UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté des Sciences

Département de Mathématiques

MEMOIRE DE MAGISTER

En Mathématiques

Spécialité : Recherche Opérationnelle

LE FLOWSHOP HYBRIDE A 2-ETAGES

Par

SAKRI Redha

Devant le jury composé de :

Hannane Farouk	Professeur, U.S.D., Blida	Président
Boudhar Mourad	Maître de conférences, U.S.T.H.B, Alger	Examinateur
Tami Omar	Chargé de cours, U.S.D., Blida	Examinateur
Derbala Ali	Maître de conférences, U.S.D., Blida	Promoteur

Blida, Octobre 2008

RÉSUMÉ

Dans un environnement informatique composé d'un « serveur » et de « M » machines parallèles identiques appelées processeurs, « N » tâches sont à exécuter sans interruption. Avant son exécution, une tâche i (i = *1,..., n)* doit être chargée sur une des machines parallèles. Ce chargement ou *setup* est réalisé par une autre machine particulière appelée serveur. Ayant terminé un chargement, le serveur est libre d'en effectuer un autre et le chargement d'une tâche doit être suivi de son exécution. Ce problème est appelé un flowshop hybride à 2-étages et est noté $P2,S1|p_i,s_i|C_{\max}$. Dans la littérature, il est prouvé NP-difficile. Dans le mémoire, un schéma rapide d'organisation et de fonctionnement d'atelier est donné. Une brève introduction, dans laquelle nous rappelons les notions de base et en présentant les problèmes d'ordonnancement est faite. Deux exemples de flowshop hybrides sont fournis. Une recherche bibliographique est menée. Nous présentons l'algorithme de Guirchoun et Martineau (2003) et deux heuristiques de Abdekhodaee et al. (2006) pour sa résolution. Des exemples illustratifs de ces algorithmes sont déroulés. Une nouvelle heuristique est proposée par nos soins. Elle s'inspire et s'influence des idées de Johnson (1954). Un exemple de son déroulement est même fourni, et une comparaison avec l'heuristique Forward de Abdekhodaee et al. (2006) est effectuée. Notre heuristique s'avère la plus efficace en fournissant des valeurs de la longueur d'ordonnancement les plus proches de l'optimum. On propose la résolution du problème étudié en comparant six algorithmes de type liste, d'une heuristique SR1 et de deux variantes AG1 et AG2 d'un algorithme génétique. Les implémentations réalisées par nos soins sont regroupées dans un logiciel pédagogique à interface conviviale.

***Mots-clés** : flow shop hybride,Makespan, machine parallèle avec serveur,*

ABSTRACT

In a computer system composed of a "server" and "M" identical parallel machines called processors," N" jobs are to be executed without interruption.

Before its execution, a job i (i = 1,..., n) must be charged on one of the parallel machines.

This loading or setup is achieved by another particular machine called server. Having finished a loading, the server is free to do another of it and the loading of a job must immediately be followed of its execution. This problem is called a 2-stage hybrid flowshop noted $P2,S1|p_i,s_i|C_{\max}$. The problem proves to be NP-difficult in the literatures. Given is a fast diagram of organization and workshop performance. Also given is a succinct, introduction, in which we remind of some basics notion and where workshop scheduling problems are highlighted. Two hybrid flowshop examples are provided. A bibliographic research is take. Tow algorithm of Guirchoun and Martineau (2005); and two heuristic of Abdekhodaee and al. (2006) are given for its resolution. Illustrative examples of these algorithms are run. We have also provided a new heuristic. It is inspired and influence by the ideas of Johnson (1954) algorithm. An example of its running is also supplied; and a comparison with the heuristic Forward of Abdekhodaee and al. (2006) is done. Our heuristic proves has proven to be the most the most efficient while providing values closer to optimum Makespan. We suggest resolving the problem by comparing six algorithms of type lists, a heuristic SR1 and two variants of a genetic algorithm AG1 and AG2.The implementations achieved are gathered in a pedagogic software with convivial interface.

***KEYWORD**: hybrid flowshop, makespan, parallel machine with server.*

ملخص

في نظام معلوماتي مكون من حاسوب رئيسي "M" حاسوب مرتب بشكل متوازي و المسمى حاسوب ثانوي , ننجز "N"عملية بدون توقف.قبل إنجاز عملية i (i = 1,..., n) نحملها على إحدى الحواسب الثانوية عملية التحميل تنجز من طرف الحاسوب الرئيسي.عند إنتهاء عملية التحميل يكون الحاسوب الرئيسي حر في إنجاز عملية أخري.

هذا المشكل يسمى فلوشوب مركب بطابقين. ويرمز له بالرمز $P2,S1|p_i,s_i|C_{\max}$. وقد برهن في المراجع العلمية أن هذا الإشكال جد صعب. في نظرة سريعة نعطي لمحة حول تنظيم و سير ورشة صناعية. و في مقدمة وجيزة نذكر بالمبادئ الأساسية و نقدم بعض الإشكاليات المطروحة في تنظيم الورشات الصناعية. كما نقدم تطبيقين صناعيين لفلوشوب المركب. بالإضافة على بحث بيبليوغرافي حول الموضوع. كما نعرض خوارزمية قيرشون و مارتينو (2005) و خوارزميتان لـويرث مع تطبيقهما على مثالين توضيحين.كما نقترح خوارزمية جديدة مستوحاة من خوارزمية جونسون.و نبين تطبيقها على مثال توضيحي. كما نقارن النتائج مع الحلول المنجزة بخوارزمية ويرث و في هاته المقارنة لاحظنا أن الخوارزمية المقترحة تعطي نتائج قريبة من الحل الأمثل. كما نقترح كحل للمشكل مقارنة ستة خوارزميات من نوع القائمة مع خوارزمية SR1 و نوعين من الخوارزمية الجينية َAG1 و AG2 . كل الخوارزميات المعروضة قد برمجت و جمعت في برنامج معلوماتي.

كلمات المفتاح: فلوشوب مركب ، طول الترتيب , ألآت متوازية مع ألة رئيسية.

REMERCIEMENTS

Je remercie tout d'abord, notre vénéré Allah,
Le tout puissant, à qui nous devons le tout.

Je tiens à exprimer ma gratitude à mon responsable de recherche Derbala Ali**,** maître de conférences à l'Université Saad Dahlab de Blida, pour avoir diriger les travaux de recherche. Je le remercie également pour ses précieux conseils et particulièrement pour sa disponibilité.

Je remercie Monsieur Hannane Farouk, Professeur à l'Université Saad Dahlab de Blida, pour avoir accepté d'examiner ce travail et qui m'a fait l'honneur de présider ce jury. Qu'il trouve ici l'expression de ma profonde reconnaissance.

J'adresse mes sincères remerciements à Monsieur Boudhar Mourad, Maître de conférences à l'Université Houari Boumediene d'Alger et Monsieur Tami Omar, chargé de cours à l'Université Saad Dahlab de Blida d'avoir accepté d'être membre de jury et de m'avoir fait l'honneur d'examiner mon travail.

Je ne saurais oublier de remercier mes chers parents, qui étaient toujours à mes cotés et m'avaient tant aidé et soutenu. Qu'ils trouvent ici l'expression de ma sincère gratitude et ma profonde reconnaissance.

Je remercie enfin mes chers amis Boualem, Kamel, Mohamed et tous mes camarades post-graduants du département de mathématiques pour leur bonne humeur et leur sympathie.

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION

En Algérie, l'image du processus de production que l'on a à l'esprit est le fait productif qui se limite à l'installation d'une usine donnée et à sa gestion ultérieure. On admet l'usine comme l'unique et la plus significative expression de la technique choisie et on identifie l'usine à la combinaison capital et travail. On introduit les notions de "flexibilité" dans les ateliers manufacturiers. La technologie est définie comme l'agent pour l'application de la science. Une entreprise est un lieu de transformation de matières premières en produits finis, capable de modifier continuellement sa propre structure, de façon à l'adapter au contexte dans lequel elle doit vivre et s'accroître. Le produit est une "sortie ou output" économiquement optimal.

L'ordinateur participe à une course infernale pour les sciences de l'informatique, à une cohabitation et à une participation dans toute la vie, sociale, économique et industrielle. Quels sont les liens existant entre la recherche, le développement technologique et l'évolution du système économique ? Peut-on esquisser un profil des entreprises et des activités d'avenir? Dans la suite, un concept clé émerge et est explicité : la flexibilité dans les ateliers manufacturiers.

En période de saturation de la demande, les entreprises se trouvent de plus en plus coincées entre deux contraintes : pour préserver ou accroître leur part du marché, elles cherchent à personnaliser les produits, à les adapter aux goûts ou aux contraintes techniques exactes de leurs clients ; or cet objectif commercial est contradictoire avec les principes généraux de la production qui veulent que l'on privilégie les séries longues. Le conflit permanent entre le directeur commercial et

celui de la production sera résolu grâce à l'introduction d'une organisation de type atelier flexible. Du jour au lendemain, on pourra changer la forme, la couleur voir la matière des objets pour satisfaire tel ou tel client. Ce système permet de minimiser les dangers que représentent les faillites fréquentes des petites entreprises clientes ; la souplesse de l'équipement rend les capacités de production immédiatement disponibles pour d'autres marchés.

Nous nous trouvons au début d'une phase de modifications et d'acquisitions technologiques de caractère radical, de nature à altérer les caractéristiques du processus de production.

A la lumière de l'introduction massive de nouvelles technologies, les classifications des divers types de production sont de type "petit" et "flexible". L'esquisse d'un nouveau type de processus de production conduit au concept de "flexibilité", "d'adaptation", où les décideurs n'ont pas une connaissance certaine de l'avenir, car ils sont en face d'une multiplicité d'évolutions possibles.

Une capacité de procéder à des innovations continuelles dans les caractéristiques des produits, de réussir à adapter les systèmes d'organisation, de gestion et les politiques commerciales aux mutations de l'environnement y est sous entendu.

L'introduction d'un nouveau type de processus de production, la transition d'une ancienne à une nouvelle manière de produire est faite. L'innovation est la conception d'une technologie économiquement productive. La tertiarisation du processus de production et son intégration croissante dans l'environnement réduisent de plus en plus l'importance de la phase de ce processus qui s'identifie avec l'usine, les équipements physiques qui sont l'expression de cette phase. Elle est une extension des activités industrielles.

Les petites et moyennes entreprises, dans leur ensemble et dans leur état actuel, ont peu de perspectives d'augmenter leur productivité, c'est à dire leur compétitivité dans les délais. Sauf à prendre un raccourci possible et récent qui métamorphoserait l'entreprise d'une manière globale, et pourrait la faire décoller d'un bloc, pour entrer tout droit dans l'univers du futur : la productique intégrée. Elle recouvre et fait bénéficier l'entreprise, même petite, de tout ce qu'apporte l'informatisation, de la conception d'un produit jusqu'à sa vente, en intégrant aussi bien la fabrication des différentes pièces, la facturation, la gestion du personnel, le stockage, le transport, etc. dans le processus informatique.

Tout commence avec la commande numérique au milieu des années soixante [01], dans la mécanique, plus précisément dans la machine-outil qui a joué le détonateur pour l'explosion de techniques nouvelles. Perceuses, fraiseuses, aléseuses, rectifieuses, les machines outils créent les pièces de notre univers industriel. A partir de blocs d'aciers ou débauches de fonte, d'aluminium, de titane, de cuivre…elles donnent forme aux écrous, aux vis, aux blocs-moteurs, aux arbres de transmission, engrenages, soupapes, carcasses de moteurs électriques.

La machine-outil, productrice des choses du monde contemporain, mère de tous les biens industriels, a été un pôle d'innovation. Elle a su réussir le mariage de la mécanique et de l'informatique. Elle s'est donnée un cerveau, non pas pour faire des calculs, mais pour fabriquer des pièces.

L'atelier est dit "flexible" parce que, sans opérateurs, cet ensemble de centres d'usinage et de chariots transporteurs intelligents possèdent la souplesse, la " flexibilité" d'un artisan capable de passer d'une activité à une autre, de fabriquer une pièce unique aussi bien qu'une série de pièces et de s'assurer, sans moyens humains, que son travail est bien fait. Son but est de produire sans homme. On confie à l'informatique l'exécution de tâches banales.

Dans l'entreprise intégrée, informations, ordres et communication sont véhiculés par l'informatique en flot continu. De la conception du produit à son exécution par l'atelier et à son expédition, la mémoire de l'ordinateur se charge de remplacer plans et papiers. Toutefois l'homme garde la main et reste le pilote du réseau informatisé.

Une forme réduite de l'atelier flexible est la " cellule flexible" qui comporte une ou deux machines à commande numérique équipées d'un transporteur de pièces et d'un ou deux robots de chargements. Ces cellules autonomes travaillent 24 heures sur 24 sans opérateur (avec une surveillance intermittente, toutefois, et un réapprovisionnement en matière première).

La productique apporte à l'usine du futur l'abandon définitif de l'organigramme en forme de pyramide : les qualifications des travailleurs sont peu différentes les unes des autres, leurs salaires très voisins et les tâches répétitives et harassantes disparues.

On explore dans ce mémoire les notions de "flexibilité" dans un centre informatique. Notre but est d'accumuler une connaissance pédagogique, scientifique, du savoir-faire d'une manière efficace et d'établir des implémentations d'algorithmes. Dans un environnement informatique composé d'un « serveur » et de « M » machines

parallèles identiques appelées processeurs, « N » tâches sont à exécuter sans interruption. Avant son exécution, une tâche i (i = 1,…, n) doit être chargée sur une des machines parallèles. Ce chargement ou *setup* est réalisé par une autre machine particulière appelée serveur. Ayant terminé un chargement, le serveur est libre d'en effectuer un autre et le chargement d'une tâche doit être suivi de son exécution. Ce problème est appelé un flowshop hybride à 2-étages et est noté $P2,S1|p_i,s_i|C_{\max}$. Dans la littérature, il est prouvé NP-difficile [02].

Le mémoire est organisé en cinq chapitres. Dans le premier chapitre, nous présentons les définitions et les concepts principaux liés à l'ordonnancement. Un schéma rapide d'organisation et de fonctionnement d'atelier est donné. Une brève introduction est faite, dans laquelle nous rappelons les notions de base et en présentant les problèmes d'ordonnancement d'atelier. Une classification et notation des problèmes d'ordonnancement sont exposés. Deux exemples de flowshop hybrides modélisant un bloc opératoire d'un hôpital et une industrie du verre sont fournis. Une recherche bibliographique est insérée à la fin de ce chapitre.

Dans le chapitre deux, nous présentons l'algorithme de Guirchoun et Martineau [03] pour la résolution du flowshop hybride à 2-étages. Un exemple illustratif est même fourni. Une implémentation par nos soins est réalisée.

Dans le chapitre suivant, deux heuristiques de Abdekhodaee et al.[02] pour la résolution du problème d'un seul serveur à machines parallèles sont présentées en détail. Leurs implémentations sont exposées.

Dans le chapitre quatre, une nouvelle heuristique pour la résolution du problème où une seule machine au premier étage existe appelée serveur et deux machines sont disponibles au second étage est proposée. Elle s'inspire et s'influence des idées de Johnson [04]. Elle sera présentée, un exemple de son déroulement est fourni, une implémentation par nos soins est réalisée et une comparaison avec l'heuristique Forward de Abdekhodaee et al.[02] est effectuée. Notre heuristique s'avère la plus efficace en fournissant des valeurs de la longueur d'ordonnancement les plus proches de l'optimum. Une borne inférieure fournit par Abdekhodaee est utilisée. Malheureusement cette borne n'a pas été dépassée est reste toujours d'actualité.

Dans le chapitre cinq, une étude comparative d'algorithmes de type liste, de l'heuristique SR1 et deux variantes d'un algorithme génétique est faite. On a réalisé de nombreuses expérimentations sur des données associées aux tâches générées

aléatoirement. Les différentes méthodes présentées ont été comparées et l'influence des paramètres de ces méthodes a été mesurée. Les heuristiques fournissent des ordonnancements réalisables, proches des valeurs des bornes inférieures fournies par la littérature. On peut améliorer ces valeurs des heuristiques, en lançant et en déroulant la version AG2.Une remarque est à faire. Le temps de calcul nécessaire peut être assez important, surtout pour des applications à grand nombre de tâches. Nous avons confirmé que l'heuristique SR1, proposée par nos soins, et la méthode SPT associée à une règle d'affectation FAM, sont les mieux adaptées au calcul rapide d'un ordonnancement réalisable.

Les implémentations réalisées sont regroupées dans un logiciel pédagogique à interface convivial. Il représentera sûrement un outil à exploiter et à mettre à la disposition des étudiants de notre département.

CHAPITRE 1
NOTIONS DU FLOWSHOP HYBRIDE

Nous souhaitons que le présent mémoire puisse représenter un guide et être facilement lu par tout individu qui porte un intérêt à l'optimisation combinatoire ou de la recherche opérationnelle. Nous avons estimé nécessaire de présenter en premier les définitions et les concepts principaux liés à l'ordonnancement. Une brève introduction est faite et dans laquelle nous rappelons les notions de base et en présentant les problèmes d'ordonnancement d'atelier. Une recherche bibliographique est insérée a la fin de ce chapitre. Pour plus de détails, il existe de nombreux ouvrages et références récents en ordonnancement, pédagogiques et de recherche [05], [06], [07], [08] etc.

1. L'atelier –Schéma rapide d'organisation et de fonctionnement

Des notions et définitions de termes usuels que nous utiliserons pour la description des problèmes d'ordonnancement d'ateliers de production sont données. Nous reprenons essentiellement celles de Duvivier [09].

Un atelier est formé de machines reliées par des moyens de transport (de façon analogue, un ordinateur est constitué de composants : mémoires, processeur,…reliés par des connexions). Il est destiné à fabriquer des pièces (dont le modèle théorique est appelé job), elles mêmes se décomposant en diverses tâches, et ce travail nécessite des ressources (moyens financiers, matière première, moyens de transport, …).

1.1. Tâches et travaux

Un *travail* est défini comme une activité de l'homme appliquée à la production, à la création, à l'entretien de quelque chose. Le travail est un attribut humain privilégié, il est le propre de l'homme. Il est aussi un acte parce qu'il informe, crée et projette l'homme hors de lui, dans un mouvement dont l'acte final est le reflet de sa rationalité dans l'objet crée, selon le philosophe Hegel. Le travail est une problématique d'ajustement entre matières et outils. Il existe une certaine forme de "travail" animal

chez l'abeille, la fourmi, le termite, etc…et même une forme de coopération et d'usage d'outil" chez des singes.

Une *opération* appelée aussi une tâche est un travail élémentaire à être localisé dans le temps par des dates de début d'exécution ou de fin d'exécution dont la réalisation nécessite un certain intervalle de temps appelé *durée d'exécution.*

Un *produit* est une succession de plusieurs opérations élémentaires appliquées à une matière première ou à un produit intermédiaire. Dans le cas général, ces opérations sont partiellement ordonnées dans le temps selon un graphe de précédence appelé *gamme de fabrication*.

1.2. Machines

Une *machine* est un appareil ou un ensemble d'appareils capables d'effectuer un certain travail ou de remplir une certaine fonction, soit sous la conduite d'un opérateur, soit d'une manière autonome. Elle est conçue par des ingénieurs dont le maître à penser est l'ordre technico-économique et non l'humain.

Dans le cadre des problèmes d'ordonnancement d'atelier, une machine peut représenter aussi une ressource utilisée pour mener à bien un projet.

Dans un atelier, un *étage* représente un ensemble de machines disposées en parallèle, chacune d'entre elles peut réaliser, éventuellement avec des performances différentes, une même opération sur un produit donné.

1.3. Ressources

Une *ressource* est un moyen financier, technique ou humain destiné à être utilisée pour la réalisation d'une opération. Elle est disponible en quantité limitée ou illimitée. Elle est aussi définie comme une personne capable de fournir des solutions à quelque chose. Elle est caractérisée comme un élément de la puissance, des moyens dont on dispose ou des possibilités d'action.

Une ressource est dite *supplémentaire* si elle représente un moyen technique ou humain, critique ou indispensable à la réalisation du projet, à l'exception des machines.

Elle peut être renouvelable ou non renouvelable. Une ressource est *renouvelable* si après avoir été allouée à une ou plusieurs tâches, elle redevient disponible en même quantité après la fin d'exécution de ces tâches.

1.4. Fonctions objectifs

La durée d'exécution d'un ensemble de tâches sur des machines est appelée la *longueur d'ordonnancement*. Elle est la date de fin d'exécution de la dernière tâche. Elle est aussi appelée *makespan.*

Le *flowtime* est défini comme le coût d'encours, coût associé a la présence des tâches dans un atelier.

Le flowtime moyen, le flowtime pondéré, le nombre de tâches en retard, le retard d'un projet, l'avance, etc. peuvent être des objectifs à atteindre.

1.5 Ordonnancement

Ordonnancer un ensemble de tâches, c'est programmer leur exécution en allouant les ressources requises et fixant leurs dates de débuts [08].

Un ordonnancement consiste en une programmation prévisionnelle détaillée des ressources mobilisées dans l'exécution des opérations nécessaires à la production élémentaire de biens sur un horizon très court. [10].

Un problème d'ordonnancement consiste à "programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution" ([11]).

2. Les différentes structures d'atelier

Derrière cette définition générale se cache une multitude de problèmes, dont les modèles, les problématiques et les approches de résolution peuvent varier considérablement. Dans le cadre de notre étude, nous nous intéressons plus particulièrement aux problèmes d'ordonnancement d'atelier. Les tâches sont alors généralement désignées sous le nom d'opérations et regroupées en travaux (ou *jobs)* et les ressources sont supposées disponibles. Une machine peut être considère comme une ressource.

Nous employons les termes en anglais de *flowshop, job shop*, *open shop* pour désigner les structures d'atelier que nous décrivons ci-dessous.

Selon la structure de l'atelier, on distingue généralement les situations suivantes:

- **Les problèmes *à une machine*:** dans ce cas, chaque travail est constitué d'une seule opération qui doit être effectuée sur la machine considérée

• **Les problèmes *à machines parallèles* :** chaque travail est également constitué d'une seule opération mais celle-ci peut être effectuée par plusieurs machines. En plus du problème de séquencement des opérations, la question de l'affectation des opérations aux machines se pose donc. De plus, les machines peuvent être identiques, uniformes ou quelconques. Dans ce dernier cas, l'ensemble des opérations qu'elles peuvent traiter, ainsi que les durées de traitement, peuvent varier d'une machine à l'autre

• **Les problèmes de *flowshop* :** la gamme de fabrication, qui correspond à l'ordre d'exécution des opérations au sein des travaux, est linéaire, fixée à l'avance et identique pour tous les travaux, ce qui permet de numéroter les machines dans l'ordre d'exécution des opérations à traiter

• **Les problèmes de *jobs hop* :** la gamme de fabrication est linéaire et fixée à l'avance, mais peut varier d'un travail à l'autre. Une tâche peut revenir une second fois sur la même machine.

• **Les problèmes *d'open shop* :** la gamme de fabrication n'est pas fixée à l'avance

3. Le flowshop hybride

Ce problème d'atelier correspond à une réalité industrielle où des applications de différents domaines industriels seront présentées.

Soit un processus de production en série. Les produits fabriqués passent dans un premier temps dans un premier étage, puis dans une seconde, etc. Supposons qu'il y a « k » étages en série. Dans chacun d'eux, on considère que les machines sont regroupées en parallèles. Un groupe contient un ensemble de machines susceptibles d'exécuter une même opération. Elles sont alors équivalentes dans leur fonctionnement mais pas forcément dans leurs performances. La durée d'une opération peut dépendre non seulement des ressources choisies dans le groupe mais aussi du nombre de ressources affectées à l'opération (si la ressource est une machine, on peut parler alors de machines parallèles identiques, uniforme ou différentes). On supposera qu'il n'existe qu'un groupe de ressources par étage (où chaque produit subit un nombre d'opérations au plus égal à k, k étant le nombre d'étages).

Ce type de problème est appelé **flowshop hybride à k-étages**.

Remarques 1 :

a- Dans un flowshop hybride, les gammes de fabrication sont toutes identiques - comme dans tout flowshop et les machines sont regroupées par étages.

b- De plus, si le nombre d'étages est égal à 1, alors on est en présence soit d'un problème à machines parallèles, soit d'un problème à une machine. En revanche s'il n'y a qu'une seule machine à chaque étage, alors le problème est équivalent à un problème classique de flowshop.

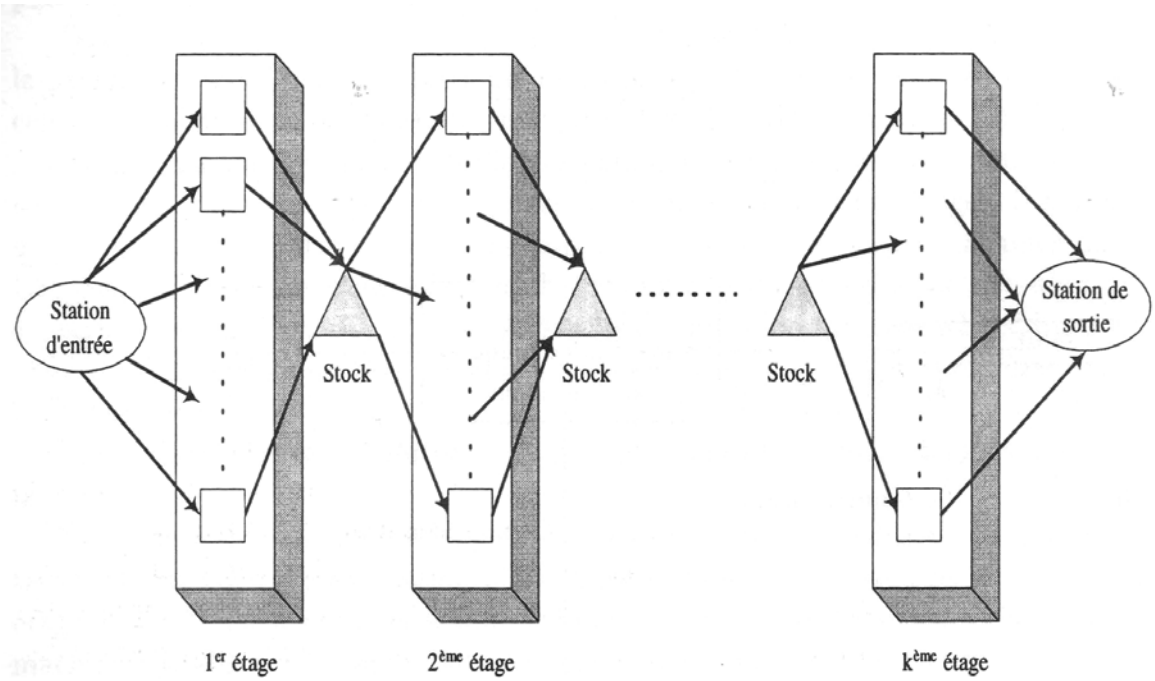


Figure 1.1 : Représentation d'un flowshop hybride à « k » étages.

La Figure 1.1 présente un problème de flowshop hybride. On considère qu'il y a une "station d'entrée". Ensuite, il y a la succession des k-étages. Chaque étage « r » comporte $M^{(r)}$ machines. Enfin, l'atelier se termine par une "station de sortie". Entre les différents étages, des zones de stockages à capacités finies ou infinies

peuvent exister.

MacCarthy [12] a réalisé un schéma hiérarchique pour la prise en compte des flowshops hybrides dans les problèmes d'ordonnancement présenté par la figure 1.2.

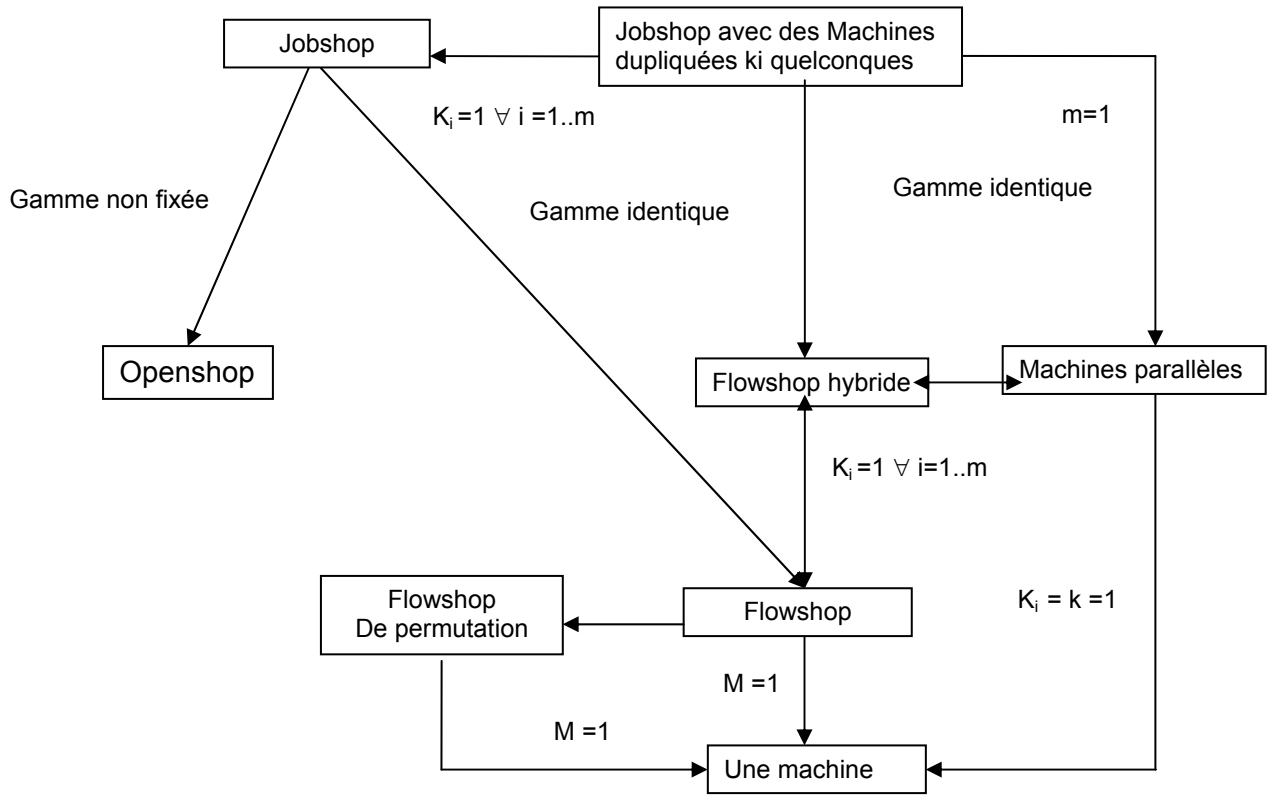


Figure 1.2 : Prise en compte hiérarchique du flow-shop hybride.

4. Classification et notation

Les notations de RinnooyKan et Graham [13] ne sont pas suffisantes pour prendre en considération le problème de flow-shop hybride. Vignier [14] a actualisé et étendu la notation à trois champs $\alpha/\beta/\gamma$.

Le champ « α » se compose de quatre paramètres représenté par $\alpha_1\alpha_2\left(\left(\alpha_3\alpha_4^{(l)}\right)_{l=1}^{\alpha_2}\right)$.

Le premier paramètre désigne le type de problème. Si α_1 = FH, le problème désigne

un flow-shop hybride. α_1 peut prendre ses valeurs dans l'ensemble {Ø, O, J, F}.

Le second paramètre α_2, indique le nombre d'étages. Dans un flowshop hybride il est supérieur à un.

α_3 est égale soit à Ø, P, Q, R pour désigner respectivement un problème à une seule machine, à machines parallèles, identiques, uniformes ou quelconques.

Le nombre de machines de l'étage « r », $M^{(r)}$ en général, est indiqué dans α_4. $M^{(r)}(t)$ indique que ce nombre est variable en fonction du temps. C'est la notion de *profil variable*, elle permet de tenir compte de la disponibilité éventuelle des ressources (Sanlaville, 1992)[15]. α_4 peut être représentée par le Ø. α_3 et α_4 sont répétés par couple, autant de fois qu'il y a d'étages.

• Le champ ß définit l'ensemble des contraintes prises en compte (aussi bien celles concernant les travaux que celles concernant le procédé de fabrication). Une forme générique de ce champ peut être notée $\beta = \beta 1, \beta 2, .$ Les βi peuvent prendre de nombreuses valeurs en fonction de la particularité du problème étudié par rapport au problème de base.

Exemple 1: $FH5,\left(\left(P3^{(l)}\right)_{l=1}^{2},\left(P2^{(l)}\right)_{l=3}^{5}\right)$ désigne un flow-shop hybride à 5 étages composé de 3 machines identiques aux deux premiers étages et de 2 machines identiques pour les autres. Il se note aussi FH5, ((P3$^{(1)}$ P3$^{(2)}$ P2$^{(3)}$ P2$^{(4)}$ P2$^{(5)}$) ou FH5 ((P3)^2 - (P2)^3).

5. Des exemples de flowshop hybride

Dans de nombreuses entreprises manufacturières, des problèmes d'ordonnancement se posent en termes de flowshops hybrides. Nous présentons ci-dessous deux exemples en détail.

5.1. Bloc opératoire d'un hôpital

Chaabane [16] a étudié les systèmes du Gestion prédictive des Blocs Opératoires. L'ordonnancement des salles d'opérations a été modélisé comme un problème de type flow-shop hybride à 2-étages sans temps d'attente entre les étages et avec des contraintes de précédences.

Un bloc opératoire est composé de m (l) salles d'opérations et d'une salle de réveil ou SSPI (Salle de Soins Post Interventionnelle). La salle de réveil est composée à son tour de m (2) lits équipés pour accueillir le patient après son intervention chirurgicale et assurer son réveil. La figure suivante illustre le cheminement des patients lors de leur séjour dans le bloc opératoire Les provenances des patients et les destinations à la sortie du bloc opératoire diffèrent suivant plusieurs critères (gravité de l'état de patient avant et après l'intervention, complications qui peuvent survenir au cours de l'intervention ou au cours du réveil etc...).

Le patient est hospitalisé dans le service de soins la veille du jour de l'intervention. Le jour de l'intervention, il est transféré de sa chambre vers la salle d'opérations où il subira son intervention (1). Une fois l'intervention réalisée, il est transféré de la salle d'opérations à la salle de réveil où il est affecté à un des lits de réveil (2). Une fois réveillé, le patient retourne à sa chambre (3). Ce cheminement est un cheminement normal, dans le cas où il n'y aurait pas de complications imprévues suite à l'intervention ou au cours du réveil.

L'ordonnancement a pour objectif de préciser le détail de la réalisation des interventions en salles d'opérations et en salle de réveil de manière à faciliter le travail des infirmiers, des aides-soignants et des brancardiers. Chaque intervention s'est vue affectée à une salle d'opérations sans préciser son ordre de passage dans la journée. La figure suivante illustre le cheminement des patients lors de leur séjour dans le bloc opératoire

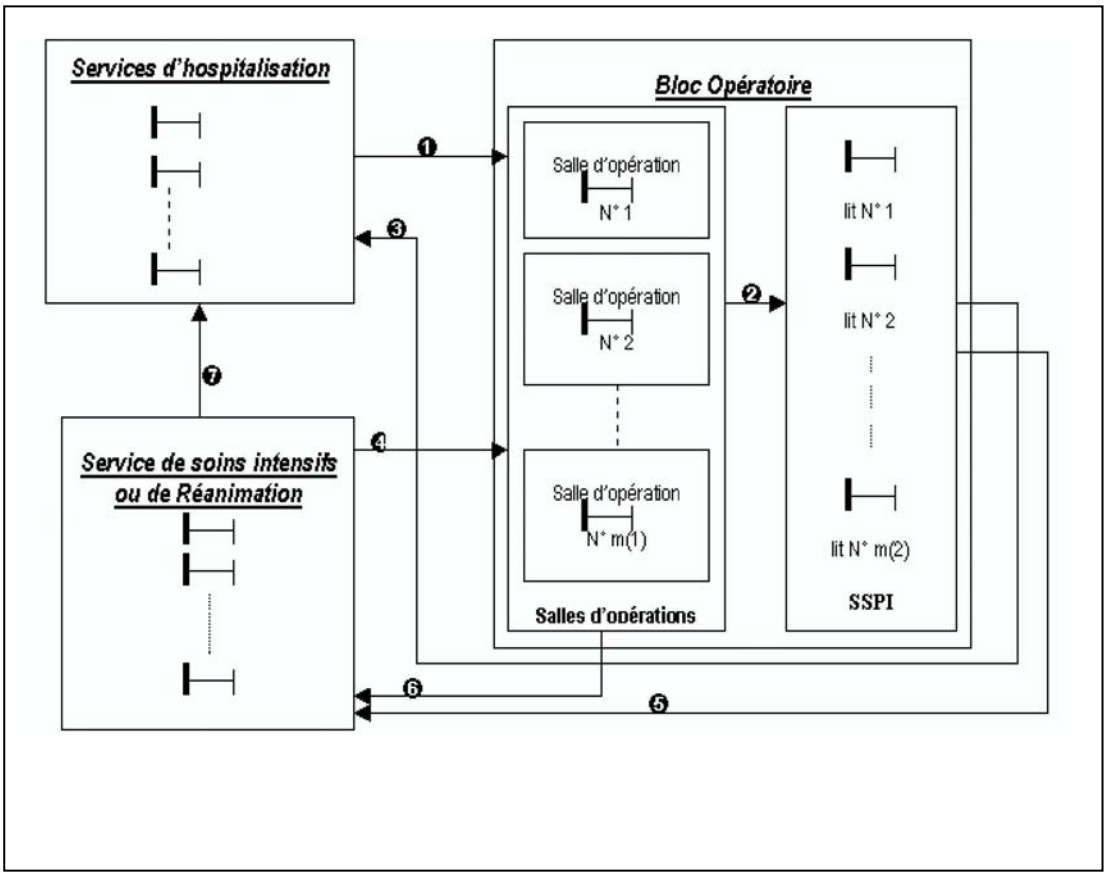


Figure 1.3. Provenance et destination des patients avant et après l'intervention.

Le problème à résoudre s'énonce comme suit : les N travaux à ordonnancer modélisent les patients. Chaque patient doit passer tout d'abord dans une salle d'opérations et ensuite dans un lit de la salle de réveil sans temps d'attente entre les deux. Le bloc opératoire se compose de m (l) salles d'opérations et m (2) lits en salle de réveil (m (l) < m (2)). Un flow-shop hybride à 2-étages définit donc l'organisation à ordonnancer (deux ensembles de ressources existantes en plusieurs exemplaires sont utilisés en séquence). L'hypothèse de ressources identiques est retenue pour chaque étage, mais on réserve le droit d'interdire une salle d'opérations en cas de sous-équipement vis à vis d'une intervention donnée. Les temps opératoires p (i, k) sont égaux aux durées d'intervention pour l'étage 1 et aux durées de réveil pour l'étage 2. A l'étage 1, une contrainte de nettoyage de la salle d'opérations doit être prise en compte. Elle peut être modélisée par un temps de démontage d'outils (Rnsd) au niveau de la classification des problèmes d'ordonnancement. Nous

chercherons à minimiser la plus grande date d'achèvement des travaux (interventions) afin de minimiser les heures supplémentaires.

5.2. L'industrie du verre [14]

Décrivons le procédé de fabrication des bouteilles en verre et montrons qu'il s'agit bien d'un problème de flow-shop hybride. Le procédé de fabrication est semi-continu. La partie continue, les fours, consiste à mélanger et à maintenir en fusion les matières premières et à distribuer le verre en fusion sous forme de gouttes appelées "gobs" aux machines de moulage. Ensuite le procédé est discontinu. Les gobs parviennent aux machines de moulage afin de constituer les différentes formes de bouteilles. Le procédé de moulage fait intervenir bien évidemment un moule mais aussi un procédé de fabrication comme le " pressé-soufflé" et le "soufflé-soufflé". Ensuite, un ensemble d'opérations à chaud et à froid permet de finir le produit [17]. Une illustration de ce type de fabrication est proposée en Figure 1.4.

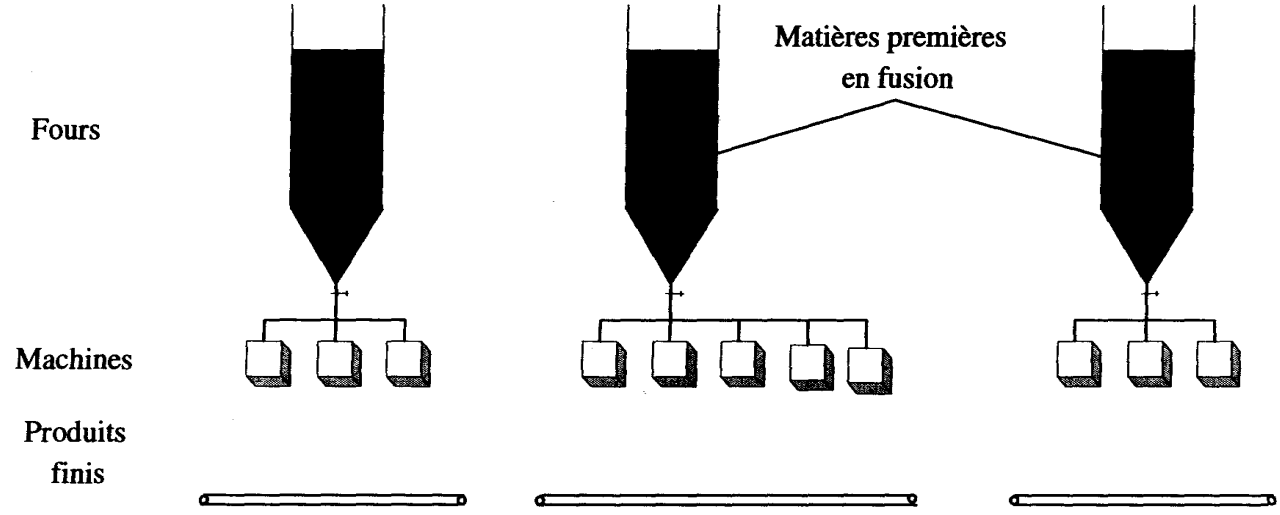


Figure 1.4. Représentation d'une fabrication de bouteilles de verre.

On peut donc considérer que l'outil de production est un ensemble de machines implantées, par groupe et en parallèle, en aval de distributeurs répartis eux-mêmes dans M usines sur un territoire géographique. Les distributeurs fournissent la matière première en fusion aux machines qui se chargent du moulage. Il existe différents types de machines aptes à traiter, avec plus ou moins de rapidité, des articles plus ou moins hauts, plus ou moins lourds: les cadences de fabrication sont donc différentes par machine et par article; elles peuvent varier dans des proportions importantes. Il faut retenir qu'à un instant donné, des machines reliées à

un même distributeur ne peuvent produire que des articles de même qualité. De plus, pour des raisons tant humaines qu'économiques et techniques, les changements de qualité sur les distributeurs sont délicats [18].

Il est clair que ce type de production est un problème particulier de flow-shop hybride à 2 étages: au premier étage il y des distributeurs c'est-à-dire les fours et au deuxième étage, les machines de moulage. De nombreuses contraintes sont à prendre en compte (temps de montage dépendant de la séquence au premier étage dus aux changements de couleurs dans les fours, temps de montage non dépendant de la séquence au deuxième étage, temps de recouvrement permettant de modéliser le processus discontinu, dépendances liant chaque machine du premier étage avec un sous ensemble des machines du deuxième étage...).

Nous trouverons aussi le modèle de flowshop hybride dans l'industrie de la moquette. Le premier étage est constitué d'une vingtaine de "machines à tricoter" qui fournissent des rouleaux de moquettes de diverses qualités, avec divers motifs à partir d'un rouleau de plastique et de très nombreuses bobines de fil. Les moquettes, généralement sans couleur, peuvent être stockées dans un immense entrepôt. A la demande, ces rouleaux sont repris pour subir une opération de teinture. La troisième opération consiste à enduire l'envers des rouleaux d'une couche de latex. Une dernière opération réalise l'emballage du rouleau de moquette.

Le modèle de flowshop hybride est aussi présent dans les entreprises de plastique. En période de saturation de la demande, les entreprises se trouvent de plus en plus coincées entre deux contraintes : pour préserver ou accroître leur part du marché, elles cherchent à personnaliser les produits, à les adapter aux goûts ou aux contraintes techniques exactes de leurs clients ; or cet objectif commercial est contradictoire avec les principes généraux de la production qui veulent que l'on privilégie les séries longues. Le conflit permanent entre le directeur commercial et celui de la production sera résolu grâce à l'introduction d'une organisation de type atelier flexible. Du jour au lendemain on pourra changer la forme, la couleur voire la matière des objets pour satisfaire tel ou tel client.

6. Un état de la connaissance et de la recherche sur le flowshop hybride à 2-étages

De notre recherche bibliographique, nous reprenons, dans leur grande ligne et leurs principales originalités, les travaux qui, selon nous, marquent des jalons importants

dans la résolution du flowshop hybride à 2-étages. Dans sa thèse de doctorat, Vignier [14] a rédigé un grand chapitre « état de l'art » où il a relaté un grand nombre de publications.

6.1 Les travaux précurseurs Vignier (1999) a cité notamment les travaux de Mac Naughton [19], Shen et Chen [20], Buten et Shen [21], Horn [22], Paul [23], Langston [24], Gupta [25], Koulamas et Smith [26] et Sriskandarajah et Sethi [27].

6.2 Les travaux des continuateurs de la période 1990-1999,

Les références citées sont Sheral [28], Gupta et Tunc [29], Deal et Hunsucker [30] , Lee et LIN [31], Gupta et Tunc [32], Lee et Vairaktarakis [33], Gupta, Hariri et Potts [34], Proust et Grunenberger[18], Chen [35], Li[36], Vignier et al. [37], Guinet et al.[38], Koulamas [39], Houari et M'Hallah [40] et Riane et al. [41].

6.3. Les recherches et travaux contemporains (2000-2008)

Par travaux contemporains, nous sous entendons les travaux effectués dans le dernier octonat.

De récents travaux ont traité les problèmes de flow-shop hybrides à 2-étages de domaines variés. Ils ont proposé des méthodes exactes de résolution pour des problèmes de petites tailles. Les méthodes exactes utilisées étaient de type Séparation et évaluation (Branch and Bound) et à solveurs pour les modèles de programmation linéaire en nombres entiers.

- Glass et al. [42] considèrent le problème des machines parallèles quelconques. Ils fournissent une heuristique, analysent sa performance et étudient la complexité du problème de machines parallèles.

- Hall [43] a décrit une notation et un schéma de classification des problèmes d'ordonnancements à machines parallèles. Il fournit des résultats sur la complexité de certains problèmes. Pour les deux fonctions objectif, longueur d'ordonnancement et coût d'encours, si le nombre de machines est égal à deux ou quelconque, les temps d'exécutions de tâches unitaires ou arbitraires, les problèmes sont montrés polynomiaux. Si les temps d'exécutions des tâches sur le serveur sont quelconques

et non unitaires sur les machines, la complexité est encore un problème ouvert. Il propose la règle SPT/FAM, une méthode polynomiale en O(nlog(n)) de résolution du problème $P2,S|s_i = 1|\sum C_{\max}$. Elle consiste a trié les travaux selon le plus petit temps d'exécution pour le serveur et les affecte à la première machine disponible.

- Kravchenko et Werner [44] proposent une heuristique pour minimiser la somme des dates de fin des travaux dans le cas du temps de chargement des tâches *unitaires* appelés « setup ». En tout étape de l'algorithme, on choisit une tâche, de plus petit temps d'exécution et qui ne génère pas un conflit parmi les tâches non encore ordonnancées. Si un telle tâche n'existe pas, on choisira arbitrairement une tâche pour exécution.

- Brucker et al. [45] ont montré que les problèmes $P2,S|p_i,s_i|C_{\max}$, $P2,S|s_i = s|C_{\max}$ et $P2,S|p_i = p|C_{\max}$ sont NP-Durs. Même pour le cas le plus simple, où les temps d'exécution sont unitaires, le problème reste NP-dur. Par contre le cas particulier de $P2,S|p_i + s_i = a|C_{\max}$ est résolu polynomialement. Ils montrent aussi que $P2,S|p_i = p, r_i, s_i = s|C_{\max}$ est polynomial avec r_i , le temps de début de chargement des tâches.

- Pour le problème $P2,Sl|s_i|C_{\max}$, Abdekhodaee [46] a montré qu'il est NP-Complet au sens fort et a présenté une modélisation par un programme quelconque en nombres entiers. Il a également étudié le cas où les temps d'exécution des tâches sont considérés comme « courts » et de longueurs égales. Il a proposé deux heuristiques efficaces en O(nlog(n)).

- On appelle « machine batch », une machine qui peut exécuter simultanément un lot de tâches. Ling-Huey [47] considère un flowshop hybride à 2-étages avec une machine batch au premier étage et une seule machine au second étage où les temps d'attente des tâches au second étage ne dépassent pas une valeur, borne supérieure donnée. Il a proposé une heuristique et un modèle en nombres entiers. Par une expérimentation numérique, il a étudié et justifié l'efficacité de cette heuristique.

• Oguz et al. [48] ont proposé des heuristiques pour la résolution du flowshop hybride à 2-étages afin de minimiser la longueur de l'ordonnancement. Les heuristiques sont gloutonnes. Les tâches sont ordonnancées selon une liste donnée, formée par des règles de priorités entre les tâches définies dans la bibliographie. Quelques bornes inférieures sont fournies pour être utilisées dans les études de performance. L'analyse en moyenne de ces heuristiques est faite par une étude expérimentale en générant aléatoirement des instances de problèmes. Les résultats suggèrent quelle sont efficaces.

• Les machines sont supposées ne pas être disponibles en tout instant du à une panne impromptue ou à leur maintenance préventive. Xijun Wang [49] a traité le flowshop hybride à 2-étages avec une disponibilité des machines limitées afin de minimiser la longueur de l'ordonnancement. Il a étudié ce problème en supposant que les intervalles du temps d'indisponibilité appelés des « creux » sont connus en avance. Il a montré que ce problème est NP-dur dans le sens fort même s'il n'y a qu'une seule machine à un étage et une seule période creuse pour la dite machine. Il a aussi étudié la notion d'approximabilité de ce modèle et a montré APX –difficulté de ce problème avec des « creux » sur tout le second étage ou sur tout le premier étage avec au moins un « creux » . Ils fournissent deux algorithmes avec des ratios très proches du plus mauvais cas dans le cas d'une seule machine au premier étage et présentant qu'un seul « creux ». Ils fournissent encore un algorithme avec un ratio pour le cas des périodes de d'indisponibilités dans un centre informatique à un serveur.

• JINXING XIE et al. [50] ont étudié le problème de flowshop hybride à 2-étages à machines parallèles à chaque étage sous la contrainte no-wait entre les étages afin de minimiser la longueur d'ordonnancement. Ils ont développé une heuristique de type MDA (minimum deviation algorithm). Par une expérimentation numérique en moyenne, ils ont montré l'efficacité de cette heuristique comparativement à d'autres algorithmes d'approximations connus. Un grand nombre de simulations ont été effectuées sous divers conditions d'ateliers et les résultats ont montré que les solutions fournies sont proches de l'optimum sous la majorité des conditions.

- Dans cet article, Abdekhodaee et al. [51] ont étudié le problème d'un centre informatique formé d'un serveur et de deux machines parallèles. Les temps d'exécutions des tâches sur le serveur sont unitaires et l'objective est de minimiser la longueur d'ordonnancement. Les articles de la littérature considèrent souvent que le temps de chargement nécessite à la fois le serveur et une machine. Dans cet article, cette contrainte est relaxée et le problème est celui d'un flowshop hybride à 2-étages sans espace d'attente entre les deux étages. Un algorithme optimal et polynomial en O(n log(n)) est proposé. Il est montré que la résolution optimale de ce problème conduit à une solution optimale au problème sans la contrainte d'espace tampon entre les deux étages.

- Allaoui [52] a étudié un cas très fréquent dans le monde industriel, la ou les machines peuvent être soumises à quelques périodes d'indisponibilité due aux activités de maintenance corrective où préventive. Il a étudié le cas d'une seule machine au premier étage et plusieurs machines au second étage pour minimiser le makespan. Il considère que chaque machine est soumise à au plus une période d'indisponibilité et que les dates du début et de la fin de chaque période sont connues à l'avance. Il discute la complexité du problème et donne un modèle pour la méthode Séparation et Evaluation. Il étudie la performance de trois heuristiques appelées Liste algorithme, LPT Algorithme et H-Heuristique.

- Jiyin Liu [53] a étudié des lots des tâches en transmission continue dans un flowshop hybride à deux étages afin de minimiser le makespan, un temps de chargement est pris en compte avant l'exécution de chaque lot sur une machine. Pour le problème où le nombre de lots est donné, il a montré qu'il est optimal d'utiliser une méthode de rotation pour l'allocation et le séquencement des sous lots sur les machines. La taille du sous-lot est optimisée en utilisant la programmation linaire .Il considère le problème des sous-lots égaux et développe une méthode efficace pour déterminer le nombre optimale de sous lot .Pour le cas général,il propose une heuristiques et une méthode optimale,il analyse la performance pour le cas des sous lots égaux.

- Low [54] a étudié un cas particulier du flowshop hybride à deux étages avec « m »

machines quelconques au premier étage et une machine serveur au deuxième étage. L'objectif est de minimiser le temps de fin d'exécution le plus tôt de la dernière tâche dans le système. Pour l'exécution d'une tâche arbitraire, il est supposé qu'une opération peut être remplacée partiellement sur d'autre machine de premier étage, dépendant des contraintes d'exécution sur des machines. De tels problèmes d'ordonnancement surviennent dans certains cas pratiques et industriels de la fabrication des semi-conducteurs, l'industrie électronique, la fabrication des réacteurs et moteurs d'avions et de l'industrie pétrochimique. Il démontre que ce problème est NP-Dur, et fournit quelques heuristiques efficaces pour sa résolution. Par une expérimentation numérique, il montre qu'une combinaison de la règle de Johnson modifiée et de la règle First-Fit fournit une solution meilleure que les heuristiques.

• Hadda et al. [55] traitent un cas particulier du problème d'ordonnancement de flow shop hybride à deux étages. Le premier étage est constitué d'une seule machine (machine commune), quant au second, il comprend « m » machines dédiées. Chaque job nécessite exactement deux opérations. La première opération doit être réalisée sur la machine commune. Ensuite, et suivant le type du job, la deuxième opération est prise en charge par l'une des m machines du second étage. Contrairement à la plupart des travaux qui supposent que toutes les machines sont toujours disponibles sur l'horizon d'étude, ils supposent que l'une des m+1 machines est indisponible sur une période prédéterminée. Leur intérêt est la minimisation du makespan. Ils présentent un ensemble de propriétés ainsi qu'une analyse du pire des cas selon que la période d'indisponibilité concerne la machine commune ou l'une des machines du second étage.

CHAPITRE 2
PRÉSENTATION ET IMPLÉMENTATION DE L'ALGORITHME DE GUIRCHOUN ET MARTINEAU POUR LA RÉSOLUTION DU FLOWSHOP HYBRIDE À 2-ÉTAGES

2.1. Introduction

Dans un environnement informatique composé d'un « serveur » et de « M » machines parallèles identiques appelées processeurs, « N » *tâches* sont à exécuter sans interruption. Avant son exécution, une tâche i (i = *1,..., n)* doit être chargée sur le serveur. Ayant terminé un chargement, le serveur est libre d'en effectuer un autre et le chargement d'une tâche doit être suivi de son exécution. Ce problème est appelé un flowshop hybride à 2-étages et est noté $P2,S1|p_i,s_i|C_{\max}$.

Dans la littérature, il est prouvé NP-difficile.

Dans cet environnement, nous considérons que les temps de chargement d'une tâche sont tous unitaires. De plus, les temps de transfert ou de communication entre le serveur et les machines parallèles sont supposés négligeables. Le but est de déterminer un ordonnancement réalisable qui minimise la somme des dates de fin des travaux appelé « flowtime ». La récupération des données nécessaires à l'exécution d'une tâche est réalisée par la machine serveur. Durant ce chargement, il n'est pas nécessaire que la machine parallèle soit disponible. En effet, toutes les machines parallèles ont à leur disposition un coprocesseur de communication qui leur permet de recevoir les informations du serveur tout en continuant leurs opérations. Ce système informatique s'interprète comme un flowshop hybride à deux étages. Une seule machine « serveur » est disponible au premier étage, les « m » machines parallèles sont au second étage. Nous supposons qu'aucune aire de stockage ou une « zone tampon » n'existe entre les deux étages. Le flowshop hybride est important.

L'algorithme de Guirchoun et Martineau [03] est présenté au paragraphe 2. Un exemple illustratif est même fourni au paragraphe suivant. Une implémentation par nos soins est réalisée au paragraphe 4. Une conclusion avec des perspectives est exposée à la fin de ce chapitre.

2.2. Modélisation d'un système informatique par un flowshop hybride :

Si on se place dans un environnement de calcul, la récupération des données nécessaires à l'exécution d'une tâche est réalisée par une machine du réseau informatique couramment appelée serveur de réseau.

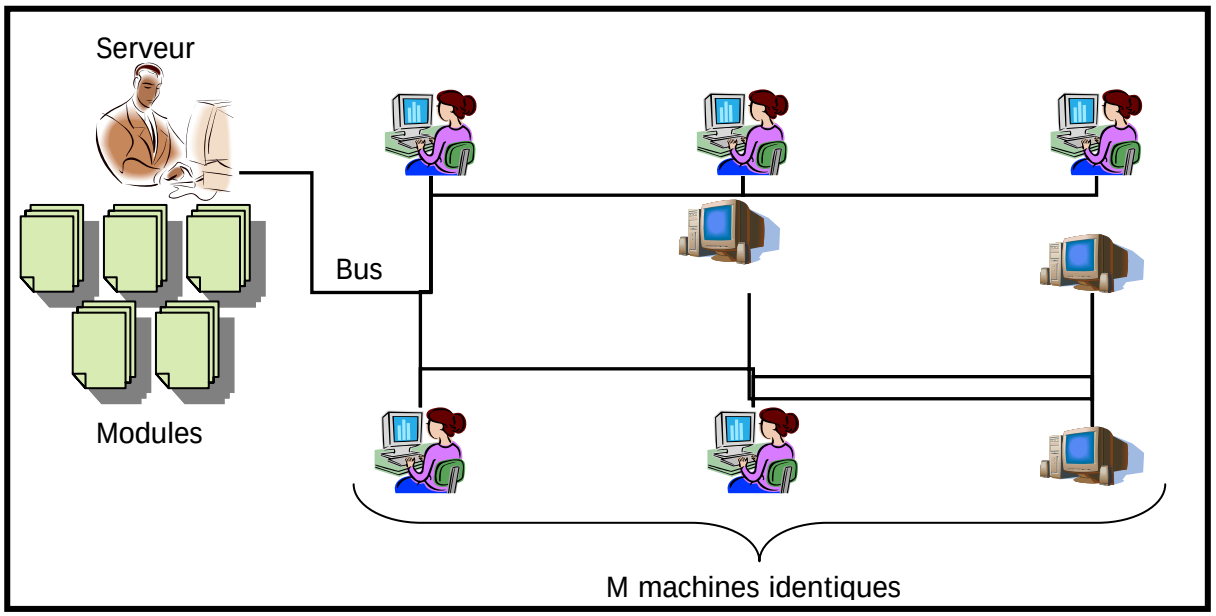


Figure. 2.1. Salle Réseau de connexion

Les systèmes informatiques se modélisent comme des centres formés d'un seul serveur et à machines parallèles.

Si les temps d'exécution des tâches sont égaux au premier étage, les temps de chargement sont aussi égaux et si l'objectif est le flowtime, Guirchoun et Martineau [03] ont montré en utilisant une réduction polynomial que le problème flowshop hybride à une seule machine au premier étage et « m » machines parallèles au second étage à contrainte no-wait entre les deux étages est aussi difficile que le problème de « m » machines parallèles.

Nous l'énonçons ci-dessous sans preuve.

Théorème 1: [03]

Pour $\alpha \in \{m,o\}$, si les temps de chargement sont égaux s_i et les temps d'exécution p_i sont arbitraires, le problème $P\alpha, S1|s_i = s|\sum C$ se réduit polynomialement à celui de $FH2,(1,P\alpha)|p_i = s, nouwait|\sum C_i$.

2.3. Algorithme de Guirchoun et Martineau [03]

Guirchoun et Martineau [03] ont étudié le problème d'ordonnancement de tâches sur des machines parallèles à un seul serveur dans un système informatique. Ce problème est noté FH2, (1, Pm)/nowait, pi1 = 1, $p_{i,2}$ > / $\sum C_i$. Pour un cas particulier, où les temps d'exécution des temps sur les machines sont entiers, ils étaient capables de proposer un algorithme qu'ils montrent polynomial.

L'algorithme proposé donne une solution optimale en supposant que les temps d'exécution au deuxième étage sont des entiers strictement positifs. Les temps d'exécution au premier étage sont unitaires, l'algorithme s'établit par :

Algorithme 2.1. Algorithme de Guirchoun et Martineau [03]

Début
$\Gamma s = 0, \Gamma_1 = 1, \Gamma_2 = 2, \ldots, \Gamma_m = m.$
Déterminer les ensembles A et B
Tant que (AUB≠Ø) **Faire**
  Soit M_k La machine telle que $\Gamma_k = \min\{\Gamma_k | k=1 ; \ldots ; m\}$
  Si ((∃compatible de A) ou (B≠Ø)) alors
    Placer le travail i de A sur M_k
    A←A\{i}
    $\Gamma_k = \Gamma_k + P_{i2}$
  Sinon
    Placer le travail i de B de plus petit $P_{i,2}$ sur M_k
    B←B\{i]
    $\Gamma_k = \Gamma_k + P_{i2}$
  Finsi
Fin tant que.
Fin

Γ_k représente la date de fin d'exécution de la dernière tâche sur la machine M_k avec $1 \leq k \leq m$ (i.e., relativement à un ordonnancement partiel courant). *(Γs, Γ_1, Γ_2,..., Γ_m)* décrit le profil de l'ordonnancement partiel où Γ_s représente la date de fin de la dernière tâche sur le serveur.

L'ensemble des travaux est divisé en deux sous ensembles *A* et *B* où $A = \{i \mid p_{i,2} < m\}$ et $B = \{i \mid p_{i,2} \geq m\}$. L'ensemble *B* est trié selon un ordre non décroissant des $p_{i,2}$.

Nous dirons qu'une tâche *i* est *compatible* avec l'ordonnancement partiel courant, si après son exécution, toutes les dates de disponibilité des machines restent différentes.

Quant une tâche *i* est compatible et si elle s'est exécutée sur la machine *k* on a forcément

$\Gamma_k \neq \Gamma_v \ \forall\ v, k$, $1 \leq v \leq m$ et $v \neq k$.

2.4 Exemple illustratif :

Soit à exécuter 10 tâches dans un système informatique formé d'un serveur et 4-machines parallèles. Les temps d'exécution des tâches sont donnés par le tableau ci dessous :

Le temps de chargement de chaque tâche sur le serveur p_{i1}est de « une » unité de temps.

I	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\boldsymbol{p_{i1}}$	1	1	1	1	1	1	1	1	1	1
$\boldsymbol{p_{i2}}$	1	2	3	3	4	5	6	8	9	9

Tableau 2.1. Temps d'exécutions des tâches les machines

Les sous ensembles A et B sont formés respectivement de tâches dont $p_{i2} < 4$, $p_{i2} \geq 4$,4 étant le nombre des machines. Ils seront alors: A = {1, 2, 3, 4} et B = {5, 6, 7, 8, 9, 10}.

L'application de l'algorithme sur l'instance donne l'ordonnancement suivant.

Itération 1 :

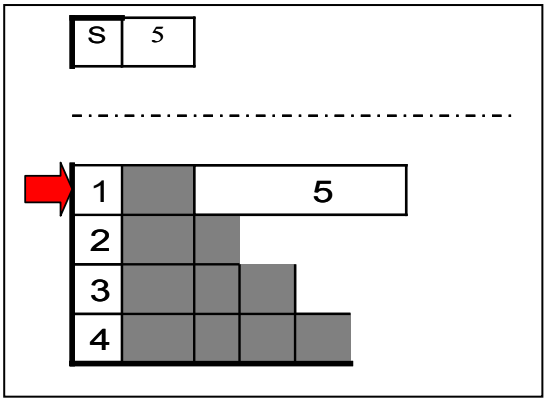


Figure 2.2 : Itération 1

Itération 2 :

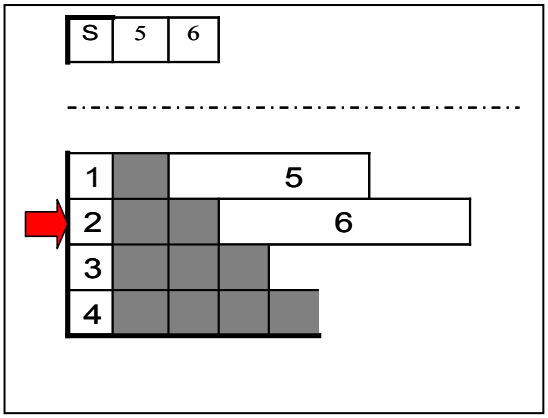


Figure 2.3 : Itération 2

On exécute la tâche **5** sur la machine **1**

On exécute la tâche **6** sur la machine **2**

Les itérations se succèdent. On omettra volontairement de les reproduire. Nous donnons l'avant dernière et la dernière itération.

……

Itération 9 :

On exécute la tâche **10** sur la machine **2**

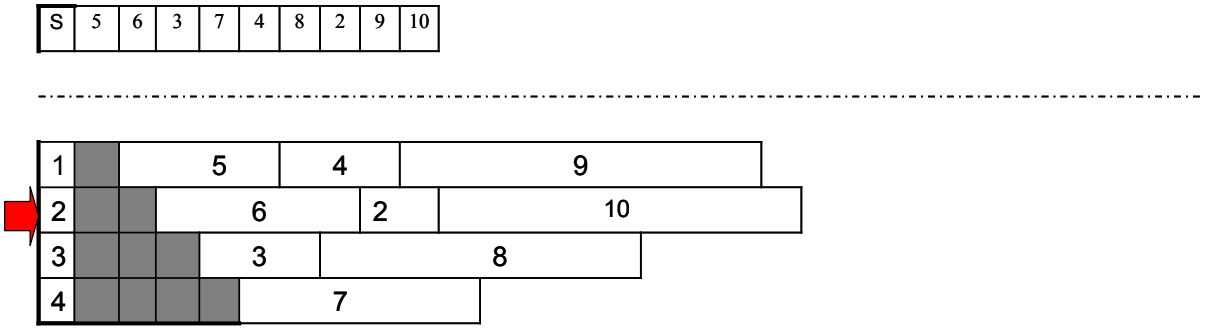


Figure 2.4 : Itération 9

A l'itération 10 :

On exécute la tâche **1** sur la machine **4**

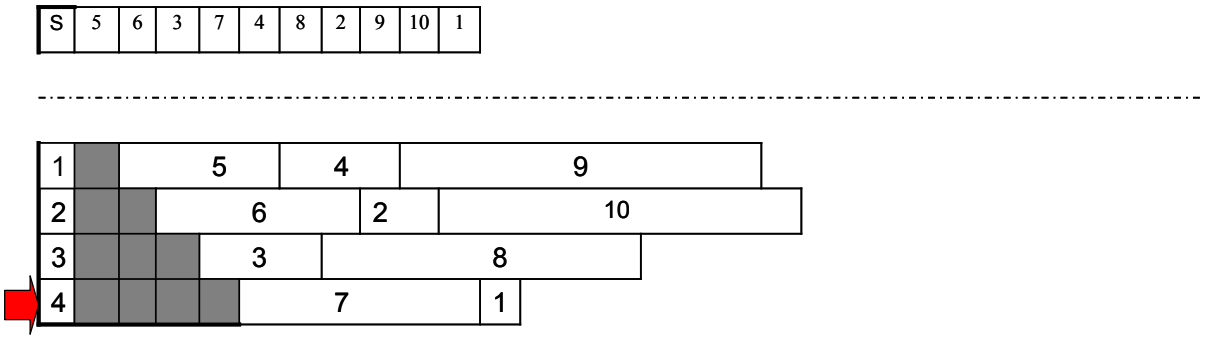


Figure 2.5: Ordonnancement final et optimal.

Ainsi se lit directement l'ordonnancement optimal de longueur 105 unités de temps. Les temps de fin d'exécution sont C_1 = 11, C_2 = 9, C_3 = 6, C_4 = 8, C_5 = 5, C_6 = 7, C_7 =10, C_8 = 14, C_9 = 17 et C_{10} = 18.

2.5. Implémentation

Dans l'algorithme, les représentations respectives des dates de disponibilité des machines « k » au second étage et des deux sous ensembles A et B sont données par un vecteur dynamique et une structure de type « ensemble ». Pour trier les éléments de l'ensemble B selon l'ordre non décroissant de ses temps d'exécution, on utilise un algorithme de « tri rapide », de l'anglais « Quick sort ». L'ensemble A ne nécessite pas un tri de ses éléments. Sachant que l'algorithme nécessite la détermination des tâches dites « compatibles » sur la machine « k » au second étage, ci-dessus définis, on a programmé une fonction de « renvoi » du numéro de la dite tâche. Sinon, si aucune tâche n'est compatible, une valeur nulle lui est attribué.

L'ensemble des tâches est représenté par une matrice dynamique à deux colonnes A_d. La première et la seconde colonne de A_d représentent respectivement le numéro de la tâche et son temps d'exécution au deuxième étage.

Pour identifier les tâches exécutées sur les machines, une matrice B de 3 colonnes et « k » lignes est utilisée. Ces colonnes représentent respectivement le numéro de la tâche, sa date de début d'exécution et sa date de fin d'exécution. L'indice de chaque ligne représente l'indice de la machine au second étage. Nous sous-entendons par instance, les données formées de paramètres attribués aux tâches d'un problème. Pour l'instance en cours, une procédure de calcul du flowtime est utilisée où sa valeur s'affiche dans l'interface qui sera développée ci dessous.

Cette implémentation est réalisée et développée par nos soins en utilisant un langage orienté objets où l'interface graphique est donnée dans la figure.2.6. Elle est du type « naïve », car elle est simple sans sophistications matérielles. Elle se compose de trois fenêtres respectivement de génération d'instances du problème, de la donnée de la séquence d'ordonnancement optimale et de l'affichage de la solution représentée par son digramme de Gantt.

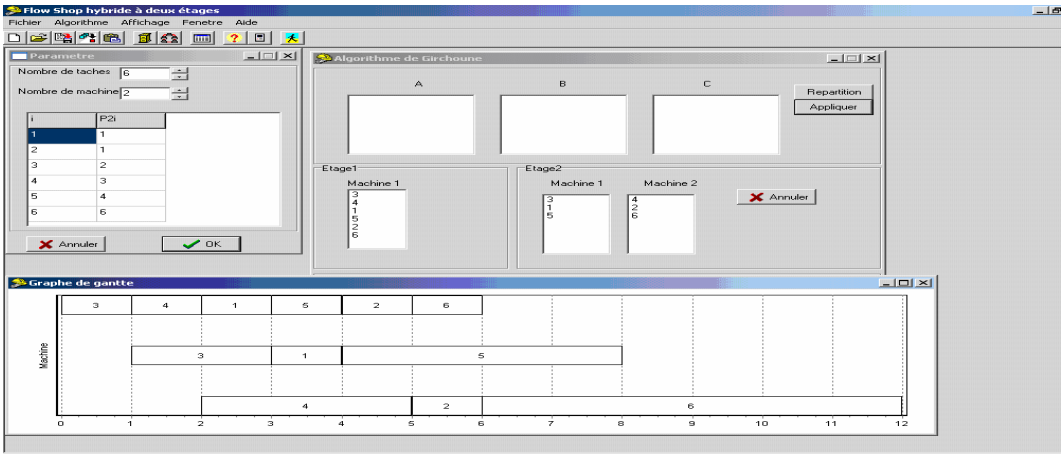


Figure 2.6 Fenêtre de l'application

En l'absence d'un jeu de tests pour ces problèmes, et pour tester l'efficacité de cette algorithme, un générateur a été réalisé par nos soins. La fenêtre de la figure 2.7 représente l'image d'une génération de données aléatoires et des paramètres d'instance d'un problème.

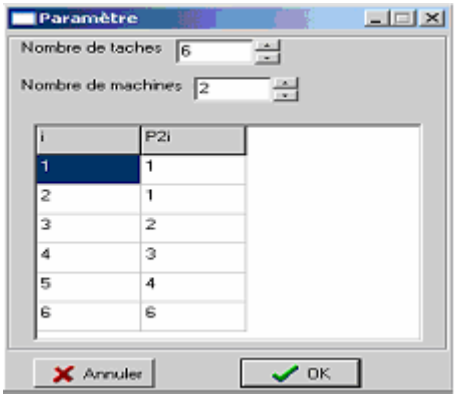


Figure 2.7. Génération de paramètres d'instances d'un problème.

Nous décrivons les principales fonctions et procédures utilisées pour implémenter la algorithme. Une procédure de chargement est utilisée pour une instance d'un fichier texte fournie par le générateur du jeu de tests. Après chargement, les données sont stockées dans une matrice D à trois colonnes représentant chacune d'elle respectivement le numéro de la tâche, son temps de chargement et son temps d'exécution. Une procédure de « tri rapide » est appliquée une seconde fois sur la matrice D pour ordonner les tâches selon l'ordre croissant de leur temps d'exécution. Pour déterminer la tâche admissible suivante, celle qui sera exécutée ultérieurement, on utilise une procédure de renvoie du résultat de type logique, vraie si la tâche est admissible, faux sinon. L'algorithme génère une séquence des tâches, qu'on affecte aux deux machines parallèles du deuxième étage suivant la règle FAM (First available machine). Pour représenter les tâches affectées aux machines, une matrice est associée pour chaque machine. Elle est composée de 3 colonnes associées au numéro de la tâche, sa date début d'exécution et sa date fin d'exécution. Elle est utilisée pour le calcul de la valeur de l'objectif makespan, et pour générer les états de sortie.

Deux états de sorties sont atteints. L'application réalisée permet également d'afficher des digrammes de Gantt donnés par la figure 2.8 ci-dessous. Le diagramme peut être copié et enregistrer sous une forme d'image à importer.

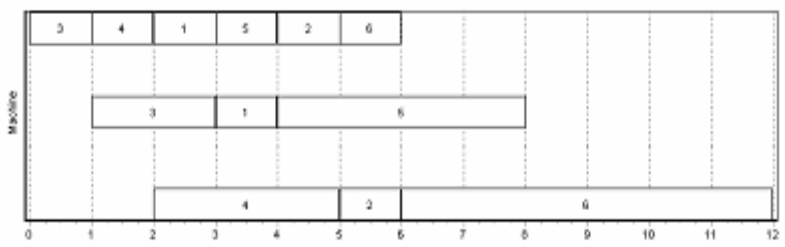


Figure 2.8 : Diagramme de Gant d'ordonnancement.

Si le nombre de tâches est assez grand, le diagramme devient illisible. Notre application a la possibilité d'éditer les résultats sous forme d'un état imprimable. Un exemple de telle application est ici fourni.

2.6 Conclusion et perspectives

L'avancé technologique sur les concurrents est une garantie essentielle pour le succès dans la compétition entre les entreprises. Le système d'atelier flexible permet de minimiser les dangers que représentent les faillites fréquentes des petites entreprises clientes; la souplesse de l'équipement rend les capacités de production immédiatement disponibles pour d'autres marchés. Il permet aussi la création de nouveaux emplois dans les activités de conception, de fabrication ou d'entretien des robots installés. Comme perspective, on se propose d’en inventer de nouvelles algorithme pour les problèmes d’atelier flexible non encore résolus par la communauté scientifique.

CHAPITRE 3
DEUX HEURISTIQUES POUR LA RESOLUTION DU PROBLEME D'UN SEUL SERVEUR A MACHINES PARALLELES

Le parallélisme a fait l'objet d'une grande attention ces dernières années grâce à son efficacité, une partie cruciale de l'intérêt des systèmes répartis. Il est d'intérêt de développer des algorithmes d'ordonnancement efficaces dans ce cadre. Nous considérons le même environnement informatique que celui vu auparavant. Il est composé d'un « serveur » et de « M » machines parallèles identiques appelées processeurs. « N » tâches sont à exécuter sans interruption. Avant son exécution, une tâche i (i = *1,..., n)* doit être chargée sur le serveur. Ayant terminé un chargement, le serveur est libre d'en effectuer un autre. Nous considérons que les temps de transfert (ou de communication) entre le serveur et les machines parallèles sont négligeables. Ce problème est considéré comme un flow shop hybrides à deux étages, avec une seule machine au premier étage (le serveur) et m machines parallèles au second étage. On étudie le cas de deux machines parallèles (m = 2). Avant d'exposer les deux heuristiques, donnons une notation et une description pour ce problème.

3.1. Notation et description

La notation des flowshop hybrides par Vignier [14] a été exposée au chapitre un. Pour des raisons pratiques, nous utilisons la notation de Abdekhodaee [02].

$P2,S1|p_i,s_i|C_{\max}$ ou $P2,S1|p_i,s_i|\sum C$ désigne un flowshop hybride à un serveur et deux machines parallèles au deuxième étage, p_i et s_i représentent respectivement le temps d'exécution et le temps de chargement d'une tâche.

Si t_i est le temps de début d'exécution de la tâche *i,* $c_i = t_i + s_i + p_i$ est le temps de fin d'exécution. Notons par $a_i = s_i + p_i$, la longueur de la tâche *i*.

Définitions 1:

- Un ensemble de tâches est régulier si pour tout couple de tâches (i, j) le temps d'exécution de la tâche i est plus court que la longueur de la tâche j,soit $p_i \leq a_j \ \forall$ (i, j) i≠j.
- Soit I_i le $i^{ème}$ temps d'oisiveté de la machine, le temps nécessaire pour la

machine, qui vient juste de terminer l'exécution de la i^{éme} tâche, de commencer l'exécution d'une tâche suivante.

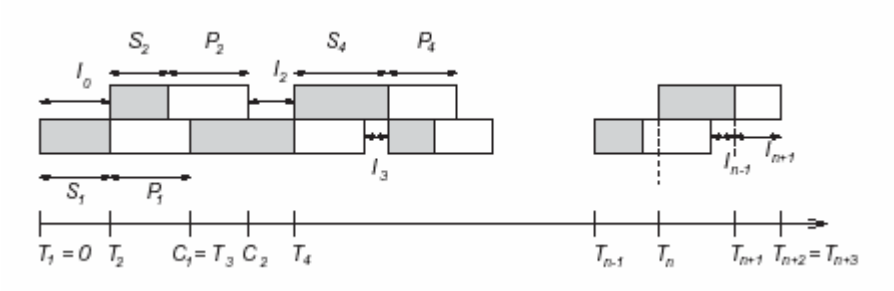


Figure 3.1. Temps d'oisiveté des machines.

Définition 2:

Le temps d'attente de serveur W_i est le temps mis entre la fin de chargement de la *(i+1)^{iéme}* tâche et le début de chargement de la *(i+2)^{iéme}* tâche d'un ordonnancement.

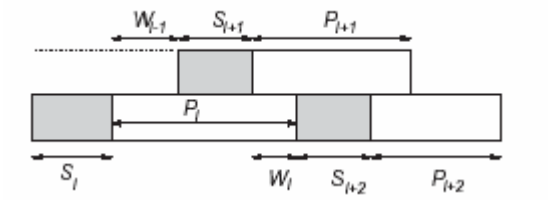


Figure 3.2 temps d'attente de serveur.

Notre but est la minimisation de la longueur d'ordonnancement.

Brucker et al. [44] ont montré que les problèmes $P2,S1|p_i,s_i|C_{\max}$, $P2,S1|s_i=s|C_{\max}$ et $P2,S1|p_i=p|C_{\max}$ sont NP-Durs si respectivement un code unaire est utilisé pour le premier problème et un code binaire pour les deux autres.

D'autre part ils montrent que le problème $P,S1|p_i=p,r_i,s_i=s|C_{\max}$ est polynomial où r_i est le temps de début de chargement. Même pour les cas les plus simples, s_i ou r_i unitaires, le problème reste NP-dur. Un cas particulier où les longueurs de tâches sont constantes et identiques, $P2,S1|p_i+s_i=a|C_{\max}$, est résolu polynômialement.

Théorème 1:

$\max\left\{\frac{1}{2}\left(\sum_{i=1}^{n} a_i + s_{\min}\right), \left(\sum_{i=1}^{n} s_i + p_{\min}\right)\right\}$ est une borne inférieure pour la longueur d'ordonnancement du problème $P2, S1|p_i, s_i|C_{\max}$.

Comme le problème est montré NP-Complet au sens fort, Abdekhodaee a développé deux heuristiques appelées « heuristique en avant » et « heuristique en arrière ». Nous les exposons ci-dessous. Elles ont fait l'objet d'une implémentation par nos soins en utilisant un langage évolué orienté objet. Ces heuristiques ont été testées sur un Pentium VI.

3.2.Heuristique en arrière (Backward heuristic)

Dans la suite, HB est le sigle de l'heuristique en arrière. Supposons que l'ensemble des tâches d'une séquence est un ensemble régulier définit ci-dessus, si les temps d'attente du serveur sont nuls et la dernière tâche de l'ordonnancement possède le plus petit temps d'exécution, cette séquence est optimale [02].

La notion de dernière tâche à plus petit temps d'exécution est l'étape initiale de cette heuristique montrée efficace et de complexité O (*nlog (n)*). Elle est de type « glouton ». Un ordonnancement est construit séquentiellement sans remettre en cause la position d'une tâche.

Algorithme 3.1 Heuristique HB

Début

Soient: *A* : = ensemble de tâches ordonnées selon l'ordre croissant de leur temps d'exécution.

B = Ø

i la tâche telle que p_i = min(p_i|*i*=*1*.....*n*)

A←*A*\{*i*}, *B*←*B*U{*i*}

Tant que (*A* ≠ Ø) **Faire**

Δ = s_i

Si ((∃ une tâche *i* admissible avec $p_i \leq \Delta$.) **Alors**

B←*B*U{*i*}, *A*←*A*\{*i*}

Sinon

Soit i la tâche tel que $p_i = \min(p_i | i=1...n)$

$B \leftarrow B \cup \{i\}$, $A \leftarrow A \setminus \{i\}$

Fin si

Fin tant que.

Ordonnancer les tâches de l'ensemble B sur la première machine disponible, de la tâche en queue de liste associé à B jusqu'à la tâche tête de liste.

Fin

Une tâche est admissible si elle ne génère pas un temps d'oisiveté à l'une des machines parallèles ou un temps d'attente supplémentaire du serveur.

L'heuristique utilise initialement une liste de tâches ordonnées par un ordre croissant de leur temps d'exécution. A chaque étape, elle favorise les tâches admissibles à temps d'exécution le plus élevée. Si aucune tâche admissible n'existe, on arbitrera en choisissant la première tâche de la liste. On insère cette tâche dans une liste juste avant la plus récemment ordonnancée. Pour déterminer l'ordonnancement approché fourni par l'heuristique, on attribuera la première tâche à la première machine, les autres tâches à l'une des deux machines disponibles, jusqu'à ce que toutes les tâches soient exécutées.

Exemple 1:

10 tâches sont à exécuter dans un système informatique formé d'un serveur et deux machines parallèles. Les temps d'exécution et de chargement des tâches sont donnés par le tableau ci dessous :

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
s_i	8	2	4	4	2	5	3	10	4	9
p_i	16	2	13	19	18	2	6	16	10	8

Tableau 3.1. Temps d'exécutions et de chargement des tâches.

Après exécution de l'heuristique, l'ordonnancement réalisable fournit est formé de la liste suivante (4, 5, 1, 8, 3, 9, 10, 7, 2 ,6).

Si $C_i(M_j)$ représente la date de fin d'exécution de la tâche i sur la machine j, les valeurs seront dans l'ordre de la liste $C_4(M_2) = 42$, $C_5(M_1) = 20$, $C_1(M_1) = 49$, $C_8(M_1) = 67$, $C_3(M_1) = 33$, $C_9(M_1) = 23$, $C_{10}(M_1) = 52$, $C_7(M_1) = 13$, $C_2(M_2) = 6$ et $C_6(M_2) = 44$.

L'ordonnancement réalisable est de longueur 67 et se lit directement sur le diagramme de Gant donné par la figure suivante :

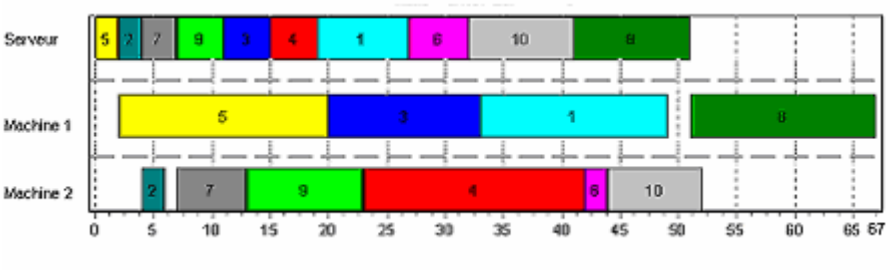


Figure 3.4. Diagramme de Gantt en appliquant l'heuristique HB

Notre implémentation permet de représenter graphiquement le diagramme de Gantt en utilisant une composante de notre langage orienté objet appelée TChart.

3.3. Une seconde heuristique de Abdekhodaee et al.: Heuristique en avant (Forward heuristic)

Le sigle HF représentera l'heuristique en avant. Elle est polynomiale en O(*nlog(n)*) comme la précédente. Son objectif est de minimiser le temps d'oisiveté de serveur. Elle s'énonce sous la forme ci-dessous :

Algorithme 3.2 Heuristique en Avant.

Etape 1 : Ordonner les tâches dans l'ordre décroissant de leur temps de chargement sur le serveur.

Attribuer au paramètre Δ la valeur nulle.

Etape 2 : Déterminer une tâche avec le plus grand temps d'exécution inférieur ou égale à Δ. Si une telle tâche n'existe pas, choisir une tâche appelée i avec le plus petit temps d'exécution. Affecter la tâche i à la première machine disponible. Soit j la dernier tâche ordonnée sur la machine 1 et k la dernier tâche ordonnée sur la machine 2 (Si k = 0 alors $C_k^2 = 0$, Si j = 0 alors $C_j^1 = 0$)

Etape 3 : Si $C_j^1 \le C_k^2$ alors $\Delta = C_k^2 - \max\left(C_k^2 - p_k, C_j^1\right)$ Sinon $\Delta = C_j^1 - \max\left(C_j^1 - p_j, C_k^2\right)$

Etape 4 : Répéter les étapes 2 et 3 jusqu'à toute la tâche sont ordonnancé

Dans cette deuxième heuristique, on ordonne les tâches dans une liste décroissante du temps de chargement des tâches sur le serveur. Son application sur le même exemple1 donne un ordonnancement de longueur 61 unités de temps, longueur plus réduite que celle donnée par l'heuristique HB. Après son exécution, (4, 5, 1, 8, 3, 9, 10, 7, 2 ,6) est l'ordonnancement réalisable obtenu. Les temps de fin d'exécution sont $C_1(M_2)$ = 40, $C_2(M_1)$ = 60, $C_3(M_2)$ = 53, $C_4(M_1)$ = 23, $C_5(M_2)$ = 24, $C_6(M_2)$ = 61, $C_7(M_2)$ = 59, $C_8(M_1)$ = 40, $C_9(M_1)$ = 50, $C_{10}(M_1)$= 58.

De même, un diagramme de Gantt à été obtenu et nous a fourni une valeur du makespan égale à 61 unité de temps. Nous omettons volontairement de le reproduire.

Un cas fréquent se rencontre dans l'industrie. Si un ensemble de tâches est régulier, leurs temps de chargements sont négligeables par rapport à leur temps d'exécution.

3.4. Approche par la régularité

La régularité d'un ensemble de tâches permet de simplifier le problème d'ordonnancement. Elle nous permet d'explorer des conditions sous lesquelles le problème peut être résolu de façon polynomiale et permet de modéliser le problème en un programme linéaire en nombres entiers [02]. Comme elle est définit précédemment, pour tout couple de tâche (i,j), $p_i \le p_j + s_j$. Dans la Figure 3.5, où le repère est par rapport aux axes temps d'exécution et temps de chargement, la régularité est représentée par l'aire hachurée en noir.

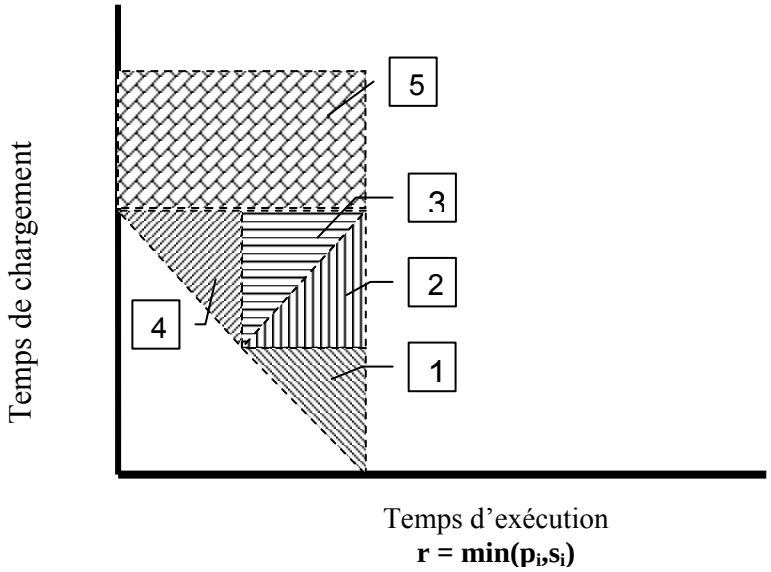


Figure 3.5 : temps de chargement en fonction des temps d'exécution.

Si la tâche de longueur la plus réduite de la région régulière est supposée de valeur r, $r = \min_i(p_i + s_i)$, alors aucun tâche ne peut se trouver dans la zone au-dessous de $p_i + s_i = r$. En outre, chaque tâche dans la partie hachurée de la Figure 3.5 a un temps d'exécution au moins égale à r. Cette région régulière, peut être divisée en plusieurs zones. Dans les zones 1 et 2, pour tout couple de tâches (i,j) les temps de chargement s_j respectif évoluent dans les intervalles respectifs] 0, r/2] $\cup$[r/2, p_i]et [r/2, p_i].

Pour chaque tâche dans la zone 3, son temps de chargement est plus grand que son temps de traitement, soit $r/2 \le p_i \le s_i$. Dans la zone 4 et 5, on a respectivement les évolutions de temps de chargement $p_i \le r/2 \le s_j$ et $p_i \le r \le s_j \quad \forall i, j$.

Il est remarqué qu'une heuristique particulière peut s'avérer efficace dans une zone et ne le pas être dans une seconde zone.

Nous rappelons que nous traitons le cas de deux machines au second étage. Un ordonnancement est dit alternatif si les tâches s'exécutent alternativement sur les deux machines. Si un ensemble de tâches se trouve complètement dans la zone 4 ou 5 alors tout ordonnancement alternatif formé d'une séquence arbitraire de tâches et une tâche de plus petit temps d'exécution placée à la dernière position, fournit un ordonnancement optimal.
En revanche, si toutes les tâches se trouvent dans zone 1, la règle LPT s'avère efficace.

En général, un problème peut ne pas satisfaire la condition de la régularité. Deux tâches consécutives peuvent s'exécuter sur la même machine.
Koulamas [39] a montré que toute séquence de tâches peut être réduite à une autre qui satisfait la condition de la régularité. Néanmoins, une telle réduction peut engendrer une dégradation de la performance de l'ordonnancement.
Présentons quelques méthodes de fusionnement de tâches pour l'obtention d'un ensemble régulier.

3.4.1. Fusionnement de tâches

Trois méthodes de réduction, y compris celle dûe à Koulamas [39], existent et fournissent un ensemble régulier.

3.4.2 Addition de tâches

Si plusieurs tâches sont de temps d'exécution réduit, leur addition peut être définie pour former une tâche de temps d'exécution la somme des temps d'exécution. Cette opération est répétée jusqu'à l'obtention d'un ensemble régulier. La figure 3.6 illustre cette procédure. Supposons qu'un ensemble de tâches est non régulier. La région de la régularité est définie par $p_j + s_j \geq p_{\max}$. Dans la figure 3.5, les tâches pour lesquelles on a $r \leq p_j + s_j < p_{\max}$ se combinent pour former de nouvelles tâches et que l'ensemble résultant soit régulier.
Pour réduire le temps d'oisiveté des machines, il est recommandé de placer la tâche à plus grand temps d'exécution et qui a attendu le moins, à la dernière position. Ceci résulte que si des tâches sont additionnées, le temps de chargement de cette

nouvelle tâche, somme des tâches, est $s_1 + p_1 + ... + s_{n-1} + p_{n-1} + s_n$ et le temps d'exécution de la dernière tâche est p_n .

Parfois, si le nouveau temps de chargement est supérieure au temps d'exécution de la nouvelle tâche, somme des anciennes tâches, cette nouvelle tâches peut ne pas être dans la zone régulière.

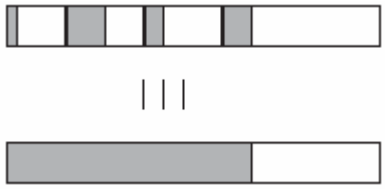


Figure 3.6 Addition de tâches.

3.4.3 Soustraction des tâches

Cette procédure réunit une tâche à temps d'exécution long avec plusieurs tâches courtes comme illustré dans Figure 3.7. On couple la tâche à temps d'exécution long et en déplace à gauche la limite droite de la région régulière de telle façon à obtenir un ensemble des tâches régulières.

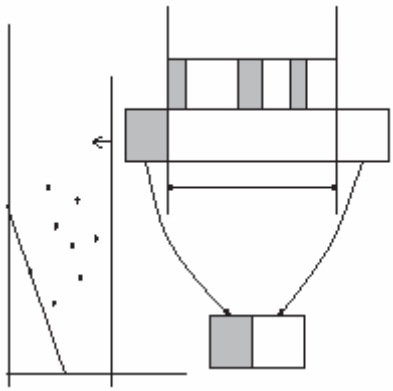


Figure 3.7 Soustraction de tâches.

3.4.4. La réduction de Koulamas

La méthode développée par Koulamas [39], utilise les deux concepts précités. La procédure est présentée comme suit. Cette approche coupe la région générale des deux côtés afin qu'elles forment un ensemble régulier. Sa complexité est en O (n^2).

Algorithme 3.3 : Algorithme de réduction de Koulamas [39].

Début

- **Etape 1:** Ordonner toutes les tâches i, i =1,…, n, dans l'ordre décroissant des pi.
- **Etape 2:** Ordonner toutes les tâches j , j = 1,…, n dans l'ordre croissant de s_j+p_i.

  //Nous devons obtenir deux sous ensembles de tâches.
- **Etape 3: Pour** i=1 **jusqu'a** n **faire**

  Pour j=1 **jusqu'a** n **faire**

  Si $s_j + p_j \leq p_i$ **alors**

  Effacer la tâche j ;

  Poser $p_i = p_i - p_j - s_j$

  Finsi

  Fin pour

  Fin pour

3.5 Implémentation :

Ces deux heuristiques ont été implémentées par nos soins en utilisant un langage orienté objet. Les instances traitées ont les caractéristiques suivantes : numéro de la tâche, son temps de chargement et le temps d'exécution. Toutes les données sont générées aléatoirement en utilisant la fonction Rand. Le programme a été testé avec succès sur un nombre de 2000 tâches. Les durées d'exécution des programmes étaient de l'ordre de quelques secondes. Dans leur article, Abdekhodaee et al. [02] fournissent une étude comparative des deux heuristiques et montrent que l'heuristique Forward HF est meilleure. De ce fait, nous avons volontairement omis de faire l'étude comparative.

CHAPITRE 4 : PRESENTATION d'une NOUVELLE heuristique POUR LE PROBLEME $P2,S1|p_i,s_i|C_{\max}$

Une bibliographie abondante et riche existe pour la résolution des problèmes de flowshop hybrides à 2-étages(voir chapitre 1, paragraphe 4). De nombreuses heuristiques ont été proposées. Elles utilisent souvent des transformations de ce schéma d'atelier pour se ramener au problème du flowshop à 2-machines qui est résolu polynomialement par l'algorithme de Johnson [04]. La figure ci-dessous représente un schéma de flowshop à 2-machines.

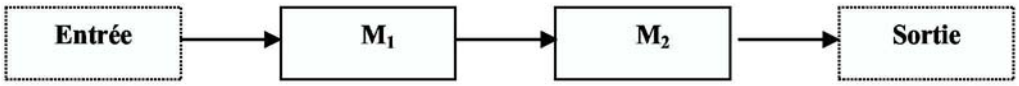


Figure 4.1 : Schéma du flowshop à 2-machines.

Dans la suite, une nouvelle heuristique pour la résolution du problème où une seule machine au premier étage existe appelée serveur et deux machines sont disponibles au second étage est proposée. Elle s'inspire et s'influence des idées de Johnson.

Elle sera présentée, un exemple de son déroulement est fourni, une implémentation par nos soins est réalisée et une comparaison avec l'heuristique Forward de Abdekhodaee et al. [02] est effectuée. Notre heuristique s'avère la plus efficace en fournissant des valeurs de la longueur d'ordonnancement les plus proches de l'optimum. Une borne inférieure fournit par Abdekhodaee est utilisée. Malheureusement cette borne n'a pas été dépassée est reste toujours d'actualité.

Rappelons succinctement l'algorithme de Johnson qui n'est qu'une condition suffisante pour l'obtention d'une solution optimale. Il est efficace et de complexité en O(nlog(n)).

4.1. Algorithme de Johnson :

Un ensemble de n tâches doit être exécuté sur deux machines M_1 et M_2. Une zone tampon ou une aire de stockage entre les machines existe. Chaque tâche doit

passer pour exécution sur la première machine, puis sur la seconde afin de minimiser la longueur d'ordonnancement.

Le job j précédera le job j+1 si Min (A_j,B_{j+1}) < Min (A_{j+1},B_j).

Règle de Johnson : Dans un flow-shop n/2/F/C_{max} où tous les jobs sont disponibles à la même date, la tâche i précède la tâche j optimalement si $Min(t_{i1}, t_{j2}) \leq Min(t_{j1}, t_{i2})$

Cette règle peut aussi se formuler sous le résultat ci-dessous.

Théorème 1 : pour le problème n / 2 / F / C_{max} avec $p_{i1} = a_i$ et $p_{i2} = b_i$, i = 1, …, n.

i) si a_k = min { a_1, …, a_n, b_1, …, b_n }alors il y a un ordonnancement optimal où le job J_k est placé à la première position de l'ordonnancement.

ii) si b_k = min { a_1, …, a_n, b_1, …, b_n }alors il y a un ordonnancement optimal où le job J_k est placé à la dernière position de l'ordonnancement.

La règle de Johnson peut se formuler d'une autre manière. Divisons les jobs en deux sous ensembles. L'ensemble 1 contient tous les jobs tel que $p_{1\,j} < p_{2\,j}$ et l'ensemble deux contient tous les jobs tel que $p_{1\,j} > p_{2\,j}$. S'il y a égalité le job est dans un des deux sous ensembles. On exécute les jobs du premier ensemble dans l'ordre croissant des $p_{1\,j}$ (SPT) et suivent les jobs du second ensemble dans l'ordre décroissant des $p_{2\,j}$ (LPT). L'ordonnancement peut ne pas être unique.

Théorème 2 : Tout ordonnancement SPT(1)-LPT(2) est optimal pour F2//C_{max}.

Une présentation algorithmique de ce résultat peut être obtenue et formulé par :

Algorithme 4.1 (Johnson(1954))

Début

Etape 1 : constituer les groupes de tâches

$U = \{i / p_{i1} \prec p_{i2}\}$

$V = \{i / p_{i1} \geq p_{i2}\}$

Etape 2 : Ordonner U selon l'ordre SPT sur la première machine (P_{i1}) et V selon l'ordre LPT sur la seconde machine (P_{i2}).

Etape 3 : la séquence optimale S* est fournie par la concaténation dans l'ordre des ensembles U et V. (S* = U $\cup$ V)

Fin

Plusieurs heuristiques ont été influencées par l'idée de l'algorithme de Johnson.
Nous vous proposons une nouvelle heuristique appelée SR1 (De Sakri Redha version 1), donnant une solution réalisable en un temps polynomial.

4.2. Heuristique SR1

Naïvement, nous regroupons les deux machines de second étage en une seule deux fois plus
puissante qu'une machine de référence. Ce procédé, nous le schématisons par la figure 4.2.

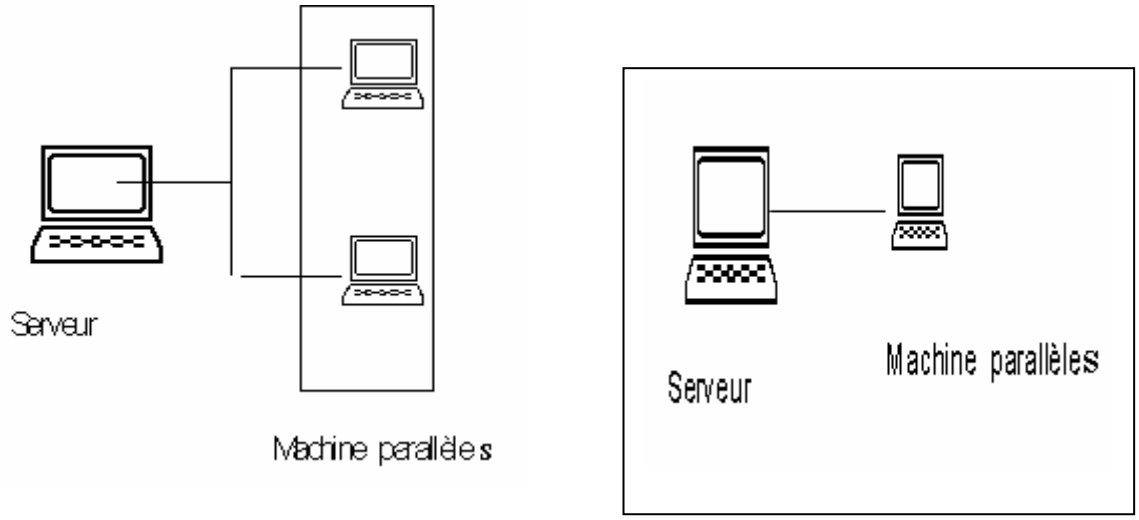


Figure 4.2 : Regroupement des deux machines au second étage.

Le nouveau schéma de droite de la figure 4.1 n'est autre qu'une disposition en flowshop à 2-machines. En appliquant l'algorithme de Johnson, une séquence de tâches optimale est obtenue. La séquence obtenue étant une liste de tâches, elle est exécutée sur le serveur. Sur les machines du second étage les tâches de la dite liste seront exécutée selon la règle d'affectation, la première machine disponible (Règle FAM, First Available Machine).

Cette heuristique SR1 se reformule sous l'écriture suivante :

Algorithme 4.2 :Heuristique SR1

Début

Déclarer s_i le temps de chargement et p_i le temps d'exécution

Toutes les machines sont disponibles à l'instant 0.

//règle de Johnson

Constituer les groupes de tâches

$$U = \{i / s_i \prec p_i\} \text{ et } V = \{i / s_i \geq p_i\}$$

Etape 2 : Ordonner U et V respectivement selon l'ordre croissant des s_i et l'ordre décroissant des p_i pour obtenir des listes partielles C et D.

Etape 3 : En concaténant les listes partielles C et D, S = C ∪ D est la séquence voulu.

//Affecter les tâches de l'ensemble S aux machines parallèles en utilisant la règle FAM

Pour toutes les tâches prises dans l'ordre de la séquence S **faire**

Placer une tâche prête pour exécution sur la machine disponible au plutôt.

Mettre a jour la date de disponibilité de la machine.

Fin pour

Fin

4.3. Finitude et complexité

Comme l'ordonnancement fournit par cette heuristique est réalisable, aucune justification n'est indispensable. Elle est gloutonne où à chaque itération on complète une solution partielle en cherchant à faire le choix le plus avantageux et définitif.

Comme le nombre de tâches est fini, les sous ensembles U et V le sont d'où l'heuristique est finie.

4.4 Exemple de déroulement de l'heuristique SR1

Soit un système informatique formé d'un serveur et deux machines parallèles. Les donnés du tableau sont les s_i , temps de chargement des tâches sur le serveur et p_i les temps d'exécution d'une tâche sur une machine du second étage. Les machines au second étage sont supposées identiques.

Tâche	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Si	1	4	6	3	2	2	10	3	9	6
Pi	13	19	18	12	2	4	3	7	1	12

Tableau 4.1 Des temps d'exécution de tâches.

(1,2,3,4,5,6,7,8,9,10) représente un ensemble de 10 tâches à exécuter.

Le système informatique est identifié selon la méthode vue ci-dessus comme un flowshop à 2-étage.

Par application de l'algorithme de Johnson, les deux sous ensembles sont :

U = (1, 6, 5, 8, 4, 2, 10, 3)

V = (9,7)

La séquence « concaténation » de tâches de l'ensemble U et V est S :=(1,6,5,8,4,2,10,3,9,7)

En appliquant la règle FAM sur cette séquence, on obtient un ordonnancement de longueur égale à 53 unités de temps. Il est représenté par le diagramme de Gantt de ci-dessus .

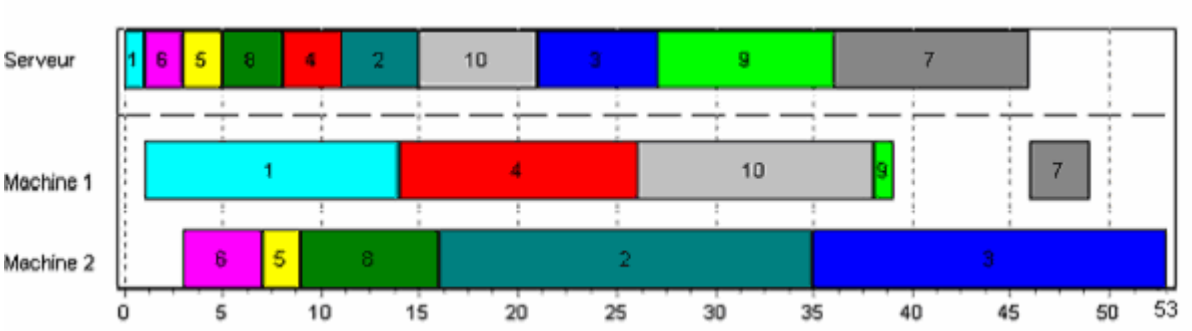


Figure 4.3. Diagramme de Gantt associé à l'heuristique SR1.

Sur le même exemple, l'heuristique *en avant* HF présentée par Abdekhodaee [02] donne un ordonnancement de longueur de 70 unités, représenté par la suivante.

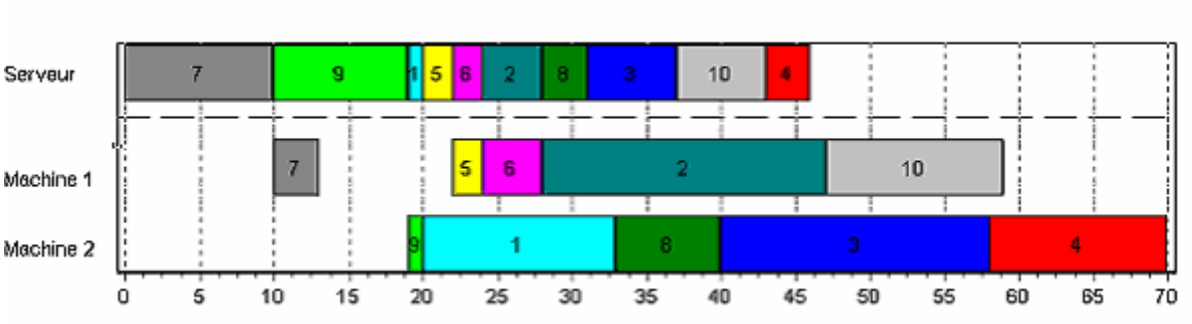


Figure 4.4. Diagramme de Gantt associé à l'heuristique HF.

L'heuristique en *arrière* HB de Abdekhodaee et al.[02], représenté par la figure 4.5., fournit un ordonnancement de longueur 55 unités.

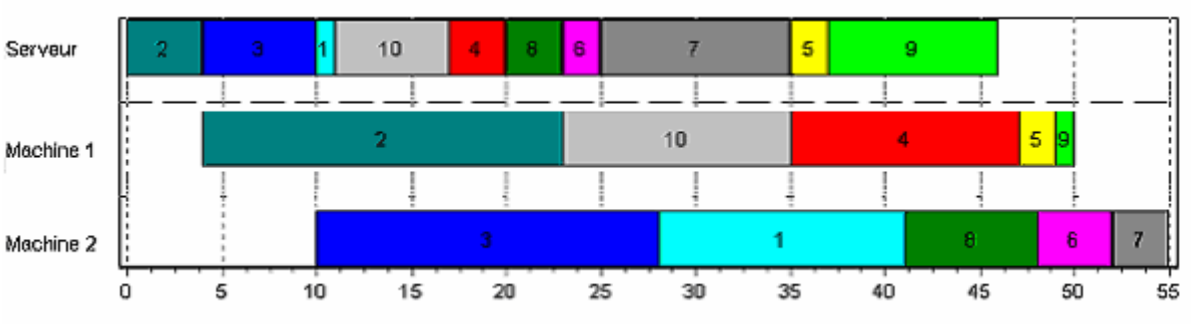


Figure 4.5. Diagramme de Gantt associé à l'heuristique HB.

Pour cet exemple, on remarque que l'heuristique SR1 a fourni un meilleur résultat que celui donné par les deux heuristiques récentes de Abdekhodaee et al. [02].

4.5 Implémentation

Cette heuristique a été implémentée en langage orienté objet sur un ordinateur Pentium 4. L'utilisation des structures de type tableaux pour stocker les différents vecteurs impose au départ une connaissance de leur taille. Le réajustement de la taille d'un tableau et l'insertion de nouveaux éléments est coûteux en temps. L'utilisation des listes chaînées est indispensable. Ce type de structure crée une liste vide, ajouter et retirer des élément (début / fin / milieu) de cette liste et de libérer la mémoire. On stocke dans une liste chaînées les paramètres des sous ensembles *U, V* et l'ensemble S. Pour trier les éléments de l'ensemble B selon l'ordre croissante et décroisant, on utilise un algorithme de « tri rapide », de l'anglais « Quick sort ».

4.6. Comparaison des heuristiques SR1 et HF

Sur un échantillon de 100 tâches, on applique l'heuristique HF et l'heuristique SR1. On calcule l'erreur relative moyenne. Notre programme a été déroulé pour un nombre supérieur à 1000 essais. Une représentation graphique s'obtienne directement par notre programme réalisé en utilisant le menu diagramme. Les résultats fournissent que 70 % des cas l'heuristique SR1 est meilleure.

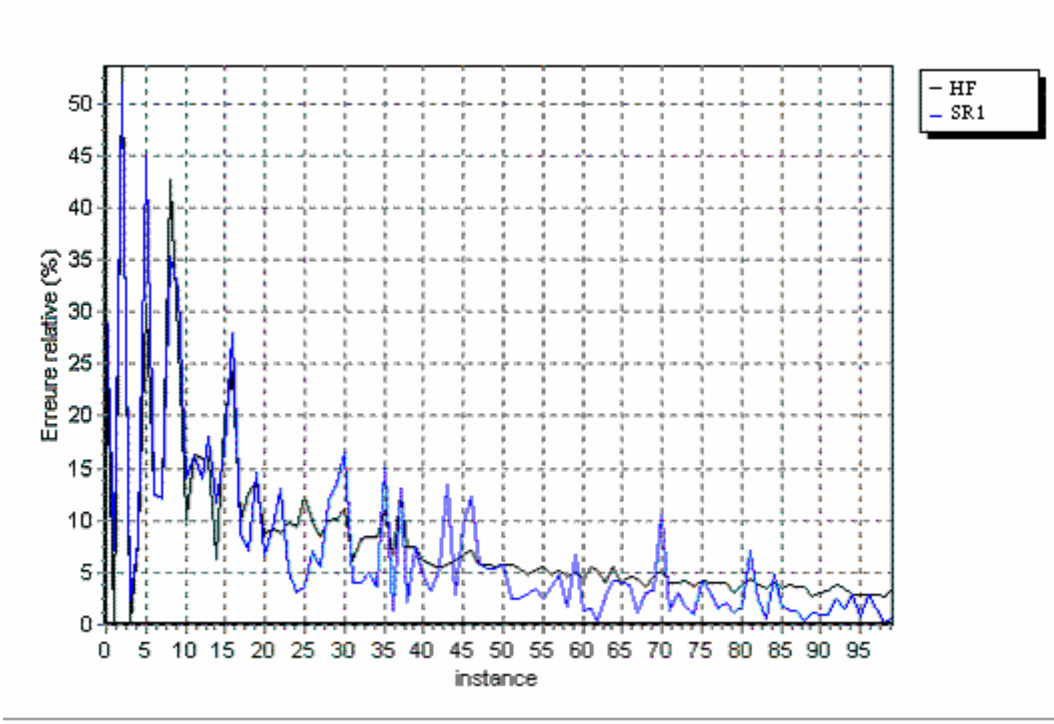


Figure 4.6: Pourcentage de l'erreur relative des deux heuristiques.

Nos données et les résultats obtenus sont présentés dans le tableau 2 de l'annexe B.

Pour chaque échantillon de 1000 instances où le nombre de tâches varie de 50 à 100, pour chaque heuristique, on calcule le nombre de fois, le pourcentage où la meilleure longueur d'ordonnancement est obtenue. Les résultats obtenus sont récapitules dans une aire circulaire.

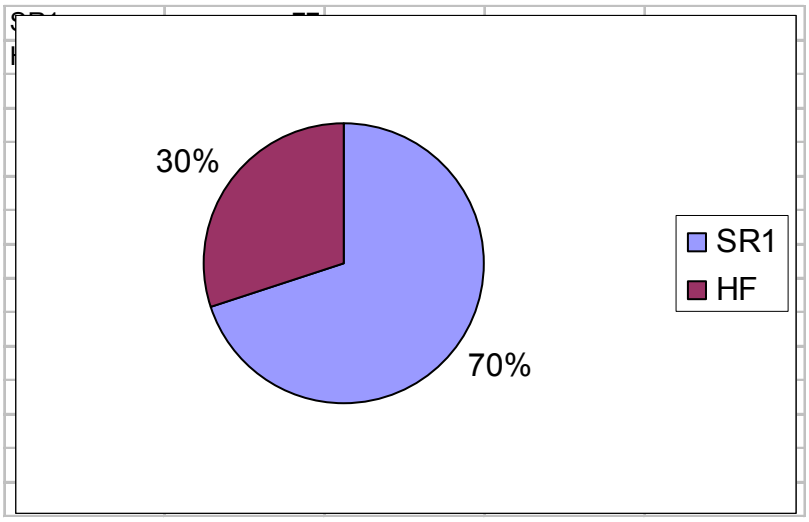


Figure 4.7. Pourcentage de meilleures longueurs d'ordonnancement entre les heuristiques SR1 et HF.

De la figure 4.8 représentant les temps d'exécutions en fonction de nombre des instances de problèmes et de la position de la courbe en rouge située en dessous, l'heuristique SR1 nécessite moins de temps de calcul que l'heuristique HF.

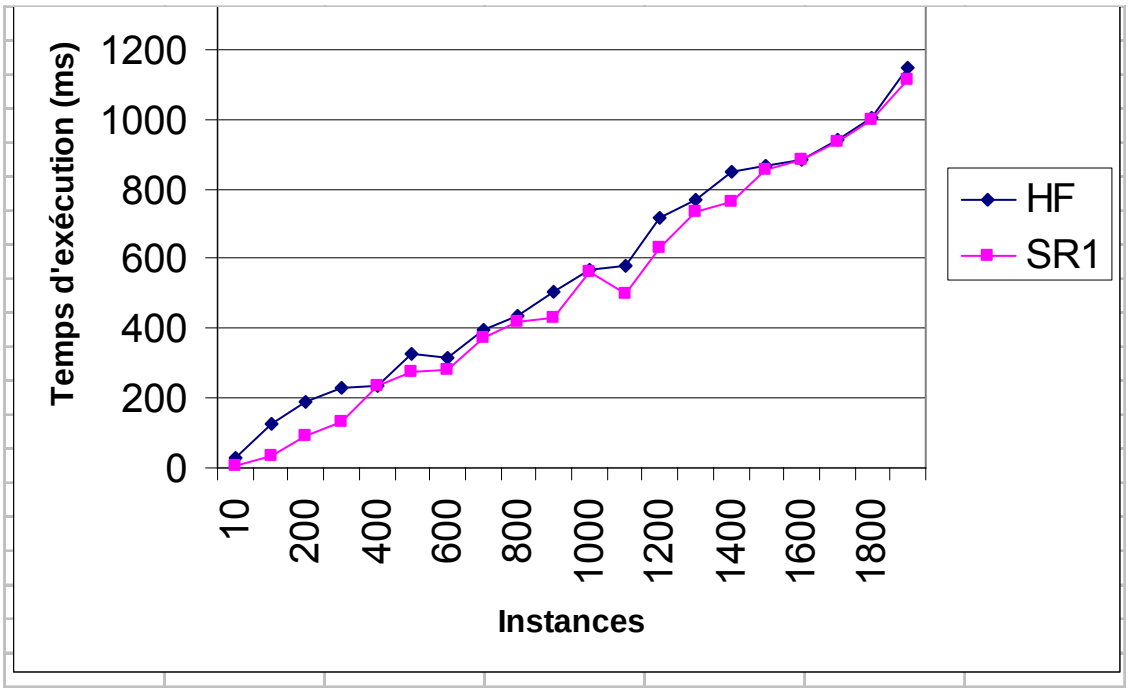


Figure 4.8 : Moyenne des temps CPU en millisecondes

Bien qu'aucune étude analytique n'a été entreprise, cette nouvelle heuristique notée SR1 s'inspire et s'influence des idées de Johnson(1954). Elle a été implémentée par nos soins. Une comparaison graphique ci-dessous avec l'heuristique Forward de Abdekhodaee et al. [02] est effectuée. De son comportement, notre heuristique s'avère asymptotiquement, de l'ordre de 1800 instances, plus efficace en fournissant des valeurs de la longueur d'ordonnancement les plus proches de l'optimum.

CHAPITRE 5

ETUDE COMPARATIVE D'ALGORITHMES DE TYPE LISTE, DE L'HEURISTIQUE SR1 ET DE DEUX VARIANTES D'UN ALGORITHME GENETIQUE

Les problèmes d'ordonnancements sont combinatoires. En général, l'existence d'un ordonnancement optimal ne se pose pas, mais sa caractérisation et sa détermination se posent. Si un problème d'ordonnancement est classé facile, un algorithme polynomial existe dans la littérature et le résoud efficacement. Par contre, s'il est classé difficile et s'il est de taille moyenne, une méthode de résolution exacte de type par séparation et évaluation ou de la programmation dynamique est recommandée. Ces deux méthodes sont généralement énumératives et souvent inefficaces si la taille des données du problème difficile est grande. La détermination de bornes suffisamment fines se pose. Dans ce cas et si on se contentera d'une solution approchée, les métaheuristiques ou heuristiques, qui sont les plus utilisées.

Le problème d'ordonnancement étudié dans ce mémoire $P2,S1|p_i,s_i|C_{\max}$ est prouvé NP-difficile par Abdekhodaee et al. [02]. On propose sa résolution par six algorithmes de type liste, d'une heuristique SR1 qu'on a proposé au chapitre 4 et de deux variantes AG1 et AG2 d'un algorithme génétique.

5.1. Les méthodes de Liste

Les méthodes de liste sont des méthodes itératives où à chaque étape on complète une solution partielle en cherchant à faire le choix le plus judicieux. Ces méthodes permettent d'obtenir une solution rapidement et donne un bon compromis entre la qualité des solutions trouvées et le temps de calcul nécessaire. Le choix effectué à une itération est définitif, on s'interdit de le remettre en cause au cours des étapes ultérieures.

Une liste des tâches est établie en fonction d'un critère de priorité. L'ordonnancement de liste est construit en affectant la tâche prête la plus prioritaire à un processeur libre lui permettant de commencer son exécution au plus tôt.

De Gelineau [56], six règles de priorité sont utilisées pour construire les listes de tâches. Nous notons par p_i la durée de l'exécution de la tâche i et s_i son temps de chargement sur le serveur.

La règle **SPT,** trie les tâches dans l'ordre croissant des durées d'exécutions p_i.

SST : Trie les tâches dans l'ordre croissant des durées de chargements s_i.

SLT: Trie les tâches dans l'ordre croissant de la somme des temps d'exécution et de chargement, $a_i=s_i+p_i$.

LPT : Trie les tâches dans l'ordre décroissant des durées d'exécutions p_i.

LST : Trie dans l'ordre décroissant des durées de chargements s_i.

LLT : Trie dans l'ordre décroissant des longueurs des tâches $a_i=s_i+p_i$.

5.1.1. Affectation des tâches

A un instant t, une tâche T_i peut être dans le système mais non encore prête pour exécution.

A titre d'exemple, si un fichier est prêt pour impression, en l'absence du papier, il ne peut l'être. On cherche à affecter les tâches prêtes à exécution sur les machines selon une liste de priorités indiquée en tenant compte des contraintes de placement. On doit s'assurer qu'il existe une affectation possible pour les tâches prêtes pour exécution.

Une machine est dite disponible pour exécution si elle est libre et prête à exécuter une tâche.

L'affectation des tâches se fera selon la règle FAM [30], affecter une tâche prête à exécution sur la première machine disponible. Les algorithmes de type liste associés à l'une des règles de priorité précédentes et la règle d'affectation FAM ont une complexité O (nlog(n)).

5.1.2. Comparaison entre les méthodes de Liste

Une méthode de liste se caractérise par la règle de priorité adoptée pour la construire. Le but de nos expérimentations a été de tester, d'évaluer et de comparer les 6 listes ci-dessus. Pour chaque une des six listes, l'idéal serait de

pouvoir comparer les résultats donnés par ces listes avec la durée d'un ordonnancement optimal. Dans ce cas l'écart entre les solutions de type Liste et l'optimum ou la distance à l'optimum en pourcentage sera déterminée. Sachant qu'en général la valeur de l'optimum C_{max} est inconnue, une borne inférieure LB de cette valeur optimale sera utilisée.

LB est définie par la plus grande valeur des expressions LB1 et LB2, où

$$LB1 = \sum_{i=1}^{n} s_i + \min_{i=1..n} p_i , \quad LB2 = \sum_{i=1}^{n} (a_i + p_i) + \min_{i=1..n} s_i ,$$

p_i, et s_i sont respectivement la durée d'exécution et la durée de chargements de la tâche i.

Le rapport d'approximation expérimental, une mesure de performance, est donnée par la formule: $\rho(I) = \frac{\left(C_{\max}^{H(I)} - LB(I)\right)}{LB(I)} \times 100$, avec:

- I : un jeu d'essai,
- $C_{\max}^{H(I)}$: Solution obtenue par une méthode H,
- LB (I) : une borne inférieure de la solution optimale.

Une analyse expérimentale sur un échantillon de 100 instances a été menée. Pour chaque une d'elle, on calcule l'erreur relative $\rho(I)$. Les résultats obtenus sont représentés par l'allure graphique par rapport à un repère orthogonal (instance x erreur).

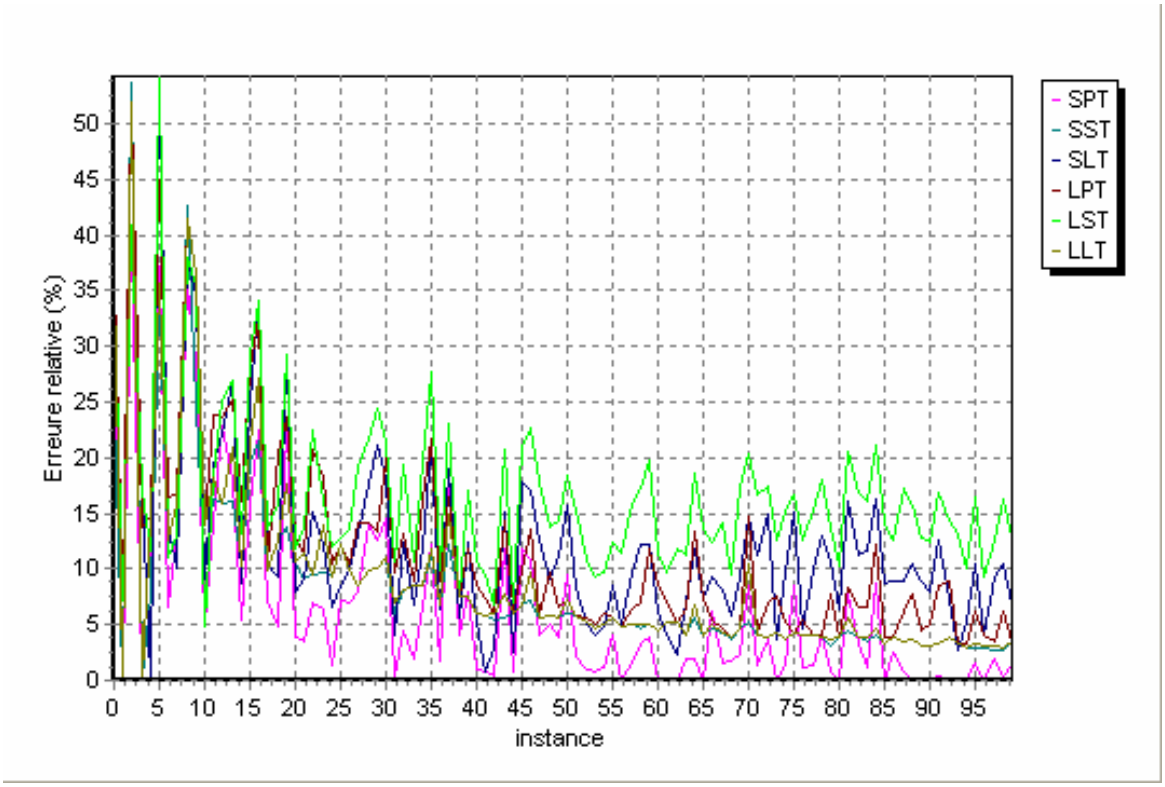


Figure 5.1 : Performance des règles listes associées à une affectation FAM.

De ce comportement graphique, nous remarquons que la liste SPT est située en général en dessous des allures des autres listes. Quelques points du plan, de la liste SPT sont entre les graphes. De ce fait, sur chaque instance de l'échantillon, on calcule le nombre de fois au la meilleure longueur est obtenue pour chaque règle de liste. Les résultats sont donnés dans le diagramme de performance.

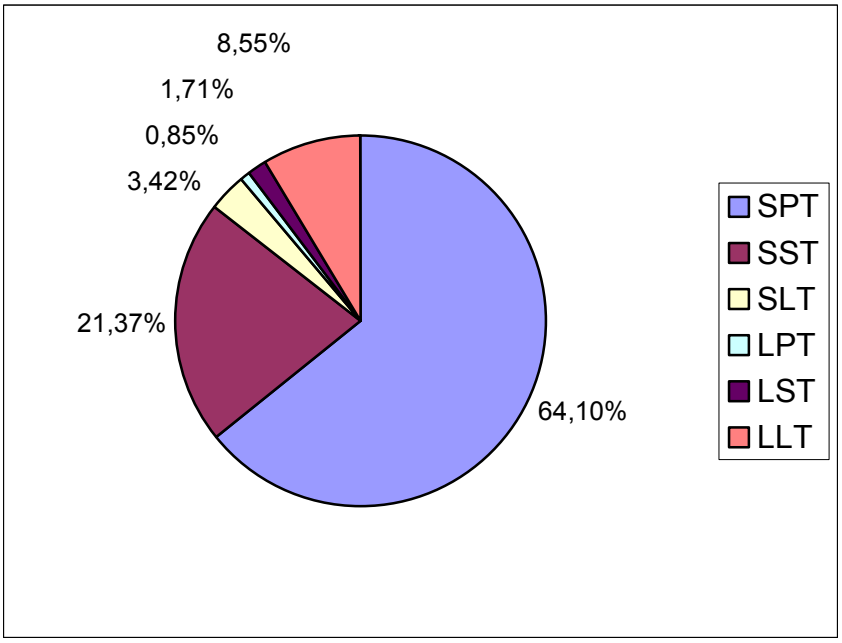


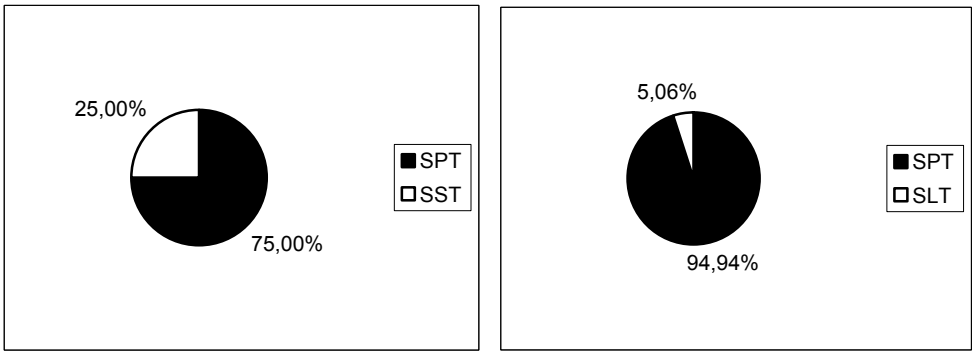
Figure 5.2. Performance des six listes en pourcentage.

La règle SPT réalise 64,10 % de meilleurs cas. Les erreurs fournies par la règle LLT varient dans l'intervalle [5 %,10 %]. Les règles SLT, LPT et SST sont moins performantes pour le problème $P2,S1|p_i,s_i|C_{\max}$.

Rarement la règle LPT donne une solution plus proche de la borne indiquée, de l'ordre de 0,85 % . Le pourcentage où les règles SST et SLT sont efficaces est respectivement de 1,71 % et 3,42 %.

Pour la résolution de notre problème particulier du flowshop hybride $P2,S1|p_i,s_i|C_{\max}$, nous confirmons graphiquement que la règle SPT est meilleure parmi les six listes.

Cette performance peut être représentée par les diagrammes ci-dessous.



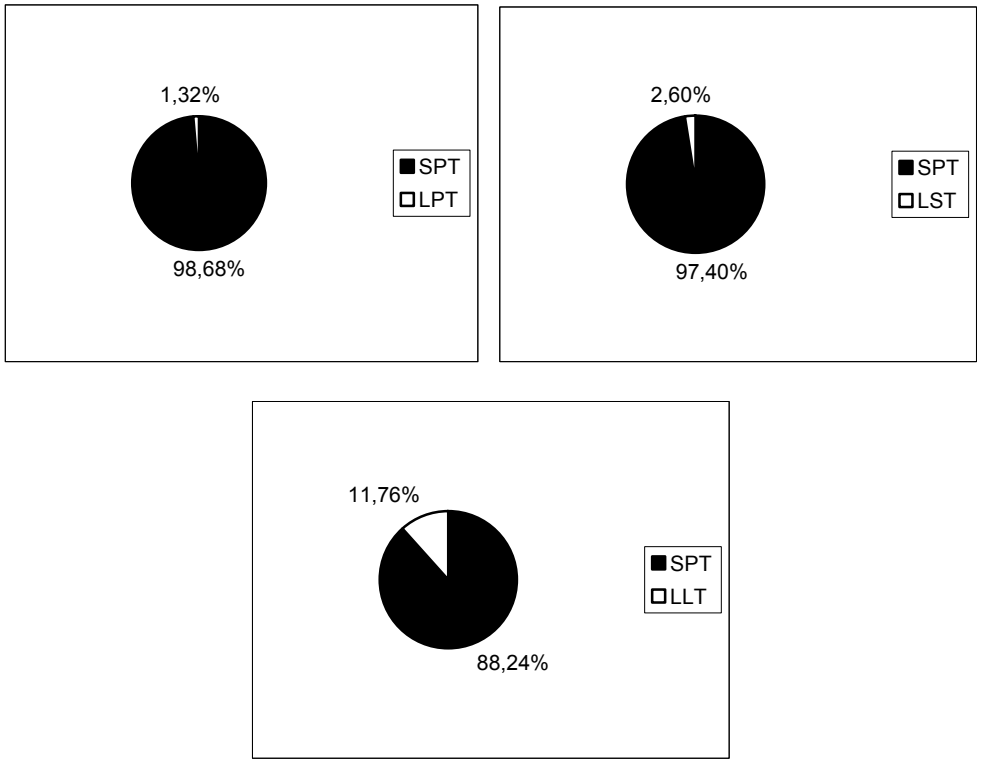


Figure 5.3 : Performance de SPT par rapport à chacune des 5 autres listes.

5.2. Description et Implémentation d'un AG

Cette classe de méthodes a été inventée par Holland dans les années soixante, pour imiter les phénomènes d'adaptation des êtres vivants. L'application aux problèmes d'optimisation a été développée ensuite par Goldberg. Par analogie avec la reproduction des êtres vivants, on part d'une population initiale de N ordonnancements réalisables. Il faut coder chaque solution, un ordonnancement réalisable, comme une chaîne de caractères par analogie à un chromosome d'une cellule vivante. Le chromosome est formé de sous chaînes appelées " gènes ", chacun codant une caractéristique de la solution.

Une itération est appelée " génération ". A chaque génération, on choisit au hasard NC paires de chromosomes à reproduire, NC < N, avec une probabilité croissante de leur adaptation. Chaque paire d'ordonnancements (x, y) choisie subit une opération de "croisement" (cross-over). Un gène est choisi aléatoirement dans x, puis permuté avec le gène de même position dans y. On pratique également un faible " taux de mutation " : NM chromosomes sont choisis et subissent une modification

aléatoire d'un gène. La nouvelle population est formée de la population précédente et des chromosomes nouveaux générés par mutation ou croisement.

On définit pour chaque solution une " mesure d'adaptation au milieu " appelée " fitness ". Pour l'ordonnancement, cette fonction est sa longueur. On élimine alors les solutions les moins adaptées pour maintenir un effectif constant de N solutions. On recommence le même processus à l'itération suivante. L'intérêt est que les bonnes solutions sont encouragées à échanger par croisement leurs caractéristiques et à engendrer des solutions encore meilleures. De plus, pour un POC, problème d'optimisation combinatoire, les solutions finales vont être concentrées autour des minima locaux, et la méthode fournira un choix de plusieurs solutions possibles au décideur.

Notons G la fonction d'évaluation choisie. Pour engendre la génération $X^{(n+1)}$ de solutions à partir de la population courante $X^{(n)}$, on applique le schéma décrit ci dessous.

Sélectionner dans $X^{(n)}$ un ensemble de paires de solutions de hautes qualités.

Appliquer à chacune des paires de solutions sélectionnées un opérateur de croisement qui produit une ou plusieurs solutions " enfants ".

Remplacer une partie de $X^{(n)}$ formée de solutions de basse qualité par des solutions " enfants " de bonne qualité;

La population $X^{(n+1)}$ est obtenue après avoir appliqué un opérateur de mutation aux solutions ainsi obtenues.

Algorithme 5.1 (Algorithme Génétique en général)

Initialisation : soit $X^{(0)} \subseteq X$, une population initiale;

Etape n : soit $X^{(n)} \subseteq X$, la population courante;

- sélectionner dans $X^{(n)}$ un ensemble de paires de solutions da haute qualité;
- appliquer à chacune des paires de solutions sélectionnées un opérateur de croisement qui produit une ou plusieurs solutions " enfants ".
- remplacer une partie de $X^{(n)}$ formée de solutions de basse qualité par des solutions enfants de haute qualité;
- appliquer un opérateur de mutation aux solutions ainsi obtenues; les solutions éventuellement mutées constituent la population $X^{(n+1)}$;

Si la règle d'arrêt est satisfaite, stop;

Sinon, passer à l'étape n + 1.

5.2.1 Codage

Un ordonnancement est fournit par la liste des tâches qui le compose. Ce codage de l'ordonnancement est indirect. Les numéros associés aux tâches ont été utilisés. Nous confondons ordonnancement et liste selon l'ordonnancement. La permutation de tâches représentant la liste est le chromosome, il représente la solution. Une population sera donc un ensemble de permutations de « n » tâches.

5.2.2 La population initiale

Deux approches pour construire la population initiale ont été utilisées :

- Pour la variante GA1 : on génère aléatoirement des permutations réalisables.
- Pour la variante GA2 : les permutations de tâches sont construites selon les six règles SPT, SST, SLT, LPT, LST, LLT et les heuristique SR1, HF et HB.

5.2.3 La sélection

La sélection aussi bien celle des individus de " haute qualité " que celle des individus de " basse qualité ", comporte généralement un aspect aléatoire. Chaque individu x_i de la population parmi laquelle se fait la sélection se voit attribuer une probabilité p_i d'être choisi d'autant plus grande que son évaluation est haute (basse dans le cas d'une sélection de " mauvais individus "). On tire un nombre " r " au hasard (uniformément sur [0, 1]). L'individu " k " est choisi tel que : $\sum_{i=1}^{k-1} p_i < r_i \leq \sum_{i=1}^{k} p_i$.

La probabilité que x_k soit choisi est aussi bien égale à p_k. Cette procédure est appelée " la roulette russe ". Elle est itérée jusqu'à ce que l'obtention d'une population de taille désirée.

5.2.4 Le croisement

Soit deux solutions x et y sélectionnées parmi les solutions de haute qualité. Un opérateur de croisement (crossover) fabrique une ou deux nouvelles solutions x' et y' en combinant x et y.

Si x et y sont deux vecteurs de 0 et 1, un opérateur de croisement classique (two-point crossover) consiste à sélectionner aléatoirement deux positions dans les

vecteurs et à permuter les séquences de 0 et 1 figurant entre ces deux positions dans les deux vecteurs.

Pour des vecteurs x = 0 1 1 0 1 1 0 0 et y = 1 1 0 0 1 0 1 0, si les positions " après 2 " et " après 5 " sont choisies, on obtient après croisement :

x' = 0 1 | 001 | 100 et y' = 1 1 | 1 0 1 | 0 1 0.

De nombreuses variantes d'un tel opérateur peuvent être imaginées. Ces variantes doivent être adaptées au codage des solutions et favoriser la transmission des " bonnes sous-structures " des solutions parents aux enfants. Pour le problème d'ordonnancement avec le codage liste des tâches, le two-point crossover ne convient pas.

Si x = A B C D E F G H et y = B E F H A D G C, en croisant entre les positions " après 2 " et " après 5 ", on obtiendrait : x' = A B | F H A | F G H et y' = B E | C D E | D G C qui ne sont pas des permutations.

Nous avons utilisé un opérateur de croisement spécialement conçu pour les listes données, le OX-crossover. Les deux solutions parentes sont " préparées " avant l'échange des séquences situées entre les deux positions choisies au hasard.

Dans cet exemple de tâches, la zone d'échange de x est préparée à accueillir la séquence des tâches F, H, A de y. Pour ce faire, on remplace chacune des tâches F, H et A dans le vecteur x par une place vide symbolisée par une *, soit * B | C D E |* G * et en commençant à la droite de la zone d'échange, on tasse les tâches restantes de la permutation dans l'ordre de la permutation x en oubliant les *, ce qui donne : D E | * * *| G B C. Les * se retrouvent dès lors dans la zone d'échange, alors que l'ordre de parcours des autres tâches n'a pas été changé. On procède de même pour y : B * | F H A |* G * devient H A | * * *| G B F.

On procède alors à l'échange des séquences, ce qui donne deux permutations enfants x' et y' :

x' = D E | F H A | G B C et y' = H A | C D E |G B F.

Un certain nombre de paires d'enfants sont ainsi générés et remplacent une partie des parents choisis parmi les moins performants.

5.2.5. La Mutation

Une mutation est une perturbation introduite pour modifier une solution individuelle, par exemple la transformation d'un 0 en un 1 ou inversement dans un vecteur binaire.

Dans l'ordonnancement, une mutation peut correspondre à une *transposition* de deux tâches. En général, on décide de muter une solution avec une probabilité assez faible de 0,1. Le but de la mutation est d'introduire un élément de diversification et d'innovation.

5.2.6 Critère d'arrêt

Nous avons choisi d'arrêter les algorithmes AG1 et AG2 lorsque le nombre total d'itérations MaxIter atteint une valeur fixée à 2000. Cette valeur ne doit pas être très grande, de l'ordre de 10.000. Dans ce cas, l'AG génère une grande partie de l'espace des solutions réalisables et diminue de son efficacité. Inversement, ce nombre d'itérations ne doit pas être réduit. L'AG n'aura pas suffisamment de temps d'améliorer la population en cours. On peut imposer un second test d'arrêt si des phénomènes de cycle apparaissent, mais ils sont rares.

5.3. Analyse expérimentale de l'AG

Le temps d'exécution, le temps de chargement et l'indice d'une tâche et le nombre de tâches constituent les paramètres d'un problème d'ordonnancement. Ils déterminent une instance ou configuration d'un problème. Un jeu de données ou jeu de tests est un ensemble d'instances.

Dans le but d'évaluer les six algorithmes de type Liste, les deux heuristiques HF et HB de Abdekhodaee et al.(2006), l'heuristique SR1 et les deux variantes AG1 et AG2 de l'algorithme génétique, nous avons considéré des échantillons de 100 instances générés aléatoirement. Les temps d'exécution p_i et de chargement s_i sont distribués uniformément sur l'intervalle [1,100] et le nombre de tâches est supposé prendre les valeurs discrètes multiples de 10 (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250 ou 300). Les programmes réalisés ont été déroulés sur un processeur Intel Pentium IV cadencé à 1.8 GHz et de 256 Mo de RAM.

La variante AG2 consiste à prendre les séquences de tâches fournies par les six de listes et les trois heuristiques comme population initiale. La longueur d'ordonnancement est plus court et s'obtienne en un temps plus rapide que celle donné par la variante AG1, où la population est générée aléatoirement. Une

comparaison graphique de AG1 et AG2 a pu être réalisée. L'allure graphique de AG1 toujours au dessus de celle de AG2 interprète l'efficacité de cette dernière variante.

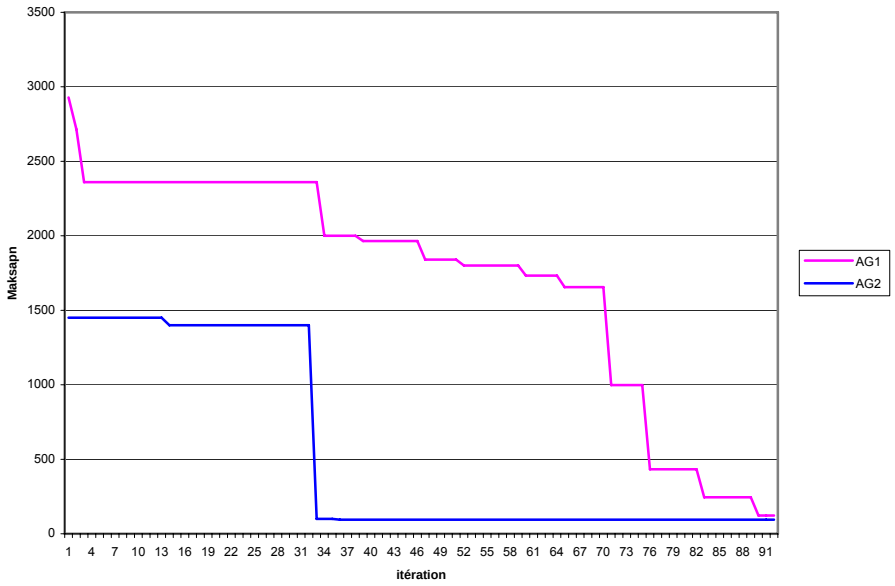


Figure 5.4 : Comportement des deux variantes AG1 et AG2.

5.4. Performance et Comparaison des listes ,des heuristiques et des deux variantes de l'AG

Le critère de comparaison est le temps de réponse ou de calcul donné par chacune des méthodes citées. Ayant pu comparer aux sous paragraphes 5.1.2 et 5.2.7 respectivement les six listes et les deux variantes AG1 et AG2, il est temps d'obtenir une comparaison globale des trois heuristiques HB, HF, SR1, les six listes et AG1 et AG2.

La règle SPT est la meilleure des listes. De même l'heuristique HB est plus performante que HF. Nous omettons de reproduire les résultats donnés par les cinq autres listes et cette dernière heuristique. Nous avons remarqué que les déroulements de AG1 et AG2 peuvent prendre des temps relativement plus grand que celui des trois heuristiques HB, HF, SR1.

Nous reproduisons pour des jeux d'essais à 2-machines au second étage, les temps de réponse des dix méthodes utilisées. Ils sont récapitulés dans le tableau ci-dessous.

Méthode / Nbr Tâches	SPT	HF	SR1	AG1	AG2
100	0,001	0,002	0,001	0,39	0,82
200	0,001	0,002	0,001	1,76	3,57
300	0,06	0,05	0,03	5,26	11,71
400	0,05	0,07	0,04	14,23	28,46
500	0,16	0,13	0,17	28,85	56,69
800	0,5	0,6	0,4	96,66	163,04

Tableaux 4.2 : Temps de calcul en fonction du nombre de tâches.

Pour un ensemble de 1000 tâches, les méthodes de liste et les heuristiques fournissent de bons résultats en moins de 1 seconde. AG1 et AG2 n'améliorent que de 4% la solution courante mais à un prix de 10 minutes de calculs. Du chapitre précédent l'heuristique HF a été montrée plus efficace que celle de HB qui sera omise de cette étude comparative. Pour des raisons de lisibilité et de visibilité du graphe, la règle SPT qui a été montré aussi la meilleure est considérée comme l'algorithme de type liste de référence.

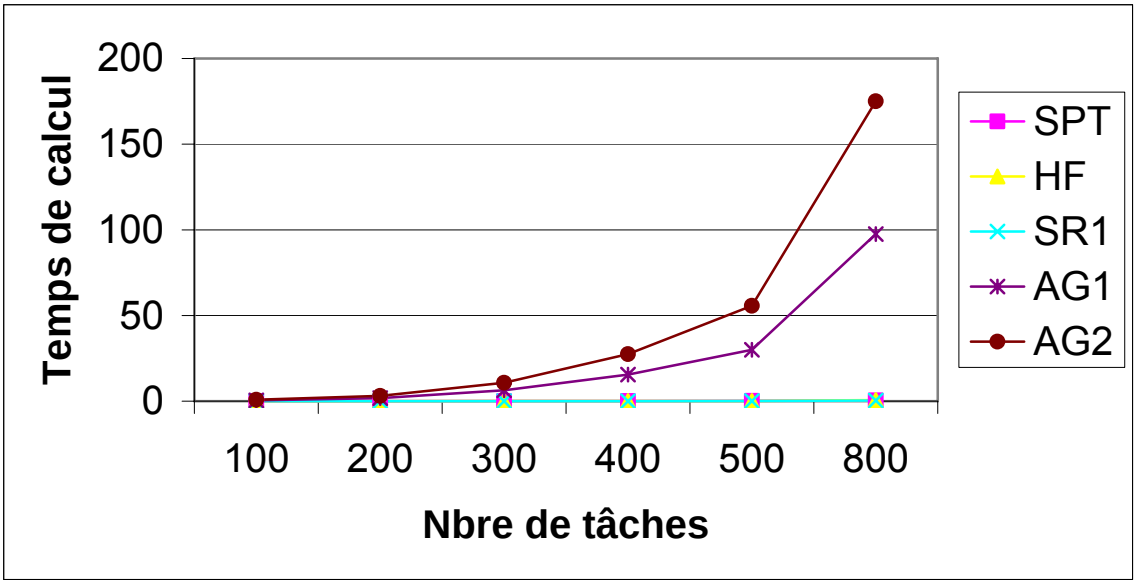


Figure 5.5 : Graphe comparatif des méthodes.

Graphiquement, si le nombre de tâches est inférieur à 200, toutes les méthodes cités ont des temps de réponse, le temps de calcul, presque identiques. Dans ce cas aucune de méthodes n'est à préférer. Si le nombre de tâches avoisine 800, les deux AG prennent des temps démesurés. Les heuristiques fournissent des temps de réponse instantanés.

On a réalisé de nombreuses expérimentations sur des données associées aux tâches générées aléatoirement. Les différentes méthodes présentées ont été comparées et l'influence des paramètres de ces méthodes a été mesurée. Les heuristiques fournissent des ordonnancements réalisables, proches des valeurs des bornes inférieures fournies par la littérature. On peut améliorer ces valeurs des heuristiques, en lançant et en déroulant la version AG2.Une remarque est à faire. Le temps de calcul nécessaire peut être assez important, surtout pour des applications à grand nombre de tâches. Nous avons confirmé que l'heuristique SR1, proposée par nos soins, et la méthode SPT associe à une règle d'affectation FAM, sont les mieux adaptées au calcul rapide d'un ordonnancement réalisable.

CONCLUSION GENERALE ET PERSPECTIVES

L'emploi, beaucoup plus qu'un concept économique, s'affirme aujourd'hui dans sa dimension sociale et psychologique. Le progrès technique, l'emploi, ces deux notions au contenu chargé d'affectivité sont au centre des contradictions que vivent les pays industrialisés. Des théories économiques analysent et évaluent l'acquisition des nouvelles technologies, associées à un processus dynamique. Le progrès technique, que nul ne refuse, apparaît confusément à beaucoup comme la source de bien des dérèglements et en particulier du chômage. Il est élément d'une stratégie industrielle, un programme de recherche afin de relever un défi. Les nouvelles technologies de l'information sont utilisées dans les entreprises, pour améliorer la souplesse de l'outil de production et lutter contre les contraintes multiples qui pèsent sur les économies avancées. Les nouvelles technologies imposent de nouveaux détours de production qui correspondent à des entreprises aux compétences nouvelles. Des industries informationnelles émergent dans les secteurs de l'industrie manufacturière. L'introduction de machines est indiquée pour l'accroissement du revenu. Le problème d'ordonnancement étudié dans ce mémoire, $P2,S1|p_i,s_i|C_{\max}$, est prouvé NP-difficile. Il se modélise comme un flowshop hybride à 2-étages à un seul serveur. On propose sa résolution par six algorithmes de type liste, d'une heuristique SR1 qu'on a proposé et de deux variantes AG1 et AG2 d'un algorithme génétique.

Pour ce qui est des perspectives possibles pour ce travail, nous envisageons d'intégrer des bornes plus fines afin de fournir un encadrement de la valeur optimale de la longueur d'ordonnancement. Ces bornes sont importantes pour le développement de méthodes de recherche par séparation et évaluation. Le cas ou en a plus qu'un serveur au premier étage peut être l'objet d'une étude ultérieure. Il serait intéressant de mener une étude complémentaire dans le cas où la fonction objective est le flowtime, le flowtime moyen, le nombre de tâches en retard etc.…

ANNEXE A :
PREUVE DE L'OPTIMALITÉ DE L'ALGORITHME DE GHIRCHOUN

Lemme 1: Pour tout ordonnancement, nous avons:

$\forall$ k,m$\leq$ k$\leq$ n :

$$\sum_{v=1}^{m} T_v + \sum_{i=1}^{n} P_{[i],2} \leq C_{[k-m+1]} + C_{[k-m+2]} + \ldots + C_{[k]}$$

où $P_{[i],2}$ représente le temps d'exécution au deuxième étage du travail en position i dans C.

Preuve

Si k = m, montrons que $T_1+T_2+..+T_{m-1}+Tm+P_{[1],2}+P_{[2],2}+...+P_{[m-1],2}+P_{[m],2}\leq C_{[1]} + C_{[2]}+...+C_{[m-1]}+C_{[m]}$.

Nous avons Ci = $C_{i,1}$ + $P_{i,2}$ pour tout 1 $\leq$ i $\leq$ m (conséquence de la contrainte no-wait). Renommons les k travaux tels que $C_{1,1}\leq C_{2,1}\leq \ldots \leq C_{k,1}$. Étant donné que $P_{i,1}$ = 1, $\forall$ i =1, ... ,k nous avons:

$C_{i,l} \geq Ti$, $\forall$ i = 1, ... ,k

donc

$Ci = C_{i,1}+ P_{i,2}\geq T_i + P_{i,2}$, $\forall$ i =1, ... ,k

$$\Rightarrow \sum_{i=1}^{k} C_i \geq \sum_{l=1}^{k} T_l + \sum_{i=1}^{k} p_{[i],2}$$

$$\sum_{i=1}^{k} C_i = \sum_{i=1}^{k} C_{[i]} \text{ et } \sum_{i=1}^{k} p_{i,2} = \sum_{i=1}^{k} p_{[i],2}$$

$$\Rightarrow \sum_{i=1}^{k} C_{[i]} \geq \sum_{l=1}^{m} T_l + \sum_{i=1}^{k} P_{[i],2}$$

Supposons que le lemme soit vrai pour l = n. Montrons qu'il l'est aussi pour n + 1.

Si le travail en position n + 1 dans C est sur la même machine que le travail en position n dans C, nous avons: $C_{[n+1]} = C_{[n]} + p_{[n+1],2}$

$$\Rightarrow C_{[n+1]} \geq C_{[n]} + p_{[n+1],2}$$

$$\sum_{l=1}^{m} T_l + \sum_{i=1}^{n} p_{[i],2} \leq C_{[n]} + \ldots + C_{[n-m+1]}$$

$$\sum_{l=1}^{m} T_l + \sum_{i=1}^{n} p_{[i],2} \le C_{[n]} + \ldots + C_{[n-m+2]} + C_{n+1} - p_{[n+1],2}$$

$$\Rightarrow \sum_{l=1}^{m} T_l + \sum_{i=1}^{n+1} p_{[i],2} \le C_{[n+1]} + C_{[n]} + \ldots + C_{[n-m+2]}$$

Supposons que le travail en position n + 1 dans C ne suive pas le travail en position n dans C mais qu'il s'exécute sur la même machine que le travail en position k dans C avec n - m + 1 ≤ k ≤ n -1, nous avons:

$C_{[n+1]} = C_{[k]} + P_{[nH],2}$

$\Rightarrow C_{[n+1]} > C_{[n-m+1]} + p_{[n+1],2}$

$$\sum_{l=1}^{m} T_l + \sum_{i=1}^{n} p_{[i],2} \le C_{[n]} + \ldots + C_{[n-m+1]}$$

$$\sum_{l=1}^{m} T_l + \sum_{i=1}^{n} P_{(i),2} \le C_{[n]} + \ldots + C_{[n-m+2]} + C_{[n+1]} - P_{[n+1],2}$$

$$\Rightarrow \sum_{l=1}^{m} T_l + \sum_{i=1}^{n} p_{[i],2} \le C_{[n+1]} + C_{[n]} + \ldots + C_{[n-m+2]}$$

Théorème L'algorithme du tableau 2 retourne une solution optimale en O(n log(n)) pour le problème

$$FH2.(1.Pm)nourrait, p_{i,1} = 1 \left| \sum C_i \right.$$

Preuve

Dans l'ensemble B, les travaux sont triés dans un ordre non décroissant de leur temps d'exécution au deuxième étage. Soit $\alpha = |\{i | p_{i,2} = m\}|$

Selon l'algorithme du tableau 2, durant les α premières étapes, tous les travaux i avec $P_{i,2}$ = m sont ordonnancés sur M_1, M_2, M_3, ... , Mm alternativement. Ainsi, à une étape arbitraire v ≤α, l'ordonnancement partiel a un profil (v,v +1,v + 2, ... ,v +m) et le serveur (au premier étage) exécute les travaux sans temps mort dans l'intervalle [0,v].

Après l'étape α, les travaux dans A ne sont pas ordonnancés. A l'étape (α + 1), l'algorithme place le premier travail l_1 de B avec $pe_{11},2 > m$, dans l'intervalle [α + 1+α + $pe_1,2$ + 1] et l'ordonnancement partiel a un profil du type (ε,ε+ 1, ε+ 2, ... , ε+m -l, ε'_1) D avec $\varepsilon'_1 \geq \varepsilon'_1$ + m.

A l'étape (α + 2), l'algorithme du tableau 2 place le plus petit travail compatible l_2 de A ou le premier travail l_2 de B dans [α+2, $\alpha+p_{e2}$,2+2] et réitère avec les travaux de A ou B jusqu'à ce que l'ordonnancement partiel ait un profil du type type (ε,ε+ 1, …..,$\varepsilon'_{1,}$….., ε'_{m-1}) avec $\varepsilon'_1 \geq \varepsilon+i+1$ ($\varepsilon'_1 \leq \varepsilon'_2 \leq \ldots\ldots \leq \varepsilon'_{m-1}$) et le serveur exécute les travaux sans temps mort dans l'intervalle [0, ε]. Alors, le processus est réitéré jusqu'à ce que l'ensemble A ou l'ensemble B soit vide.

Supposons que l'ensemble B soit vide. Alors, l'algorithme du tableau 2 place les travaux de A dans un ordre non décroissant de leur $P_{i,2}$ consécutivement sur la machine disponible au plus tôt. Comme il n'v a pas de temps mort sur le serveur et comme $\sum_{i=1}^{n} C_{i,2} = \sum_{i=1}^{n} C_{i,1} + cte$

L'ordonnancement optimal

Supposons que l'ensemble A soit vide. A cet instant, à l'itération E, le profil de l'ordonnancement partiel est (ε,ε+1, ε'_1, ε'_2 ,…. ε'_{m-1}) ,avec $\varepsilon'_i \geq \varepsilon+i+1$. Alors, les travaux restants de B sont ordonnancés selon l'algorithme SPT /FAM. Nous notons γ (le nombre de travaux non ordonnancés à l'étape ε, i.e. $\gamma=n-\varepsilon$, γ est pair ou impair et soit γ=mk+h avec h$\in${0,m-1}.

Nous avons

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{\in+h} C_{[i]} + \sum_{i=\in+h+1}^{n} C_{[i]}$$

Considérons les premiers ε+h

$$\sum_{i=1}^{\in+h} C_{[i]} = \sum_{i=1}^{\in+h} C_{[i],1} + \sum_{i=1}^{\in+h} p_{[i],2}$$

La valeur $\sum_{i=}^{\in+h} C_{[i],1}$ ne peut pas être réduite étant donné que le serveur travaille sans temps mort dans l'intervalle [0, ε+h].

Donc,

$$\sum_{i=1}^{\in+h} C_{[i],1} = p_{[1],1} + p_{[2],1} + \ldots + p_{[\in+h],1}$$

$$\sum_{i=1}^{\in+h} C_{[i],1} = \frac{(\in+h)}{2}\frac{(\in+h+1)}{2} = K$$

$$\Rightarrow \sum_{i=1}^{\in+h} C_{[i]} = K + \sum_{i=1}^{\in+h} p_{[i],2}$$

La valeur $\sum_{i=1}^{\in+h} p_{[i],2}$ ne peut pas être réduite puisque la somme contient les ε + h plus petits $p_{i,2}$ donc $\sum_{i=1}^{\in+h} C_{[i]}$ est minimum.

Considérons la seconde partie de l'ordonnancement. Comme les travaux sont ordonnancés selon l'algorithme SPT /FAM, ils sont affectés alternativement sur M_1, M_2,…. et M_m. Donc, il n'y a pas de temps mort sur les machines M_j avec $1 \leq k \leq m$ et nous avons:

$$C_{[n]} + C_{[n-1]} + ... + C_{[n-m+1]} = \sum_{k=1}^{m} T_k + \sum_{i=1}^{n} p_{[i],2}$$

$$C_{[n]} = C_{[n-m]} + p_{[n],2}$$

$$C_{[n-1]} = C_{[n-m-1]} + p_{[n-1],2}$$

$$\Rightarrow C_{[n-m]} + ... + C_{[n-2m+1]} = \sum_{k=1}^{m} T_k + \sum_{i=1}^{n-m} p_{[i],2}$$

$$\Rightarrow C_{[L]} + C_{[l-1]} + ... + C_{[l-m+1]} = \sum_{k=1}^{m} T_k + \sum_{i=1}^{l} p_{[i],2} \quad (1)$$

Avec n-mk+m≤l≤n

Le lemme 1 indique que $\sum_{k=1}^{m} T_k + \sum_{i=1}^{l} p_{[i],2}$ est une borne inférieure pour $C_{[l]}$+ $C_{[l-1]}$+….+ $C_{[l-m+1]}$,∀ 1≤ l≤n

Donc, nous en déduisons que

$$C_{[\in+h+m]} + ... + C_{[\in+h+1]} \geq \sum_{k=1}^{m} T_K + \sum_{i=1}^{\in+h+m} p_{[i],2}$$

$$C_{[n]} + C_{[n-1]} + ... + C_{[n-m+1]} \geq \sum_{k=1}^{m} T_K + \sum_{i=1}^{n} p_{[i],2}$$

Soit

$$LB = \frac{n-(\in+h)}{m} \times \sum_{k=1}^{m} T_k + \sum_{w=0}^{\frac{1}{m}(n-z)} \sum_{i=1}^{z+mw} p_{[i],2}$$

Avec

z=ε + h+m

Comme nous avons l'équation (1), l'ordonnancement obtenu avec l'algorithme du

tableau 2 pour les travaux en position z=ε + h+1 à n est tel que $\sum_{i=\epsilon+h+1}^{n} C_{(i)} = LB$, et donc est optimal.

Pour tout ordonnancementσ,

$$\sum_{i=1}^{n} C_i(\sigma) = \sum_{i=1}^{\epsilon+h} C_i(\sigma) + \sum_{i=\epsilon+h+1}^{n} C_i(\sigma)$$

Pour l'ordonnancement σ^a obtenu avec l'algorithme du tableau 2

$$\sum_{i=1}^{n} C_i(\sigma^a) = \sum_{i=1}^{\epsilon+h} C_i(\sigma^a) + \sum_{i=\epsilon+h+1}^{n} C_i(\sigma^a)$$

Nous avons montré que $\forall\sigma$

$$\sum_{i=1}^{\epsilon+h} C_i(\sigma) \geq \sum_{i=1}^{\epsilon+h} C_i(\sigma^a)$$

$\forall\sigma$

$$\sum_{i=\epsilon+h+1}^{n} C_i(\sigma) \geq \sum_{i=\epsilon+h+1}^{n} C_i(\sigma^a)$$

Donc $\forall\sigma$

$$\sum_{i=1}^{n} C_i(\sigma) \geq \sum_{i=1}^{n} C_i(\sigma^a)$$

Et ainsi ,la valeur du $\sum_{i=1}^{n} C_i(\sigma^a)$ est obtenue est optimale

Annexe B : Les tableaux des résultats expérimentaux

Ins	Borne	SPT	SST	SLT	LPT	LST	LLT
1	40	28,57	24,53	34,43	36,51	35,48	36,51
2	13	0,00	0,00	7,14	7,14	7,14	0,00
3	13	40,91	53,57	40,91	51,85	40,91	51,85
4	44	0,00	0,00	16,98	13,73	16,98	0,00
5	23	8,00	8,00	0,00	14,82	11,54	14,82
6	27	37,21	34,15	53,45	44,90	54,24	35,71
7	56	6,67	12,50	13,85	16,42	11,11	12,50
8	80	13,04	12,09	10,11	16,67	13,04	15,79
9	31	35,42	42,59	38,00	41,51	38,00	41,51
10	40	28,57	24,53	34,43	36,51	35,48	36,51
11	80	4,76	10,11	9,09	13,98	4,76	13,98
12	67	14,10	16,25	19,28	23,86	18,29	18,29
13	74	22,92	15,91	22,11	23,71	25,25	15,91
14	68	18,07	16,05	26,88	25,28	26,88	20,93
15	105	5,41	10,26	8,70	16,00	13,22	10,26
16	70	15,66	19,54	24,73	27,84	29,29	21,35
17	62	22,50	21,52	34,04	32,61	34,04	27,06
18	154	7,23	9,94	10,47	10,47	13,48	9,94
19	117	4,88	12,69	9,30	20,41	15,83	12,69
20	94	22,95	13,76	28,24	23,58	29,32	17,54
21	127	3,79	10,56	7,97	13,01	11,81	10,56
22	139	3,47	9,15	9,15	11,47	13,67	11,47
23	134	6,94	9,46	15,19	20,71	22,54	9,46
24	130	6,47	9,72	12,16	18,24	16,13	13,91
25	155	1,27	9,36	6,63	10,41	11,93	9,36
26	130	7,14	12,16	8,45	12,16	12,75	12,16
27	146	7,01	9,88	9,88	10,43	13,61	9,88
28	152	7,88	8,43	13,14	14,12	19,15	8,43
29	163	14,21	9,94	17,68	14,21	21,64	9,94
30	168	12,50	10,16	21,13	13,40	24,32	10,16
31	153	14,53	11,05	18,18	19,90	21,54	11,05
32	216	0,46	6,09	4,00	9,62	10,74	6,90
33	191	4,50	8,17	12,39	13,18	19,41	8,17
34	194	2,02	8,49	6,73	8,92	10,60	8,49
35	184	6,60	8,46	11,96	16,36	19,65	8,46
36	180	12,20	11,33	21,05	21,74	27,71	11,77
37	235	1,67	6,75	6,37	8,91	9,96	6,75
38	193	17,87	12,27	18,91	16,45	23,11	14,98
39	234	4,10	7,51	5,65	7,51	7,51	7,51
40	233	7,91	7,54	12,41	11,74	17,08	7,54
41	216	0,92	6,09	5,26	8,47	10,74	6,09
42	276	0,72	5,80	0,72	7,38	9,51	5,80
43	300	0,33	5,36	2,91	5,96	6,83	5,96
44	215	11,89	5,70	15,02	14,34	20,66	7,73
45	265	0,75	6,03	2,57	6,36	7,02	6,03
46	262	11,79	6,76	17,87	9,97	20,85	6,76
47	223	10,80	7,08	17,10	13,57	22,57	9,72
48	300	4,15	5,66	12,54	5,96	17,81	5,66
49	280	5,08	5,72	9,09	9,68	13,85	5,72
50	268	3,94	5,63	11,26	6,62	14,38	5,63

Ins	Borne	SPT	SST	SLT	LPT	LST	LLT
51	248	9,49	6,06	15,65	7,12	18,42	7,12
52	309	2,22	5,79	8,31	5,79	15,34	5,79
53	323	0,92	5,28	5,00	5,56	10,77	5,28
54	382	0,78	4,74	4,02	4,98	9,26	4,74
55	333	1,19	5,13	4,86	5,93	9,76	5,13
56	325	4,13	5,52	8,45	5,52	12,40	5,52
57	377	0,00	4,80	5,04	4,80	11,50	4,80
58	336	1,47	5,08	9,68	6,41	15,79	5,08
59	333	3,20	4,58	12,14	6,98	17,16	5,13
60	319	3,92	5,06	12,12	11,88	19,85	5,06
61	365	0,00	4,45	6,17	8,52	11,41	4,45
62	328	0,00	5,20	4,09	7,08	9,64	5,20
63	345	0,00	5,22	2,27	5,22	11,77	5,22
64	404	1,94	4,04	6,91	6,48	11,21	4,04
65	318	1,85	5,64	11,91	13,35	18,67	6,74
66	422	0,00	4,09	7,46	7,46	13,53	4,09
67	335	6,16	4,56	9,21	5,37	12,30	5,10
68	398	1,49	4,33	8,08	4,56	14,22	4,33
69	472	1,67	3,67	5,79	3,87	9,41	3,87
70	388	2,27	4,67	9,35	4,67	17,45	4,67
71	350	8,62	5,15	14,63	14,63	20,27	10,49
72	454	1,30	4,02	11,33	4,22	16,70	4,02
73	438	3,74	3,95	14,95	7,01	17,36	3,95
74	402	0,00	4,29	3,60	7,59	12,61	4,29
75	480	1,44	3,61	9,77	5,33	15,19	3,61
76	428	8,55	4,25	15,08	4,25	16,57	4,25
77	443	1,12	4,11	4,73	5,34	12,62	4,11
78	447	1,32	4,08	9,88	4,08	15,34	4,08
79	464	4,13	3,93	12,95	4,13	18,02	3,93
80	505	0,79	3,07	10,62	7,68	13,82	3,63
81	468	0,00	3,90	6,40	3,90	10,35	3,90
82	389	7,82	4,42	15,98	8,25	20,61	5,58
83	443	3,28	3,90	11,22	6,54	16,89	3,90
84	484	1,22	3,39	11,68	6,56	16,12	3,78
85	445	8,62	4,09	16,35	12,23	21,10	4,71
86	539	0,00	3,23	8,64	3,92	14,17	3,23
87	475	2,46	3,85	8,83	3,85	12,52	3,85
88	512	0,78	3,58	8,90	6,23	17,29	3,58
89	501	0,00	3,65	10,54	7,73	15,66	3,65
90	548	0,00	3,18	8,82	4,36	12,88	3,18
91	562	0,00	3,10	8,02	5,07	12,32	3,10
92	505	0,39	3,26	12,63	8,51	16,94	3,26
93	480	0,00	3,81	7,34	8,92	14,29	3,81
94	545	0,00	3,37	2,68	3,37	13,08	3,37
95	592	0,00	2,95	5,28	3,27	10,03	2,95
96	541	1,64	2,87	10,43	6,24	16,51	3,22
97	561	0,00	2,94	4,10	4,10	9,22	3,11
98	594	1,98	2,78	9,31	3,57	12,52	3,10
99	532	0,37	2,74	10,44	6,17	16,22	2,92
100	535	1,29	3,43	6,79	3,43	12,87	3,43

Tableau 1 : **Comparaison entre les six méthodes liste.**

Ins	Borne	HF	SR1	Ins	Borne	HF	SR1
1	40	27,27	32,20	**51**	248	5,70	5,70
2	13	0,00	7,14	**52**	309	5,79	2,52
3	13	53,57	51,85	**53**	323	5,28	2,42
4	44	0,00	2,22	**54**	382	4,74	2,80
5	23	8,00	8,00	**55**	333	5,13	3,48
6	27	30,77	44,90	**56**	325	5,52	2,40
7	56	12,50	12,50	**57**	377	4,80	3,58
8	80	12,09	12,09	**58**	336	5,08	4,82
9	31	42,59	35,42	**59**	333	4,58	1,77
10	40	27,27	32,20	**60**	319	5,06	6,73
11	80	10,11	13,98	**61**	365	4,45	1,35
12	67	16,25	16,25	**62**	328	5,48	1,50
13	74	15,91	13,95	**63**	345	5,22	0,29
14	68	16,05	18,07	**64**	404	4,04	2,65
15	105	6,25	11,77	**65**	318	5,64	4,22
16	70	19,54	16,67	**66**	422	4,09	4,09
17	62	24,39	27,91	**67**	335	4,56	3,74
18	154	9,94	8,88	**68**	398	4,33	1,24
19	117	12,69	7,14	**69**	472	3,67	3,08
20	94	13,76	14,55	**70**	388	4,67	3,24
21	127	8,63	6,62	**71**	350	5,15	10,49
22	139	9,15	9,15	**72**	454	4,02	1,30
23	134	8,84	12,99	**73**	438	3,95	3,10
24	130	9,72	5,11	**74**	402	4,29	1,71
25	155	9,36	3,13	**75**	480	3,61	1,03
26	130	12,16	3,70	**76**	428	4,25	4,25
27	146	9,88	7,01	**77**	443	4,11	3,28
28	152	8,43	5,59	**78**	447	4,08	1,54
29	163	9,94	11,89	**79**	464	3,93	2,11
30	168	10,16	13,40	**80**	505	3,07	1,17
31	153	11,05	16,39	**81**	468	3,90	1,68
32	216	6,09	4,00	**82**	389	4,42	7,16
33	191	8,17	4,02	**83**	443	3,90	2,42
34	194	8,49	4,90	**84**	484	3,39	0,62
35	184	8,46	3,66	**85**	445	4,09	4,71
36	180	10,89	15,09	**86**	539	3,41	1,64
37	235	6,75	1,26	**87**	475	3,85	1,25
38	193	12,67	13,06	**88**	512	3,58	1,35
39	234	7,51	2,09	**89**	501	3,65	0,40
40	233	7,54	7,54	**90**	548	2,66	1,08
41	216	6,09	4,85	**91**	562	3,10	0,88
42	276	5,80	3,16	**92**	505	3,26	0,98
43	300	5,36	5,36	**93**	480	3,81	2,44
44	215	5,70	13,31	**94**	545	3,37	1,45
45	265	6,03	2,93	**95**	592	2,95	2,79
46	262	6,76	9,66	**96**	541	2,87	0,73
47	223	7,08	12,21	**97**	561	2,94	2,94
48	300	5,66	5,66	**98**	594	2,78	1,49
49	280	5,72	5,41	**99**	532	2,74	0,19
50	268	5,63	5,30	**100**	535	3,43	0,56

Tableau 2 : **Comparaison entre HF et SR1**

REFERENCES

[1]. M.Barba. Produire sans homme. Revue Science et Vie, N° 882, Mars 1991, pp. 98-105.

[2]. Abdekhodaee, A.H, A, Wirth et H-S Gana . “Scheduling two parallel machines with a single server: the general case” .*Computers & Operations Research,*33 (2006) 994–1009

[3]. S,Guirchoun et P, Martineau. “Problèmes à machines parallèles avec un serveur dans les systèmes informatiques”. 4*eConférence Francophone de MOdélisation et SIMulation « Organisation et Conduite d’Activités dans l’Industrie et les Services »* MOSIM’03 – du 23 au 25 avril 2003 - Toulouse (France)

[4]. Jonson, S.W., “Optimal two and three-stage production schedules with setup times included”, *Naval Research Logistic Quarterly* 1, (1954), pp. 61-68.

[5]. M-L. Levy, P. Lopez, et B. Pradin. “Characterization of feasible schedules for the flow-shop problem: a decomposition approach. *In European Workshop on integrated Manufacturing Systems Engineering (IMSE'94)*, Grenoble (France), INRIA. pp. :307-315. décembre 1994.

[6]. GOThA. “Les problèmes d'ordonnancement” , *RAIRO-RO*, 27(1): pp. 77-1.50, 1993.

[7]. J. Blazewicz, K.Ecker, G.Schmidt, et J. Weglarz., “Scheduling in Computer and Manufacturing Systems*”. Springer- Verlag edition*, 1994.

[8]. Pinedo, M.L., “Scheduling: Theory, Algorithms and systems”. *Prentice Hall, Englewood cliffs*, New Jersey, (1995).

[9]. David Duvivier, “Etude de l’hybridation des méta-heuristiques, application a un problème d’ordonnancement de type JobShop”. *Thèse de doctorat de l’université de littoral côte d’Opale*,2000

[10]. Vincent Giard, Jeunet. “Modélisation du problème général d'ordonnancement de véhicules sur une ligne de production et d'assemblage”, *Journal Européen des Systèmes Automatisés,* 40(4-5), pp. 463-496, 2006

[11]. J. Carlier et P. Chretienne. *“Problèmes d'ordonnancement*

(modélsation/complexité /algorithme) ",.Masson édition, 1988.

[12]. MacCarthy, B.L. and Liu, J., "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling". *International Journal of Production Research,(*31) , pp. 59-79.

[13]. A. H. G. RinnooyKan., "Machine scheduling problems: classification, complexity and computations"*. Nijhoff, The Hague*. 1976.

[14]. Vignier, J.C. Billaut et C. Proust, 1999, "Les Flow shop hybrides : état de l'art", *RAIRO-RO*,vol. 2, n. 33, pp. 117-183.

[15]. E. Sanlaville. "Conception et analyse d'algorithmes de liste en ordonnancement préemtif" *PhD thesis*, Université de Paris VI, september 1992.

[16]. Sondes CHAABANE, "Gestion prédictive des Blocs Opératoires",*Thèse de doctorat ,L'institut National des Sciences Appliquées de Lyon*,2004

[17]. G. Chevalier, J. Barrier et P. Richard. "Production planning in the glass industry", Revue VERRE,(25): 27-29, 1996.

[18]. C.Proust, E.Grunenberger. "Planification de production dans un contexte de flow-shop hybride à deux étages: conception et interprogrammation d'ARIANE 2000. " *RAPA,* 8(5),pp 715-734, 1995.

[19]. R. MacNaughton. "Scheduling with deadlines and loss functions". *Management Science*,(6), pp. 1-12, 19.59.

[20]. V. Y. Shen et Y. E. Chen. "A scheduling strategy for the ftowshop problem in a system with two classes of processors". *In Conference on Information and Systems Science*, pp. 645-649, 1972.

[21]. R. E. Buten et V.Y. Shen. "A scheduling model for computer systems with two classes of processors". *In Sagomore Computer Conference on Paraud Processing*, 1973.

[22]. A. Horn., "Some simple scheduling algorithms". *Naval Research Logistics Quarterly*, (21): pp. 177-185, 1974.

[23]. R. J. Paul., “A production scheduling problem in the glass-container industry”. *Operations Research*, 21.(2):pp.290-302, 1979.

[24]. A.Langston. “Interstage transportation planning in the deterministic ftowshop environment”, *Operations Research*, 35(4):pp. 556-564, 1987.

[25]. W. Szwarc et J. N. D. Gupta, “A flow-shop problem with sequence dependant additive setup time”. *Naval Research Logistics Quarterly* (23), pp. 619-627;1987

[26]. Koulamas CP et Smith ML. “Look-ahead scheduling for minimizing machine interference”. *International Journal of Production Research*;pp.26-39,1988.

[27]. C. Sriskandarajah et S. P. Sethi. “Scheduling algorithms for flexible flow-shops: Worst and average case performance”, *European Journal of Operations Research*,(43): pp.143-160, 1989.

[28]. D. Sherali, S. C. Sarin, et M. S. Kodialam. “Models and algorithm for a two-stage production process. ” *Production Planning and Control*,(1):pp.27-39, 1990.

[29]. J. N. D. Gupta et E. A. Tunc. “Schedules for a two-stage hybrid flowshop with parallel machines at the second stage”. *International Journal of Production Research*., 29(7):pp.1489-1502, 1991.

[30]. D. L. Santos, J. L. Hunsucker, et D. E. Deal. “Global lower bounds for flow shop with multiple processors”. *European Journal of Operations Research*, (80):pp.112-120,1995.

[31]. C. Lee, T. C. E. Cheng, et B. M. T. Lin. “ Minimizing the makespan in the 3-machine assembly flowshop scheduling problem”. *Management Science*, 39(5):pp. 616-62.5:1993.

[32]. Gupta et Tunc ,“Scheduling a two-stage hybrid ftowshop with separable setup and removal times”. *European Journal of Operations Research*, (77):pp.415-428, 1994.

[33]. C. Y. Lee et G. L. Vairaktarakis. “Minimizing makespan in hybrid flowshop”. *Operations Research Letters*, 16(3): pp.149-158, 1994.

[34]. J.N.D. Gupta, A.M.A. Hariri, et C.N. Potts. “ Scheduling a two-stage hybrid flowshop With parallel machines at the first stage”. *Mathematics of Industrial Systems (Annals of operations research)*, 1995.

[35]. B. Chen.,“Analysis of classes of heuristics for scheduling a two-stage flowshop With parallel machines at one stage”. *Journal of Operational Research Society*, 46(2):pp.234-244, 1995.

[36]. S. Li. ,“ A hybrid two-stage flowshop With part family, batch production, major and minor setups”. *European Journal of Operations Research*, 1996.

[37]. A. Vignier, J-C. Billaut, et C. Proust. “Les problèmes de flow-shop hybride: état de l'art”. *Rapport Interne 155, LI/E3i/Univ. de Tours*, mai 1995. 100p.

[38]. A. Guinet, V. Botta, et M. Solomon.“ Minimisation du plus grand retard ou de la plus grande date de fin dans les problèmes d'ordonnancement de type ftowshop hybride”. Journées d’études: "Affectation et Ordonnancement", *Tours (France), CNRS .1 GdR Automatique / pôle SED / GT3*. pp. 95-111. septembre 1995.

[39]. Koulamas CP. “Scheduling two parallel semiautomatic machines to minimize machine interference”. *Computers and Operations Research 1996*;23(10):pp. 945-956.

[40]. Houari M'hallah. “Heuristic algorithm for the two-satge hybrid problem”. *Operations Research Letters,* 1997, vol 21, pp. 43-53.

[41]. F. Riane. C. Rac and A. Arliba,“Hybrid Auto-adaptable Simulated Annealing based Heuristic”,*Computers & Industrial Engineering*,(37) ,pp. 277-280. (1999)

[42]. C.A. Glass, Y.M. Shafransky et V.A. Strusevich,“Scheduling for Parallel Dedicated Machines with a Single Server”, Naval Research Logistics,vol. 47, pp. 304-328.

[43]. Nicholas G. Hall; Chris N. Pott et Chelliah Sriskandarajah.“Parallel machine scheduling with a common server”.*Discrete Applied Mathematics (*102) pp. 223-243, 2000

[44]. Brucker P, Dhaenens-Flipo C, Knust S, Kravchenko SA et F, Werner. “Complexity results for parallel machine problems with a single server”. *Journal of Scheduling* 2002*(5)*, pp.429–57,2000.

[45]. Shanling Li , *“A hybrid two-stage flowshop with part family, batch production, major and minor set-ups”*, European Journal of Operational Research, Volume 102, (1997), pp.142-156.

[46] . Abdekhodaee, A.H et A, Wirth. “Scheduling parallel machines with single

serveur : some solvable cases ". *Computer and operations research*, 2002(29), pp.295-315.

[47]. Ling-Huey Su, "A hybrid two-stage flowshop with limited waiting time constraints". *Computers & Industrial Engineering* 44 (2003) 409–424

[48]. C. Oguz, M. Fikret Ercan , T.C. Edwin Cheng , Y.F. Fung, "Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop", *European Journal of Operational Research* 149 (2003),pp 390-403

[49]. Wei Wang et John L. Hunsucker, " makespan distributions in flow shops with multiple processors", *Journal of the Chinese Institute of Industrial Engineers*, Vol. 21, No. 3, pp.242-250 (2004)

[50]. Jinxing X., Wenxun X., Zhixin L. AND Jiefang D., "Minimum Deviation Algorithm for Two-Stage No-Wait Flowshops with Parallel Machines". *Computers and Mathematics with Applications*,47 (2004) 1857-1863

[51]. Abdekhodaee AH, Wirth A., Gan HS. "Equal processing and equal setup times cases of scheduling parallel machines with a single server". *Computers and Operations Research* 2004(31),pp.1867–1889

[52]. Hamid Allaoui ,Abdelhakim Artiba, "Scheduling two-stage hybrid flow shop with availability constraints", *Computers & Operations Research* 33(2006) 1399-1419.

[53]. Jiyin Liu, "Single-job lot streaming in m - 1 two-stage hybrid flowshops",*European Journal of Operational Research*,187 (2008),pp.1171-1183

[54]. Chinyao Low, Chou-Jung Hsu, Chwen-Tzeng Su, "A two-stage hybrid flowshop scheduling problem with a function constraint and unrelated alternative machines", *Computers & Operations Research,* 35 (2008) 845-853.

[55]. Hatem Hadda, Sonia Hajri-Gabouj, Najoua Dridi, "Etude du flow shop hybride à deux étages avec machines dédiées sous contrainte d'indisponibilité", *CPI07, 5ième conférence Internationale sur la conception et la production intégrées*, Rabat, Maroc, 22, 23 & 24 Octobre 2007, pp.1-15

[56]. Laurence GELINEAU, Problème d'ordonnancement Multiprocesseur avec communication par diffusion,1996