,97	7,04	4,35	8,19	4,49	5,82	5,68	3,82	4,22	4,1	5,43	5,57
		$\%Dev_{moy}$	2,4	1,65	1,04	1,52	1,31	1,98	1,84	1,3	1,35	1,3	1,57	1,92	1,79	1,56	1,59
	5	#Best	5	13	28	28	33	17	14	16	19	29	54	5	3	12	15
		#LB	0	0	1	0	3	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	10,11	10,86	10,86	12,13	11,85	8,1	9,91	8,45	9,26	9,27	5,06	5,98	6,89	8	8,12
		$\%Dev_{moy}$	4,7	4,77	3,89	3,97	3,67	3,92	4,22	3,85	3,81	3,75	2,49	3,75	3,96	3,51	3,37
	7	#Best	16	20	23	28	27	33	20	13	9	16	56	13	5	4	3
		#LB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	15,32	11,22	12,54	16,47	14,55	9,93	11,06	11,96	13,75	12,11	5,52	7,38	10,59	9,27	8,81
		$\%Dev_{moy}$	6,66	6,73	6,48	6,42	6,43	5,93	6,34	6,66	6,76	6,61	3,54	4,6	5,27	5,09	5,13
100	1	#Best	1	55	37	20	20	1	18	38	27	27	10	15	14	32	32
		#LB	0	43	17	9	9	0	5	2	4	4	0	4	0	1	1
		$\%Dev_{max}$	1,68	3,36	1,09	1,56	1,56	1,86	2,93	2,18	2,94	2,94	1,29	1,67	2,07	3,22	3,22
		$\%Dev_{moy}$	0,93	0,4	0,33	0,54	0,54	0,9	0,7	0,44	0,57	0,57	0,69	0,76	0,75	0,63	0,63
	2	#Best	1	23	50	23	31	5	10	27	27	26	27	3	9	21	28
		#LB	0	12	17	13	14	0	3	1	1	2	0	1	0	0	0
		$\%Dev_{max}$	2,43	4,86	1,61	3,78	3,37	4	4,91	2,58	4,59	3,75	1,35	3,27	2,85	3,1	2,84
		$\%Dev_{moy}$	1,11	0,88	0,39	0,79	0,61	1,1	1,15	0,68	0,85	0,73	0,76	1,14	1,04	0,91	0,84
	5	#Best	2	3	13	45	49	23	4	8	40	31	59	2	3	14	12
		#LB	0	0	0	3	3	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	6,91	7,43	5,93	6,88	8,41	4,57	5,93	5,35	6,69	5,58	2,69	4,8	5,85	5,02	4,51
		$\%Dev_{moy}$	2,64	2,98	2,15	1,61	1,61	2,32	3,19	2,59	2,01	2,07	1,45	2,48	2,58	2,09	2,04
	7	#Best	31	1	20	25	31	39	2	7	25	21	85	0	1	8	3
		#LB	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	8,49	9,37	8,71	7,76	8,66	6,89	9,46	8,68	8,07	8,68	3,58	5,67	5,84	6,26	5,98
		$\%Dev_{moy}$	4,13	5,37	4,49	3,96	3,97	3,89	5,08	4,82	4,18	4,33	2,13	3,56	3,9	3,4	3,43
500	1	#Best	0	69	34	23	23	0	10	35	27	27	9	9	19	34	34
		#LB	0	52	19	16	16	0	3	0	1	1	0	6	1	1	1
		$\%Dev_{max}$	0,25	0,5	0,19	0,29	0,29	0,26	0,84	0,17	0,28	0,28	0,17	0,75	0,5	0,69	0,69
		$\%Dev_{moy}$	0,17	0,05	0,05	0,09	0,09	0,17	0,16	0,07	0,09	0,09	0,13	0,19	0,14	0,11	0,11
	2	#Best	1	29	40	33	34	1	5	40	20	27	18	0	8	24	38
		#LB	0	15	16	19	21	0	1	0	2	3	0	0	0	1	1
		$\%Dev_{max}$	0,33	1,41	0,18	1,3	0,32	0,4	1,06	0,26	0,55	0,31	0,22	0,58	0,62	0,79	0,51
		$\%Dev_{moy}$	0,18	0,2	0,07	0,13	0,09	0,18	0,31	0,09	0,14	0,1	0,14	0,28	0,25	0,17	0,12
	5	#Best	1	0	15	35	69	11	0	0	44	48	65	0	0	21	14
		#LB	0	0	0	6	16	0	0	0	0	0	0	0	0	3	0
		$\%Dev_{max}$	1,62	2,98	1,54	0,98	1,24	0,89	2,3	1,44	1,43	1,27	0,63	1,88	1,63	1,77	1,65
		$\%Dev_{moy}$	0,51	1,22	0,4	0,23	0,15	0,42	1,27	0,57	0,28	0,25	0,27	1,05	0,9	0,47	0,49
	7	#Best	5	0	2	45	58	18	0	1	35	46	88	0	0	7	6
		#LB	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	2,33	4,31	3	2,66	1,89	2,19	4,05	2,9	3,03	2,43	1,17	2,81	2,67	2,18	2,24
		$\%Dev_{moy}$	1,21	2,65	1,43	0,63	0,61	1,17	2,56	1,77	0,95	0,9	0,65	1,87	1,83	1,1	1,08
1000	1	#Best	0	76	31	21	21	0	10	38	26	26	6	6	18	34	34
		#LB	0	56	22	15	15	0	5	2	1	1	0	5	0	1	1
		$\%Dev_{max}$	0,16	0,36	0,09	0,14	0,14	0,1	0,71	0,09	0,12	0,12	0,07	0,35	0,22	0,11	0,11
		$\%Dev_{moy}$	0,09	0,01	0,02	0,05	0,05	0,08	0,11	0,03	0,04	0,04	0,06	0,1	0,06	0,05	0,05
	2	#Best	0	46	48	26	21	1	2	52	22	25	12	0	16	16	42
		#LB	0	24	23	11	8	0	0	1	5	4	0	0	0	0	1
		$\%Dev_{max}$	0,16	0,36	0,08	0,57	0,14	0,12	0,67	0,11	0,33	0,15	0,1	0,48	0,7	0,82	0,47
		$\%Dev_{moy}$	0,09	0,05	0,03	0,05	0,05	0,08	0,19	0,04	0,07	0,05	0,07	0,17	0,11	0,09	0,05
	5	#Best	1	0	38	41	51	10	0	5	40	52	58	0	0	24	18
		#LB	0	0	3	6	12	0	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	0,36	1,62	0,45	0,37	0,61	0,38	1,75	0,85	0,71	0,59	0,23	1,03	1,09	0,81	0,69
		$\%Dev_{moy}$	0,17	0,59	0,09	0,08	0,06	0,19	0,82	0,25	0,13	0,09	0,11	0,67	0,56	0,22	0,19
	7	#Best	1	0	0	37	66	2	0	0	45	55	77	0	0	11	12
		#LB	0	0	0	2	5	0	0	0	0	1	0	0	0	0	0
		$\%Dev_{max}$	1,15	2,81	1,44	1,1	0,98	1,01	2,39	1,85	0,89	0,92	0,62	1,93	1,67	1,21	1,25
		$\%Dev_{moy}$	0,59	1,76	0,7	0,22	0,18	0,55	1,76	1,01	0,3	0,27	0,3	1,3	1,21	0,49	0,49

TAB. A.3 – Temps de préparation et d'exécution prennent leurs valeurs dans un même intervalle, $m = 2$

n	$\|R\|$		HP-SPET	HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1
10	1	#Best	2	5	42	17	20	7	7
		#LB	0	0	8	4	1	0	0
		$\%Dev_{max}$	52,17	45,83	36,84	36,36	42,85	50	50
		$\%Dev_{moy}$	28,23	26,77	14,92	18,85	18,85	24,05	24,05
	2	#Best	0	2	46	22	21	10	10
		#LB	0	0	10	5	1	1	1
		$\%Dev_{max}$	50	47,61	36,84	36	41,66	42,85	42,85
		$\%Dev_{moy}$	28,57	26,73	15,71	18,63	18,95	21,51	21,45
	5	#Best	3	1	52	46	21	17	22
		#LB	0	0	23	22	6	8	9
		$\%Dev_{max}$	60	57,14	36	33,33	42,85	55	55
		$\%Dev_{moy}$	20,72	19,85	9,78	10,37	13,49	14,89	14,91
	7	#Best	1	1	52	55	12	17	18
		#LB	0	0	28	37	2	7	7
		$\%Dev_{max}$	50	41,66	29,16	33,33	45,45	50	35
		$\%Dev_{moy}$	16,13	15,8	6,36	6,87	11,6	12,14	12,05
50	1	#Best	1	18	5	12	24	21	21
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	27,45	24,75	27,72	28,43	27,08	27,61	27,61
		$\%Dev_{moy}$	13,14	9,94	15,48	13,27	10,03	9,74	9,74
	2	#Best	4	20	21	6	14	15	25
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	31,31	26,73	23,23	26,78	25,25	27,08	22,22
		$\%Dev_{moy}$	16,02	13,07	12,77	15,13	13,2	13,13	12,24
	5	#Best	1	5	33	3	25	28	21
		#LB	0	0	0	0	0	0	00
		$\%Dev_{max}$	32,35	26,47	25,68	33,01	25,49	25,47	25,25
		$\%Dev_{moy}$	22,41	18,28	15,19	19,97	15,82	15,18	15,4
	7	#Best	0	6	34	2	16	26	27
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	33,96	37,62	25,74	30,3	28,71	29,7	29,52
		$\%Dev_{moy}$	22,41	19,69	16,56	20,87	18,09	16,72	16,72
100	1	#Best	9	30	0	6	31	19	19
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	26,9	20,39	23,71	20,72	26,39	18,78	18,78
		$\%Dev_{moy}$	9,53	7,39	13,94	10,59	7,56	7,8	7,8
	2	#Best	1	14	8	2	23	11	47
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	25,37	17,32	22,27	24,75	19,4	21,78	22,27
		$\%Dev_{moy}$	11,04	9,23	10,78	11,84	9,18	9,02	8,01
	5	#Best	0	3	24	0	10	33	43
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	27,75	25,83	21,95	28,57	21,02	19,5	20,87
		$\%Dev_{moy}$	19,03	15,35	13,38	18,71	14,14	12,41	12,34
	7	#Best	0	2	21	0	4	52	33
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	32,65	24,63	23,61	31,86	26,8	23,15	21
		$\%Dev_{moy}$	22,28	18,37	16,45	22,17	17,74	14,64	15,08
500	1	#Best	16	11	0	0	71	6	6
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	8,4	6,91	15,82	10,68	6,22	8,9	8,9
		$\%Dev_{moy}$	3,38	3,08	11,09	6,28	1,86	3,83	3,83
	2	#Best	27	17	0	0	21	4	36
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	13,77	11,12	14,96	17,02	9,35	11,92	11,31
		$\%Dev_{moy}$	5,28	5,34	8,01	8,2	5,3	6,54	4,9
	5	#Best	0	0	38	0	2	25	39
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	14,82	12,54	10,28	16,1	12,05	11,85	11,46
		$\%Dev_{moy}$	9,9	7,88	6,53	11,67	7,27	6,39	6,37
	7	#Best	0	0	5	0	1	57	40
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	19,47	15,96	14,17	19,41	13,98	11,74	12,57
		$\%Dev_{moy}$	14,19	11,18	9,92	15,45	10,37	7,91	8,11
1000	1	#Best	20	2	0	0	79	1	1
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	7,71	6,66	14,6	11,27	5,63	7,59	7,59
		$\%Dev_{moy}$	2,58	2,66	10,91	5,8	1,49	3,62	3,62
	2	#Best	47	15	0	0	12	1	29
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	8,08	7,05	9,63	10,86	7,33	10,02	8,06
		$\%Dev_{moy}$	4,33	4,7	7,41	7,45	4,9	6,18	4,46
	5	#Best	2	0	74	0	0	14	12
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	10,95	9,2	7	13,96	8,3	8,07	7,9
		$\%Dev_{moy}$	7,74	6,44	4,66	9,65	5,96	5,42	5,42
	7	#Best	0	0	5	0	0	54	44
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	15,19	12,52	10,04	16,13	11,18	9,9	9,48
		$\%Dev_{moy}$	11,36	9,21	7,55	13,12	8,21	6,33	6,35

TAB. A.4 – Temps d'exécution supérieurs aux temps de préparation, $s_i \in [1, 5], p_i \in [1, 10]$, $m = 5$.

n	$\|R\|$		HP-SPET	HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1
10	1	#Best	0	0	65	65	5	2	2
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	77,96	74,57	37,83	35,13	62,85	61,76	61,76
		$\%Dev_{moy}$	30,45	30,43	11,68	11,68	26,57	26,92	26,92
	2	#Best	0	0	48	55	5	4	3
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	63,15	63,15	35,08	35,08	63,15	59,7	59,7
		$\%Dev_{moy}$	37,38	36,87	16,35	15,95	30,08	31,68	32,12
	5	#Best	0	0	63	63	4	4	2
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	72,41	72,41	39,13	40,98	66,19	70,68	70,68
		$\%Dev_{moy}$	42,96	43,03	21,55	21,29	37,16	39,15	39,65
	7	#Best	1	1	50	59	6	3	3
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	62,9	66,12	41,09	43,83	64,51	76,11	74,62
		$\%Dev_{moy}$	44,35	44,39	24,25	23,84	38,92	41,88	41,57
50	1	#Best	2	5	22	18	23	15	15
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	10,16	10,98	12,46	12,75	12,17	14,28	14,28
		$\%Dev_{moy}$	6,26	6,11	5,26	5,36	5,26	5,73	5,73
	2	#Best	2	4	26	17	17	13	16
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	9,57	9,57	10,45	10,7	12,96	12,83	11,64
		$\%Dev_{moy}$	6,8	6,68	5,27	5,72	6,03	6,65	6,16
	5	#Best	7	5	16	20	6	16	15
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	12,02	13,88	13,52	14,08	14,28	15,72	16,66
		$\%Dev_{moy}$	8,36	8,51	7,92	8,12	9,05	8,71	9,11
	7	#Best	11	12	23	13	12	15	7
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	17,55	17,55	17,12	15,13	18,22	19,94	18,45
		$\%Dev_{moy}$	10,02	9,93	9,81	9,84	10,99	11,01	11,29
100	1	#Best	2	7	10	12	28	22	22
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	4,37	4,63	9,02	9,59	5,78	6,04	6,04
		$\%Dev_{moy}$	3,15	3,02	3,86	3,89	2,55	2,73	2,73
	2	#Best	5	16	2	5	18	14	28
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	5,41	4,68	9,02	8,77	6,7	8,55	6,38
		$\%Dev_{moy}$	3,31	3,3	4,2	4,45	3,32	3,77	3,32
	5	#Best	31	25	5	2	13	12	21
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	6,08	6,78	9,12	10,89	9,01	9,35	8,89
		$\%Dev_{moy}$	4,16	4,24	5,86	6,16	5,35	4,88	4,72
	7	#Best	34	33	1	0	7	22	11
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	7,52	6,78	10,82	11,03	10,11	9,94	11,36
		$\%Dev_{moy}$	5,12	5,17	7,38	7,87	6,71	6,13	6,29
500	1	#Best	1	6	0	0	31	23	23
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	0,73	0,96	5,33	6,51	0,93	0,94	0,94
		$\%Dev_{moy}$	0,63	0,62	3,74	3,42	0,48	0,5	0,5
	2	#Best	32	31	0	0	17	2	20
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	0,81	1,03	4,64	4,37	1,92	3,1	1,59
		$\%Dev_{moy}$	0,63	0,66	2,65	2,86	0,84	1,67	0,79
	5	#Best	86	17	0	0	0	0	0
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	1,28	1,48	5,67	6,05	2,84	2,52	2,39
		$\%Dev_{moy}$	0,84	1,02	3,56	3,93	1,92	1,59	1,59
	7	#Best	81	18	0	0	0	1	0
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	1,92	1,92	6,2	6,91	3,92	3,13	3,36
		$\%Dev_{moy}$	1,19	1,39	4,75	5,19	2,83	2	1,96
1000	1	#Best	1	6	0	0	35	33	33
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	0,36	0,49	5,11	5,11	0,47	0,53	0,53
		$\%Dev_{moy}$	0,31	0,3	3,8	3,32	0,24	0,24	0,24
	2	#Best	35	41	0	0	11	0	19
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	0,53	0,48	3,28	3,46	1,33	2,43	1,01
		$\%Dev_{moy}$	0,31	0,31	2,36	2,51	0,56	1,54	0,46
	5	#Best	95	7	0	0	0	0	0
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	0,81	1	4,17	4,5	1,92	1,93	1,91
		$\%Dev_{moy}$	0,49	0,66	2,89	3,31	1,46	1,29	1,24
	7	#Best	93	7	0	0	0	0	0
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	1,01	1,16	5,03	5,55	2,73	2,17	2,03
		$\%Dev_{moy}$	0,66	0,84	3,91	4,35	2,06	1,44	1,41

TAB. A.5 – Temps d'exécution supérieurs aux temps de préparation, $s_i \in [5, 10], p_i \in [10, 50]$, $m = 5$.

n	$\|R\|$		HP-SPET	HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1
10	1	#Best	0	2	18	11	8	3	3
		#LB	0	0	4	2	0	2	2
		$\%Dev_{max}$	50,76	46,88	42,32	41,36	44,89	48,43	48,43
		$\%Dev_{moy}$	29,67	27,74	18,46	19,35	20,38	24,28	24,28
	2	#Best	2	1	19	9	10	15	17
		#LB	0	0	3	4	0	1	1
		$\%Dev_{max}$	54,97	54,27	40,86	36,05	41,53	42,42	41,91
		$\%Dev_{moy}$	27,8	26,18	16,03	17,92	19,97	18,79	18,67
	5	#Best	1	3	33	26	11	10	12
		#LB	0	0	8	7	0	2	2
		$\%Dev_{max}$	50,25	46,11	32,75	40,1	54,73	44,66	44,66
		$\%Dev_{moy}$	22,08	22	12,06	13,29	16,16	17,7	17,75
	7	#Best	2	0	34	33	9	9	8
		#LB	0	0	10	16	1	0	0
		$\%Dev_{max}$	44,84	47,32	34	40,88	43,87	37,86	37,05
		$\%Dev_{moy}$	17,4	18,16	8,68	9,53	12,79	13,97	14,06
50	1	#Best	2	10	0	2	33	11	11
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	29,15	25,75	30,84	31,22	27,71	27,54	27,54
		$\%Dev_{moy}$	13,56	10,75	15,56	14,63	10,14	11,26	11,26
	2	#Best	2	16	13	1	24	18	20
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	30,88	23,23	27,27	28,15	24,48	25,55	23,18
		$\%Dev_{moy}$	14,58	12,58	13,31	15,36	12,6	12,78	12,44
	5	#Best	3	5	18	2	12	29	30
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	29,15	29,91	28,12	30,18	26,82	22,4	23,74
		$\%Dev_{moy}$	21,32	18,61	17,23	20,65	17,32	15,63	15,76
	7	#Best	1	1	29	1	21	23	28
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	30,24	30,3	27,95	29,27	27,91	31,36	32,64
		$\%Dev_{moy}$	22,29	20,01	17,52	21,76	18,53	17,57	17,49
100	1	#Best	4	21	0	2	44	9	9
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	18,43	17,44	23,08	23,98	18,63	17,64	17,64
		$\%Dev_{moy}$	9,16	7,52	14,08	12,27	6,88	8,73	8,73
	2	#Best	5	21	4	0	16	21	26
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	20,92	18,34	20,51	21,6	18,91	20,55	19,7
		$\%Dev_{moy}$	10,39	8,65	10,86	12,56	9,24	8,52	8,22
	5	#Best	0	11	10	0	14	41	26
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	23,36	21,63	20,05	26,39	21,39	20,17	19,55
		$\%Dev_{moy}$	15,69	13,14	13,55	17,29	12,83	11,34	11,63
	7	#Best	0	2	11	0	7	52	29
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	26,08	25,52	22,17	28,8	23,98	21,12	20,22
		$\%Dev_{moy}$	19,11	17,01	16,36	20,37	16,37	14,14	14,33
500	1	#Best	48	3	0	0	47	1	1
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	7,49	6,56	15,32	12,72	6,29	10,96	10,96
		$\%Dev_{moy}$	22,54	3,41	11,65	7,88	2,5	5,17	5,17
	2	#Best	51	20	0	0	10	1	19
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	9,34	7,68	10,98	11,91	9,68	9,97	9,11
		$\%Dev_{moy}$	4,67	4,9	7,85	8,51	5,35	5,83	5,08
	5	#Best	0	5	8	0	5	35	47
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	12,97	10,19	9,38	14,67	9,23	9,52	8,24
		$\%Dev_{moy}$	8,41	6,81	6,92	10,91	6,98	5,95	5,84
	7	#Best	0	0	0	0	1	46	53
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	15,47	12,5	14,79	19,19	12,25	11,83	10,95
		$\%Dev_{moy}$	11,56	9,45	10,01	14,37	9,41	7,2	7,27
1000	1	#Best	80	1	0	0	20	0	0
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	4,12	5,51	15,39	11,4	5,39	9,4	9,4
		$\%Dev_{moy}$	1,34	3,02	11,36	7,28	1,97	4,77	4,77
	2	#Best	75	17	0	0	2	2	5
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	6,98	6,52	9,1	11,28	7,11	8,71	7,56
		$\%Dev_{moy}$	3,51	4,37	7,23	7,84	4,85	5,57	4,59
	5	#Best	0	3	24	0	3	36	34
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	9,45	7,9	7,84	12,05	7,46	7,8	7,55
		$\%Dev_{moy}$	6,3	5,43	4,9	9,12	5,51	4,73	4,76
	7	#Best	0	0	1	0	1	50	48
		#LB	0	0	0	0	0	0	0
		$\%Dev_{max}$	11,65	9,72	9,51	14,01	9,35	8,36	8,34
		$\%Dev_{moy}$	9,21	7,51	7,56	11,8	7,51	5,6	5,63

TAB. A.6 – Temps d'exécution supérieurs aux temps de préparation, $s_i \in [10, 50], p_i \in [50, 100]$, $m = 5$.

n	$\|R\|$		HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1	HS-LPET	HS-LET
10	1	#Best	3	38	36	20	17	17	63	69
		#LB	0	24	23	11	10	10	41	45
		$\%Dev_{max}$	50	47,61	40,9	45,83	28,57	28,57	47,61	40,9
		$\%Dev_{moy}$	14,25	8,44	7,94	10,18	10,41	10,41	7,09	6,27
	2	#Best	0	56	73	23	25	25	35	43
		#LB	0	42	55	14	12	14	28	36
		$\%Dev_{max}$	43,47	21,73	21,73	21,73	27,27	27,27	45,71	60
		$\%Dev_{moy}$	10,21	3,37	3,42	6,12	5,87	6,04	10,19	9,87
	5	#Best	3	59	80	24	28	29	19	22
		#LB	0	48	67	17	19	19	18	22
		$\%Dev_{max}$	29,41	26,47	23,52	30,76	23,91	23,91	61,76	58,53
		$\%Dev_{moy}$	8,65	4,01	3,25	6,14	4,81	4,78	15,55	15,1
	7	#Best	1	62	85	23	21	21	9	16
		#LB	0	52	72	17	16	16	8	15
		$\%Dev_{max}$	26,82	18,75	18,75	21,73	20,83	20,83	47,82	57,8
		$\%Dev_{moy}$	8,09	2,55	2,02	5,31	4,74	4,71	16,04	15,21
50	1	#Best	0	0	0	0	0	0	56	56
		#LB	0	0	0	0	0	0	52	50
		$\%Dev_{max}$	13,4	23,01	20,63	19,84	17,46	17,46	12,69	13,17
		$\%Dev_{moy}$	6,25	7,92	9,37	6,32	6,94	6,94	1,39	1,5
	2	#Best	11	12	9	28	28	31	2	3
		#LB	0	1	2	0	0	1	1	0
		$\%Dev_{max}$	8,92	13,09	12,19	12,16	7,53	7,54	38,65	40,52
		$\%Dev_{moy}$	3,97	4,2	5,22	3,61	3,3	3,27	14,9	14,9
	5	#Best	0	9	3	3	60	62	0	0
		#LB	0	1	0	0	4	5	0	0
		$\%Dev_{max}$	24,21	22,61	29,16	21,27	16,47	15,26	75,6	69,04
		$\%Dev_{moy}$	10,55	8,73	9,8	9,07	3,71	3,67	38,21	38,53
	7	#Best	0	7	2	3	65	64	0	0
		#LB	0	1	0	0	1	4	0	0
		$\%Dev_{max}$	22,77	22,1	21,8	24,73	13,68	12,76	68,08	69,8
		$\%Dev_{moy}$	10,65	8,4	8,65	9,1	3,69	3,81	36,94	38,17
100	1	#Best	0	0	0	0	0	0	50	50
		#LB	0	0	0	0	0	0	43	41
		$\%Dev_{max}$	8,82	17,24	20,58	13,4	12,13	12,13	8,82	17,27
		$\%Dev_{moy}$	4,86	8,05	9,02	5,7	6,21	6,21	1,17	1,42
	2	#Best	43	2	0	11	25	25	1	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	7,43	9,59	10,69	8,9	7,78	7,78	34,05	34,15
		$\%Dev_{moy}$	3,01	4,47	5,86	3,75	3,31	3,35	16,83	16,67
	5	#Best	0	0	0	1	67	60	0	0
		#LB	0	0	0	1	0	0	0	0
		$\%Dev_{max}$	25,79	28,6	29,93	27,07	19,74	18,15	73,56	86,3
		$\%Dev_{moy}$	11,66	10,93	11,62	10,64	3,95	4,25	45,71	47,7
	7	#Best	0	1	0	0	61	52	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	22,25	24,03	25,51	23,44	12,16	16,32	91,09	81,8
		$\%Dev_{moy}$	11,8	10,58	11,35	10,73	3,82	4,17	46,41	46,9
500	1	#Best	0	0	0	0	0	0	30	7
		#LB	0	0	0	0	0	0	2	0
		$\%Dev_{max}$	6,05	11,65	11,96	6,84	7,94	7,94	4,12	3,62
		$\%Dev_{moy}$	4,15	8,2	8,64	5,43	6,23	6,23	0,81	1,09
	2	#Best	96	0	0	0	2	1	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	4,65	7,8	8,5	6,05	5,64	5,64	27,31	26,3
		$\%Dev_{moy}$	2,22	4,93	5,6	3,43	3,08	3,07	20,22	20,36
	5	#Best	0	0	0	0	66	45	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	18,07	18,67	18,67	16,8	8,12	7,29	74,53	71,51
		$\%Dev_{moy}$	11,65	12,13	12,58	11,66	2,74	2,74	57,77	58,15
	7	#Best	0	0	0	0	53	53	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,8	21,03	20,85	22,45	8,94	8,47	82,34	84,12
		$\%Dev_{moy}$	14,09	14,07	14,54	13,76	3,45	3,47	64,85	65,17
1000	1	#Best	0	0	0	0	0	0	29	2
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	5,19	10,26	10,39	6,59	7,2	7,2	1,91	2,87
		$\%Dev_{moy}$	3,98	8,18	8,45	5,29	5,97	5,97	0,71	1,05
	2	#Best	100	0	0	0	0	0	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	5,62	8,15	8,57	6,04	5,92	5,9	26,61	27,05
		$\%Dev_{moy}$	2,13	5,02	5,92	3,4	3,09	3,09	21,7	21,85
	5	#Best	0	0	0	0	60	53	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	15,12	16,64	17,8	15,64	6,84	6,9	77,44	77,08
		$\%Dev_{moy}$	10,9	11,5	11,75	11,18	3,05	3,05	60,62	61,16
	7	#Best	0	0	0	0	69	38	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	18,64	17,77	18,29	17,29	6,19	5,91	79,4	79,66
		$\%Dev_{moy}$	13,3	13,39	13,86	13,19	2,94	3,04	69,59	69,8

TAB. A.7 – Temps d'exécution inférieurs aux temps de préparation, $s_i \in [5, 10], p_i \in [1, 5]$, $m = 5$.

n	$\|R\|$		HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1	HS-LPET	HS-LET
10	1	#Best	1	26	52	22	14	14	33	77
		#LB	0	19	42	17	11	11	23	61
		$\%Dev_{max}$	46,47	38,38	34,32	38,38	38,8	38,8	38,38	38,8
		$\%Dev_{moy}$	5,64	3,18	3,72	3,4	4,12	4,12	2,54	2,71
	2	#Best	4	34	67	24	23	21	17	38
		#LB	1	19	49	13	11	11	12	30
		$\%Dev_{max}$	21,91	11,53	25,64	11,53	12,38	12,38	40,6	62,4
		$\%Dev_{moy}$	3,54	1,89	2,33	2,1	2,33	2,35	8,65	11,8
	5	#Best	0	36	80	32	27	29	7	13
		#LB	0	26	64	23	16	18	6	13
		$\%Dev_{max}$	31,05	23,6	21,76	23,6	14,28	14,28	49,45	59,62
		$\%Dev_{moy}$	3,8	2,64	2,15	2,56	1,37	1,32	14,51	18,31
	7	#Best	1	44	86	35	22	23	8	23
		#LB	0	33	69	27	15	15	8	23
		$\%Dev_{max}$	21,48	38,01	28,71	40,49	20,6	20,66	58,91	59,5
		$\%Dev_{moy}$	3,19	1,95	1,82	2,13	1,92	1,91	13,02	13,72
50	1	#Best	0	0	0	0	0	0	41	69
		#LB	0	0	0	0	0	0	27	67
		$\%Dev_{max}$	8,89	10,36	10,37	12,5	8,34	8,34	8,45	6,32
		$\%Dev_{moy}$	3,9	2,58	4,54	2,62	4,02	4,02	0,58	0,78
	2	#Best	2	46	8	36	13	15	1	1
		#LB	0	1	0	1	0	0	0	1
		$\%Dev_{max}$	22,6	10,38	9,6	14,56	7,94	8,42	35,32	45,55
		$\%Dev_{moy}$	3,11	1,8	3,05	1,84	2,11	2,15	15,47	18,04
	5	#Best	0	5	3	2	60	53	0	0
		#LB	0	0	2	0	1	0	0	0
		$\%Dev_{max}$	24,16	22,47	21,42	20,44	16,71	16,56	69,01	70,6
		$\%Dev_{moy}$	10,12	8,66	10,08	8,8	3,56	3,79	37,06	38,45
	7	#Best	0	6	0	6	71	56	0	0
		#LB	0	0	0	0	3	3	0	0
		$\%Dev_{max}$	25,65	21,02	21,73	22,35	17,55	18,24	61,8	83,4
		$\%Dev_{moy}$	8,82	7,5	8,34	7,72	3,16	3,3	35,48	37,6
100	1	#Best	0	0	0	0	0	0	36	60
		#LB	0	0	0	0	0	0	28	59
		$\%Dev_{max}$	8,97	7,15	8,64	6,13	9,38	9,38	4,27	4,5
		$\%Dev_{moy}$	3,66	2,63	4,47	2,4	4	4	0,47	0,5
	2	#Best	4	39	1	50	6	6	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	7,9	6,67	9,84	6,6	7,41	7,41	38,37	33,3
		$\%Dev_{moy}$	2,51	1,83	3,1	1,77	2,2	2,19	16,94	18,6
	5	#Best	0	1	0	0	64	50	0	0
		#LB	0	0	0	0	2	1	0	0
		$\%Dev_{max}$	27,37	23,21	26,61	23,63	14,9	19,18	76,29	74,2
		$\%Dev_{moy}$	11,47	10,27	11,43	10,35	3,18	3,6	43,1	46,35
	7	#Best	0	0	0	0	64	46	0	0
		#LB	0	0	0	0	0	1	0	0
		$\%Dev_{max}$	23,78	22,41	24,57	23	13,3	13,49	73,78	72,6
		$\%Dev_{moy}$	11,95	10,85	11,73	10,98	4,19	4,62	45,94	47,43
500	1	#Best	0	0	0	0	0	0	30	21
		#LB	0	0	0	0	0	0	15	12
		$\%Dev_{max}$	4,53	3,43	6,44	3,44	5,31	5,31	0,9	1,9
		$\%Dev_{moy}$	3,5	2,43	4,77	2,37	4,14	4,14	0,21	0,38
	2	#Best	0	45	0	61	0	1	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	4,91	4,4	5,8	4,44	5,08	5,07	25,62	27,53
		$\%Dev_{moy}$	2,22	1,58	3,07	1,55	2,11	2,11	19,75	21,55
	5	#Best	0	0	0	0	60	45	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	20,73	18,31	19,38	17,48	12,43	9,3	71,88	73,6
		$\%Dev_{moy}$	13,69	13,08	13,66	12,78	3,44	3,77	57,17	59,28
	7	#Best	0	0	0	0	62	38	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	20,07	19,56	21,37	19,7	8,5	10,12	80,42	81,53
		$\%Dev_{moy}$	15,48	14,17	15,58	14,26	4,13	4,31	64,18	65,9
1000	1	#Best	0	0	0	0	0	0	24	15
		#LB	0	0	0	0	0	0	1	3
		$\%Dev_{max}$	4,38	2,97	6,01	3	5,26	5,26	0,71	1,09
		$\%Dev_{moy}$	3,54	2,35	4,8	2,36	4,17	4,17	0,23	0,31
	2	#Best	0	47	0	54	0	0	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	4,7	4,38	5,76	4,24	5,06	5,06	26,52	28,5
		$\%Dev_{moy}$	2,14	1,54	2,98	1,53	2,1	2,11	20,54	22,31
	5	#Best	0	0	0	0	72	38	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,17	20,91	20,1	22,3	8,5	8,74	76,66	78,7
		$\%Dev_{moy}$	14,16	14,14	14,52	14,22	3,48	3,55	61,02	63,3
	7	#Best	0	0	0	0	60	42	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,07	20,67	21,35	21,09	11,84	12,83	85,63	86,8
		$\%Dev_{moy}$	15,32	14,65	15,57	14,76	4,1	4,28	69,83	71,6

TAB. A.8 – Temps d'exécution inférieurs aux temps de préparation, $s_i \in [10, 50], p_i \in [5, 10]$, $m = 5$.

n	$\|R\|$		HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1	HS-LPET	HS-LET
10	1	#Best	1	20	31	8	8	8	35	77
		#LB	0	11	22	2	4	4	18	46
		$\%Dev_{max}$	41,54	48,78	41,33	43,58	20,22	20,22	48,78	41,33
		$\%Dev_{moy}$	10,33	5,6	6,22	6,56	7,74	7,74	4,67	3,92
	2	#Best	0	29	56	15	26	24	24	53
		#LB	0	16	35	6	16	14	13	29
		$\%Dev_{max}$	18,85	49,56	34,27	18,07	23,24	23,24	49,56	45,14
		$\%Dev_{moy}$	8,14	4,01	3,36	4,98	4,26	4,26	10,18	9,01
	5	#Best	0	35	70	11	22	19	5	22
		#LB	0	21	46	5	11	10	4	18
		$\%Dev_{max}$	31,11	30,9	30,2	33,33	31,11	31,11	69,52	66,47
		$\%Dev_{moy}$	8,38	4,18	2,78	6,07	4,2	4,19	18,05	17,9
	7	#Best	0	39	72	16	18	19	9	20
		#LB	0	29	54	8	10	11	7	17
		$\%Dev_{max}$	29,91	28,71	25,87	34,54	18,96	12,74	72,52	51,9
		$\%Dev_{moy}$	6,82	2,81	2,73	4,26	3,32	3,18	14,81	13,79
50	1	#Best	0	0	0	0	0	0	33	65
		#LB	0	0	0	0	0	0	20	40
		$\%Dev_{max}$	18,12	24,54	22,83	20,06	14,78	14,78	24,81	15,7
		$\%Dev_{moy}$	7,01	7,87	9,66	6,67	7,9	7,9	2,11	1,72
	2	#Best	9	18	5	18	15	18	2	2
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	8,38	11,67	12,90	11,66	11,74	10,49	38,93	35,13
		$\%Dev_{moy}$	4,19	3,9	5,31	3,69	3,68	3,63	14,72	15,16
	5	#Best	0	2	2	2	58	53	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	28,83	31,45	31,62	26,73	23,87	23,92	69,12	85,14
		$\%Dev_{moy}$	11,69	10,6	11,15	11,06	4,03	4,19	40,46	40,9
	7	#Best	0	4	1	1	75	49	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,93	18,9	18,25	20,46	11,96	14,2	59,37	60,03
		$\%Dev_{moy}$	10,28	8,28	9,22	9,09	3,36	3,82	37,17	38,06
100	1	#Best	0	0	0	0	0	0	42	35
		#LB	0	0	0	0	0	0	13	25
		$\%Dev_{max}$	10,64	13,85	14,91	12,01	12,41	12,41	6,07	5,8
		$\%Dev_{moy}$	5,73	7,13	9,29	5,93	7,17	7,17	0,92	1,22
	2	#Best	17	8	0	35	15	18	0	1
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	8,28	7,75	14,75	7,84	9,05	9,28	28,57	27,51
		$\%Dev_{moy}$	3,75	4,15	6,22	3,43	3,7	3,64	18,32	18,14
	5	#Best	0	0	0	0	64	49	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	26,69	23,41	27,57	25,22	14,63	14,68	79,16	74,33
		$\%Dev_{moy}$	11,69	11,26	11,71	11,24	3,43	3,83	45,7	45,76
	7	#Best	0	0	0	0	68	45	0	0
		#LB	0	0	0	0	1	0	0	0
		$\%Dev_{max}$	24,27	21,26	21,86	21,18	11,2	11,96	67,34	65,36
		$\%Dev_{moy}$	12,41	11,22	12	11,28	3,88	4,23	46,15	46,8
500	1	#Best	0	0	0	0	0	0	29	12
		#LB	0	0	0	0	0	0	3	1
		$\%Dev_{max}$	6,51	9,45	11,22	7,61	9,66	9,66	2,76	2,54
		$\%Dev_{moy}$	4,87	7,03	9,03	5,47	6,85	6,85	0,62	0,87
	2	#Best	92	0	0	2	1	5	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	4,63	6,11	8,21	5,36	5,31	5,27	29,78	27,5
		$\%Dev_{moy}$	2,73	4,26	5,92	3,34	3,42	3,42	20,67	20,86
	5	#Best	0	0	0	0	59	44	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	20,94	21,29	20,58	19,83	9,84	11,22	69,68	73,36
		$\%Dev_{moy}$	13,61	14,09	14,39	13,46	3,07	3,22	57,91	58,35
	7	#Best	0	0	0	0	65	35	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	22,18	22,26	21,88	21,93	9,92	10,35	79,71	81,68
		$\%Dev_{moy}$	15,75	15,29	15,93	15,06	3,92	4,15	65,62	66,28
1000	1	#Best	0	0	0	0	0	0	23	12
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	5,76	8,63	10,97	6,64	8,15	8,15	1,66	2,49
		$\%Dev_{moy}$	4,88	7,22	9,14	5,5	6,8	6,8	0,62	0,84
	2	#Best	100	0	0	0	0	0	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	3,25	5,31	7,15	3,99	4,19	4,2	25,41	25,34
		$\%Dev_{moy}$	2,61	4,26	5,86	3,28	3,37	3,37	21,03	21,53
	5	#Best	0	0	0	0	48	55	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,05	21,57	22,17	21,62	8,9	13,75	74,24	74,62
		$\%Dev_{moy}$	14,59	15,32	15,75	14,83	3,17	3,13	60,8	61,27
	7	#Best	0	0	0	0	60	40	0	0
		#LB	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	19,65	19,94	20,43	19,39	9,31	9,45	79,73	81,77
		$\%Dev_{moy}$	14,82	14,74	15,44	14,54	3,12	3,22	69,48	69,92

TAB. A.9 – Temps d'exécution inférieurs aux temps de préparation, $s_i \in [50, 100], p_i \in [10, 50]$, $m = 5$.

n	$\|R\|$		HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1	HS-SPET	HS-LPET	HS-LET
10	1	#Best	0	37	34	8	10	10	2	48	48
		#LB	0	11	16	0	5	5	0	19	25
		$\%Dev_{max}$	65	31,66	40	33,47	40	40	65	38,7	58,33
		$\%Dev_{moy}$	27,94	11,5	13,62	18,76	20,3	20,3	26,4	10,08	12,08
	2	#Best	2	37	56	12	7	7	1	26	34
		#LB	0	15	31	5	4	4	0	13	23
		$\%Dev_{max}$	71,42	40,9	45,83	45,83	55	55	3,33	50	61,9
		$\%Dev_{moy}$	27,22	10,3	9,71	17,23	18,64	18,76	28,84	13,46	15,44
	5	#Best	0	36	71	7	16	17	0	8	11
		#LB	0	17	40	3	7	7	0	6	7
		$\%Dev_{max}$	58	46	26	54	54	54	88	74	62
		$\%Dev_{moy}$	22,04	8,07	5,9	15,44	13,9	13,69	33,85	18,74	18,23
	7	#Best	0	39	86	6	11	11	0	8	21
		#LB	0	26	49	3	4	4	0	6	17
		$\%Dev_{max}$	56,25	30	40,625	65,62	36,36	36,36	73,75	58,75	76,87
		$\%Dev_{moy}$	19,15	5,85	3,41	14,09	12,17	12,13	30,78	17,96	15,95
50	1	#Best	1	0	1	15	5	5	2	44	22
		#LB	0	0	0	0	0	0	0	8	4
		$\%Dev_{max}$	31,19	48,71	38,46	26,49	28,03	28,03	41,02	40,17	35,89
		$\%Dev_{moy}$	14,5	16,63	15,58	11,96	14,19	14,19	13,07	8,81	10,68
	2	#Best	8	21	12	19	20	19	0	3	4
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	28,22	30,64	36,29	32,25	29,03	23,38	56,45	53,22	42,85
		$\%Dev_{moy}$	11,31	10,51	12,21	9,97	10,09	9,71	24,74	19,44	22,07
	5	#Best	0	21	4	6	42	42	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	36,43	33,6	34,66	29,45	28	25,6	79,84	72,66	68,9
		$\%Dev_{moy}$	15,28	10,28	13,84	12,23	8,2	8,25	42,12	37,22	38,6
	7	#Best	0	21	6	5	49	48	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	23,48	24,67	28,03	26,58	15,54	15,67	65,6	71,21	64,39
		$\%Dev_{moy}$	13,31	8,92	10,96	11,26	6,83	6,95	39,85	36,14	37,6
100	1	#Best	1	0	0	7	2	2	8	50	12
		#LB	0	0	0	0	0	0	0	1	0
		$\%Dev_{max}$	23,85	27,63	28,8	17,54	21,1	21,1	24,77	22,66	26,22
		$\%Dev_{moy}$	12,41	16,6	15,57	10,92	12,76	12,76	10,4	7,94	10,81
	2	#Best	22	8	2	17	20	35	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	24,65	25,2	27,68	20,71	19,63	20,31	42,62	38,81	45,8
		$\%Dev_{moy}$	9,26	11,16	13,23	9,49	8,79	8,62	23,44	20,59	24,67
	5	#Best	0	5	0	0	57	47	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	29,48	24,3	32,66	27,4	20,7	20,35	76,89	74,08	83,26
		$\%Dev_{moy}$	15,3	12,37	16,39	13,12	7,54	7,75	50,14	46,96	48,84
	7	#Best	0	2	0	0	56	56	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	30,44	27,57	34,53	27,3	23,29	21,68	76,41	77,91	82,05
		$\%Dev_{moy}$	15,53	12,23	15,26	13,17	7,23	7,21	49,78	47,87	48,9
500	1	#Best	0	0	0	9	0	0	30	52	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	14,45	24,91	20,51	13,72	16,87	16,87	14,57	13,72	18,98
		$\%Dev_{moy}$	10,81	18,28	15,77	9,97	12,56	12,56	8,53	8,03	11,48
	2	#Best	56	0	0	15	20	12	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	18,58	22,36	24,4	19,68	22,28	20,94	43,17	42,67	41,53
		$\%Dev_{moy}$	7,78	12,05	13,72	8,67	8,44	8,37	25,8	25,38	29,06
	5	#Best	0	0	0	0	57	43	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	20,95	22,39	28,62	21,17	12,83	13,12	76,14	76,13	76,95
		$\%Dev_{moy}$	13,46	13,6	17,84	13,68	6,57	6,71	59,63	58,69	60,65
	7	#Best	0	0	0	0	68	34	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	23,48	23,34	28,6	22,91	13,08	12,26	79,31	80,32	81,52
		$\%Dev_{moy}$	16,9	16,27	20,33	16,28	7,12	7,4	66,85	66,12	67,9
1000	1	#Best	0	0	0	4	0	0	34	50	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	14,07	24,44	19,65	12,71	16,15	16,15	14,28	13,95	17,95
		$\%Dev_{moy}$	10,35	18,12	15,36	9,61	12,31	12,31	8,17	7,8	11,36
	2	#Best	67	0	0	11	8	15	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	15,13	19,11	20,27	17,21	15,86	15,51	38,92	38,42	40,46
		$\%Dev_{moy}$	7,36	11,69	13,12	8,26	8,05	7,95	26,27	26,06	29,5
	5	#Best	0	0	0	0	50	55	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	19,89	20,93	24,46	20,74	12,85	12,85	76,78	76,38	76,02
		$\%Dev_{moy}$	13,54	13,74	18,14	14,08	7,21	7,15	63,92	63,69	65,5
	7	#Best	0	0	0	0	59	45	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,33	21,69	27,43	20,68	12,47	12,77	85,77	84,3	87,4
		$\%Dev_{moy}$	16,44	15,93	20,41	16,01	7	7,14	71,88	71,75	73,07

TAB. A.10 – Les temps d'exécution et de préparation prennent leurs valeurs dans l'intervalle $[1, 10]$, $m = 5$.

n	$\|R\|$		HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1	HS-SPET	HS-LPET	HS-LET
10	1	#Best	1	25	21	10	2	2	3	42	46
		#LB	0	7	12	0	1	1	0	14	22
		$\%Dev_{max}$	51,81	44,11	52,58	57,01	49,6	49,6	61,73	44,11	45,08
		$\%Dev_{moy}$	23,74	12,72	15,08	16,57	19,97	19,97	23,63	10,6	12,24
	2	#Best	1	34	48	13	15	14	0	17	33
		#LB	0	18	27	4	3	4	0	7	22
		$\%Dev_{max}$	57,52	38,52	40	53,96	35,71	35,71	65,27	50,34	53,7
		$\%Dev_{moy}$	19,05	8,62	9,24	13,34	12,51	12,65	25,15	14,59	13,8
	5	#Best	0	30	62	9	10	11	0	9	12
		#LB	0	18	42	4	5	6	0	7	9
		$\%Dev_{max}$	44,62	33,91	28,8	42,69	37,5	37,5	79,87	67,07	64,03
		$\%Dev_{moy}$	17,26	6,64	5,55	11,28	10,11	10,14	29,24	18,71	18,51
	7	#Best	0	41	83	6	13	13	0	9	17
		#LB	0	21	55	3	6	6	0	8	15
		$\%Dev_{max}$	31,6	34,83	21,21	30,3	24,39	24,39	62,26	54,24	50
		$\%Dev_{moy}$	14,86	4,41	2,85	9,27	9,24	9,17	24,46	15,41	14,9
50	1	#Best	0	0	0	2	1	1	1	33	23
		#LB	0	0	0	0	0	0	0	2	4
		$\%Dev_{max}$	28,95	33,38	33	29,59	29,34	29,34	26,61	23,9	26,39
		$\%Dev_{moy}$	16,16	18,91	17,99	14,1	15,66	15,66	11,8	7,84	10,22
	2	#Best	3	8	3	17	20	32	1	7	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,87	25,64	34,33	21,71	20,71	21,4	43,7	39,41	53,09
		$\%Dev_{moy}$	11,86	11,6	14,02	10,5	9,8	9,57	23,43	19,7	21,18
	5	#Best	0	10	0	2	47	49	0	0	0
		#LB	0	0	0	0	0	1	0	0	0
		$\%Dev_{max}$	41,28	36,67	35,36	36,02	27,77	31,5	86,21	79,96	71,67
		$\%Dev_{moy}$	16,32	12,1	14,69	13,33	7,46	7,76	44,4	40,78	40,73
	7	#Best	1	10	5	4	50	48	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	25,57	25,3	28,1	24,62	16,77	21,29	74,06	70,2	70,8
		$\%Dev_{moy}$	12,28	8,68	10,22	9,84	5,45	5,47	39,06	35,55	35,4
100	1	#Best	0	0	0	2	0	0	4	46	18
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	24,81	30,26	32,75	21,37	26,03	26,03	25,88	24,63	29,64
		$\%Dev_{moy}$	14,81	18,85	16,9	13,32	15,6	15,6	9,3	7,56	9,2
	2	#Best	7	4	1	18	34	27	1	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	17,47	21,8	22,91	21,34	15,88	15,5	36,29	36,41	39,03
		$\%Dev_{moy}$	9,68	11,1	12,42	9,48	8,43	8,54	23,13	21,11	22,58
	5	#Best	0	0	0	1	53	48	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	29	31,11	30,36	30,27	19,03	22,09	73,27	76,33	74,98
		$\%Dev_{moy}$	16,02	14,21	16,74	14,12	7,09	7,42	49,7	47,78	48,6
	7	#Best	0	0	0	0	60	42	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	26,49	24,54	26,88	25,06	15,75	16,47	73,07	68,91	71,72
		$\%Dev_{moy}$	14,31	12,24	14,48	12,09	5,65	6,12	48,03	46,39	46,9
500	1	#Best	0	0	0	0	0	0	18	47	2
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	17,49	25	21,29	16,77	19,26	19,26	13,06	11,58	15,52
		$\%Dev_{moy}$	13,33	19,7	16,96	12,31	14,86	14,86	7,52	7,07	9,08
	2	#Best	34	0	0	14	25	28	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	12,95	15,83	17,75	12,72	10,84	11,13	31,57	32,06	32,45
		$\%Dev_{moy}$	8,52	11,96	12,92	8,93	8,36	8,33	23,7	23,46	25,36
	5	#Best	0	0	0	0	59	41	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	26,09	25,42	29,9	26,22	14,5	15,46	79,61	81,34	78,31
		$\%Dev_{moy}$	15,27	15,28	18,09	15,01	6,09	6,16	59,39	58,63	59,6
	7	#Best	0	0	0	0	54	46	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	24,21	25,65	29,35	23,57	13,21	14,26	85,96	85,42	84,3
		$\%Dev_{moy}$	17,27	16,94	19,86	16,31	6,16	6,3	66,71	65,85	66,8
1000	1	#Best	0	0	0	0	0	0	29	56	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	16,62	25,53	20,73	15,31	19,55	19,55	12,7	11,61	14,84
		$\%Dev_{moy}$	13,03	19,7	16,81	12,12	14,6	14,68	7,02	6,78	9,12
	2	#Best	41	0	0	15	21	24	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	10,93	15,43	16,03	11,7	11,5	11,32	31,15	31,67	31,5
		$\%Dev_{moy}$	8,3	12,11	12,8	8,65	8,37	8,36	24,63	24,43	26,05
	5	#Best	0	0	0	0	55	46	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	20,01	22,08	23,22	20,22	10,9	11,16	74,28	73,57	72,8
		$\%Dev_{moy}$	14,6	14,76	17,75	14,75	6,25	6,32	62,5	62,22	63,6
	7	#Best	0	0	0	0	53	48	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	22,4	22,32	24,92	20,73	11,16	10,91	80,9	80,32	83,21
		$\%Dev_{moy}$	16,6	16,55	19,37	16,05	5,91	5,9	71,22	71,05	71,6

TAB. A.11 – Les temps d'exécution et de préparation prennent leurs valeurs dans l'intervalle $[10, 50]$, $m = 5$.

n	$\|R\|$		HP-SET	HP-LPET	HP-LET	HP-LPT	HP-SCN	HP-SCN1	HS-SPET	HS-LPET	HS-LET
10	1	#Best	1	13	18	4	3	3	1	39	54
		#LB	0	5	13	0	1	1	0	12	27
		$\%Dev_{max}$	57,09	50,68	41,29	64,72	34,22	34,22	53,42	50,68	45,87
		$\%Dev_{moy}$	19,05	13,37	13,72	13,95	15,93	15,93	14,39	8,32	8,21
	2	#Best	0	20	35	11	17	20	0	24	34
		#LB	0	11	16	4	5	6	0	10	17
		$\%Dev_{max}$	43,7	43,53	42,27	30,25	26,18	26,18	51,41	43,53	43,7
		$\%Dev_{moy}$	13,22	7,93	7,93	9,23	7,98	7,79	16,9	10,68	10,9
	5	#Best	1	36	61	12	16	17	0	12	25
		#LB	0	21	34	6	6	6	0	8	18
		$\%Dev_{max}$	42,81	33,83	26,34	30,53	31,21	31,21	59,92	53,24	45,54
		$\%Dev_{moy}$	9,36	4,14	3,43	5,9	4,91	4,93	19,17	14,35	14,48
	7	#Best	0	40	77	17	13	12	0	8	24
		#LB	0	28	59	13	6	5	0	7	22
		$\%Dev_{max}$	20,79	19,06	19,06	23,52	19,87	19,87	48,73	44,77	43,7
		$\%Dev_{moy}$	7,09	2,83	2,57	4,72	4,12	4,1	18,11	13,75	12,26
50	1	#Best	0	0	0	0	1	1	6	30	29
		#LB	0	0	0	0	0	0	0	1	5
		$\%Dev_{max}$	31,07	42,12	37,58	31,27	32,66	32,66	26,24	24,45	21,31
		$\%Dev_{moy}$	17,44	21,36	22,87	19,13	17,19	17,19	8,25	6,37	7,04
	2	#Best	1	8	2	8	26	30	2	4	5
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	27,8	24,69	27,54	28,32	16,14	15,72	37,39	34,86	41,97
		$\%Dev_{moy}$	11,4	11,61	13,24	10,91	8,96	8,96	19,11	17,08	17,65
	5	#Best	0	1	2	1	56	54	0	0	0
		#LB	0	0	0	0	0	2	0	0	0
		$\%Dev_{max}$	26,2	26,49	27,48	26,66	21,42	19,9	80,29	78,92	75,29
		$\%Dev_{moy}$	12,49	10,54	11,66	10,68	4,39	4,46	38,92	36,91	37,35
	7	#Best	0	0	2	1	65	55	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	25	22,67	20,31	21,26	14,01	14,54	65,47	72,56	69,08
		$\%Dev_{moy}$	11,87	9,89	10,31	10,13	4,16	4,35	39,61	38,9	38,53
100	1	#Best	1	0	0	0	0	0	8	23	11
		#LB	0	0	0	0	0	0	0	0	1
		$\%Dev_{max}$	23,87	35,9	33,5	27,95	24,76	24,76	23,25	21,25	23,37
		$\%Dev_{moy}$	15,25	21,12	21,14	18,67	16,29	16,29	7,4	6,51	6,8
	2	#Best	14	5	0	2	37	30	2	1	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,66	29,9	29,8	24,57	15,9	15,26	41,69	37,18	43,8
		$\%Dev_{moy}$	10,29	12,66	14,04	11,59	8,65	8,63	21,15	19,95	20,41
	5	#Best	1	0	0	0	57	45	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	34,6	30,6	35,16	33,17	19,58	19,67	77,81	75,18	77,55
		$\%Dev_{moy}$	15,12	14,03	15,14	14,44	5,07	5,13	48,08	47,27	47,6
	7	#Best	0	0	0	0	68	39	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	29,92	26,38	30,93	25,42	15,68	14,63	78,02	76,21	72,38
		$\%Dev_{moy}$	13,98	12,83	14,02	12,75	4,31	4,7	48,27	47,62	47,32
500	1	#Best	0	0	0	0	0	0	14	22	6
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	18,09	26,92	26,6	24,29	21,28	21,28	10,74	10,96	12,11
		$\%Dev_{moy}$	14,5	22,08	18,67	17,55	15,74	15,74	6,23	6,03	6,76
	2	#Best	9	0	0	1	46	44	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	12,72	17,03	18,12	14,9	11,81	11,53	28,11	29,78	29,43
		$\%Dev_{moy}$	9,1	12,76	13,18	11,03	8,05	8,04	22,72	22,77	23,18
	5	#Best	0	0	0	0	52	48	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,66	24,18	23,01	23,11	10,86	11,6	78,52	73,14	78,45
		$\%Dev_{moy}$	14,81	14,98	15,95	14,7	3,91	3,96	58,19	57,48	57,71
	7	#Best	0	0	0	0	61	39	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	23,53	22,55	23,96	24,96	9,85	9,75	84,28	80,43	80,28
		$\%Dev_{moy}$	16,86	16,47	17,6	16,3	4,48	4,62	65,9	65,72	65,79
1000	1	#Best	0	0	0	0	0	0	23	24	9
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	17,19	26,3	22,26	21,41	18,77	18,77	9,07	9,33	10,63
		$\%Dev_{moy}$	14,54	22,34	18,27	17,66	15,84	15,84	6,19	6,09	6,76
	2	#Best	5	0	0	0	52	43	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	11,46	15,22	17,57	13,42	10,28	10,51	27,91	27,8	28,29
		$\%Dev_{moy}$	9,12	12,9	13,05	10,93	7,9	8	23,52	23,58	24,32
	5	#Best	0	0	0	0	54	46	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	22,01	20,9	21,03	21,3	8,55	8,52	74,46	73,62	73,72
		$\%Dev_{moy}$	14,52	14,57	15,81	14,39	3,92	3,95	61,01	60,92	61,4
	7	#Best	0	0	0	0	65	35	0	0	0
		#LB	0	0	0	0	0	0	0	0	0
		$\%Dev_{max}$	21,83	19,26	21,01	20,16	6,3	7,4	69,84	69,95	70,67
		$\%Dev_{moy}$	15,56	15,43	16,45	15,2	3,63	3,76	59,51	59,5	69,9

TAB. A.12 – Les temps d'exécution et de préparation prennent leurs valeurs dans l'intervalle $[50, 100]$, $m = 5$.