TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

INTRODUCTION

Face à l'évolution des conditions du marché mondial et concurrentiel, l'industrie manufacturière a dû relever plusieurs défis. Outre la croissance des ventes, l'évolution rapide des techniques de production et les exigences imposées par les clients, les entreprises manufacturières sont également à la recherche de solutions pour accroître l'efficacité de leurs procédés de production. En vue d'atteindre ou de maintenir une position favorable en ce qui à trait au marché, les entreprises doivent mieux gérer leurs ressources et fournir des produits et/ou des services de la plus haute qualité possible dans un délai court et à moindre coût.

Pour répondre aux besoins du client, de nombreuses entreprises cherchent à optimiser leur production. Pour cela, il est nécessaire de disposer de la stratégie la plus performante en vue de l'organisation et de la conduite de la production de manière à répondre rapidement aux perturbations que rencontre l'entreprise. Pour être compétitif, la production cellulaire est largement utilisée dans les systèmes de fabrication. Cette dernière apparaît comme une stratégie de production apte à résoudre les problèmes de complexité et les longs délais de fabrication des systèmes de production.

Dans ce mémoire, nous nous intéressons aux problèmes d'optimisation de la sélection du meilleur routage, de la conception des cellules manufacturières ainsi que les problèmes d'ordonnancement cellulaire. Ces trois problèmes sont étroitement liés (figure 0.1). La conception des cellules manufacturières ne peut donc se faire qu'une fois que le routage de fabrication des produits a été sélectionné; de même, l'ordonnancement cellulaire ne peut se faire que si les cellules sont conçues.

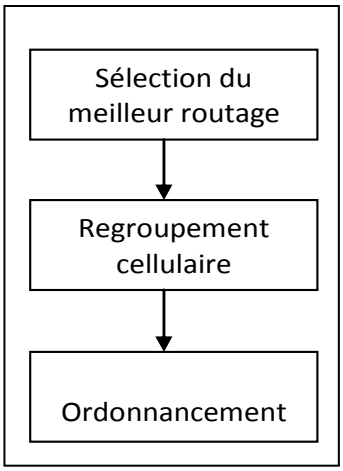


Figure 0.1 Séquence des trois problèmes.

Notre travail consiste à résoudre ces problèmes en utilisant diverses techniques d'optimisation. À cette fin, nous allons diviser notre travail en trois parties :

- la première partie consiste à sélectionner le meilleur routage pour minimiser le coût de fabrication;
- la deuxième partie a comme caractère essentiel de faire le regroupement cellulaire afin de réduire les coûts de manutention, les délais de production et les temps de transport;
- la troisième partie consiste à faire l'ordonnancement des cellules pour minimiser le makespan et *le flowtime*.

Ce mémoire est constitué de cinq chapitres. Le premier chapitre décrit l'état de l'art et fait état de la littérature des travaux qui s'intéressent à la production cellulaire. Au deuxième chapitre, nous traitons le problème de sélection du routage de fabrication optimal ainsi que le problème de regroupement cellulaire. Afin de résoudre ces problèmes, nous avons eu recours d'abord à l'application de l'algorithme de recuit simulé (RS) et l'algorithme du grand déluge étendu (GDE). Ensuite, nous comparons la performance de ces deux algorithmes avec l'algorithme génétique (AG) et l'algorithme ZODIAC en tenant compte de la qualité des regroupements obtenus. Le chapitre 3 met en évidence une étude du problème d'ordonnancement. Dans ce chapitre, nous appliquons l'algorithme GDE aux problèmes d'ordonnancement dans un environnement *flowshop* (production en série). Afin de déterminer la performance de

l'algorithme utilisé, nous comparons le makespan obtenu en utilisant l'algorithme GDE avec celui obtenu par le AG et l'algorithme du RS. Ensuite, nous appliquons cet algorithme (GDE) au problème d'ordonnancement cellulaire dont l'objectif est de minimiser le makespan et le *flowtime*. Par la suite, nous présentons une méthode de traitement des éléments exceptionnels et une heuristique d'amélioration afin de minimiser le temps mort des machines. Finalement, nous présentons les résultats obtenus et nous comparons la performance de notre approche avec la méthode « *SVS-algorithm »*.

En résumé, ce mémoire présente une nouvelle méthode pour résoudre des problèmes de formation des cellules de production et d'ordonnancement cellulaire en considérant des éléments exceptionnels tout en minimisant le temps mort des machines.

CHAPITRE 1

ÉTAT DE L'ART

1.1 Introduction

Les performances des entreprises dépendent fortement des méthodes d'organisation et d'exploitation de leurs ressources. C'est pourquoi, il est nécessaire de disposer d'un système de production qui réponde au besoin du marché. Parmi les systèmes adoptés, il y a la production cellulaire. Dans ce chapitre, nous allons examiner ce système de production ainsi que des systèmes traditionnel *jobshop* (multigammes) et *flowshop*.

1.2 Technologie de groupe

La technologie de groupe (TG) (comme son nom l'indique), consiste à regrouper des choses semblables. La TG est une méthode industrielle qui profite de la similarité des produits en les regroupant en familles. Le concept clé de la TG est de diviser totalement les produits en famille en fonction de leurs caractéristiques communes.

L'idée principale derrière la division des produits est de regrouper les pièces similaires ensemble afin d'améliorer la productivité des systèmes industriels et de réduire le coût et le temps de production.

1.3 Systèmes de production

Les systèmes de production sont classés en fonction du volume de production et du nombre des pièces produites. Une grande variété de produits traités provoque une diminution de volume de produits. Pour une grande demande, le système *flowshop* est le plus adopté, mais avec une variété réduite de produits. Pour une grande gamme, le système de production *jobshop* est plus efficace pour un volume de production réduit, entre les deux, on trouve le

système de production cellulaire qui profite des avantages des deux systèmes. Il vise une plus grande efficacité et flexibilité pour les petites et moyennes séries.

1.3.1 Le système de production *jobshop*

Pour une demande réduite, le système de production *jobshop* est utilisé à condition que s'ajoute un lot de produits très grands pour réduire le prix de production unitaire. Un environnement est dit *jobshop* si les équipements sont regroupés par type d'opération dans des ateliers différents. Les ateliers de production sont alors constitués de machines semblables. Le système *jobshop* est moins vulnérable aux perturbations causées par les bris mécaniques ou l'absence de personnel. Dans un atelier *jobshop* ou multi-gammes, la production offre une grande flexibilité lors du traitement d'une grande variété de pièces.

Dans un environnement *jobshop*, le cheminement des produits entre les ateliers se croise (figure 1.1), ce qui provoque des difficultés de manutention. En outre, le problème d'ordonnancement est complexe, car chaque atelier peut être un endroit d'entrée, de passage ou de sortie du produit.

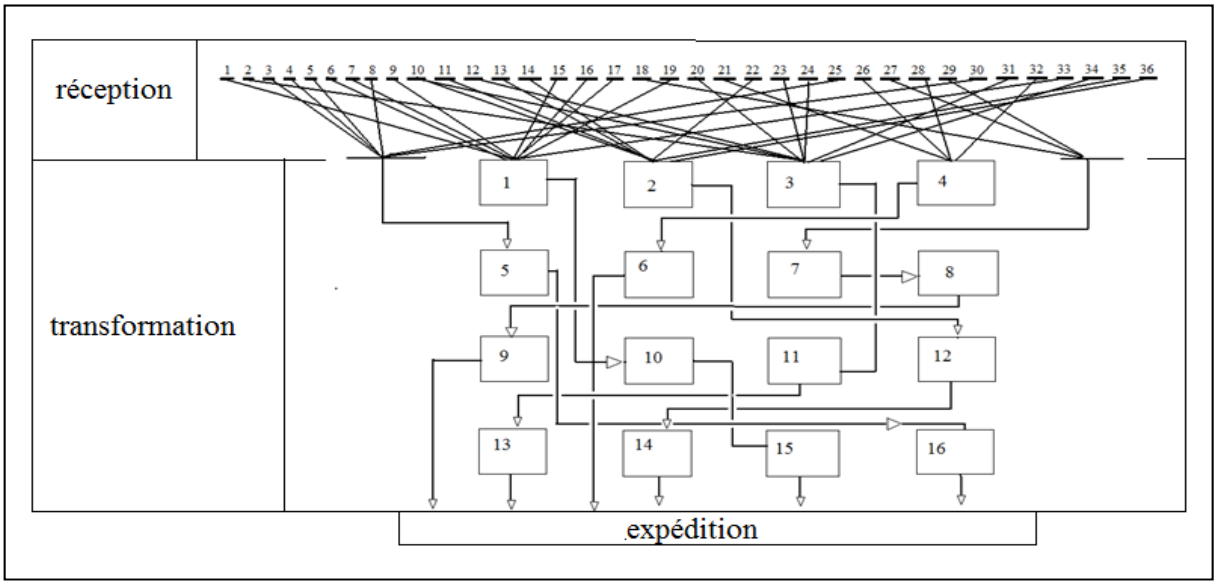


Figure 1.1 La production dans un atelier *job shop*.

1.3.2 Le système de production *flowshop*

Un environnement de production *flowshop,* appelé aussi mono-gamme, est préférable lorsque la demande est élevée avec un nombre de produits réduit, ce qui a pour effet de réduire la variation dans le processus de production pour les différents types de produits. Ce système de production offre une meilleure efficacité si les produits fabriqués ayant une variété réduite. Chaque ligne de production est sélectionnée pour fabriquer un ensemble de produits à variété réduite. Dans un environnement *flowshop,* le problème de croisement de cheminement de produits (figure 1.2) est évité, ce qui réduit les problèmes de manutention. De surcroît, le problème d'ordonnancement est moins complexe qu'un environnement *jobshop;* pour cela, il suffit de trouver la séquence de pièces optimales qui minimise le makespan. Cependant, ce système demeure moins flexible qu'un environnement *jobshop.*

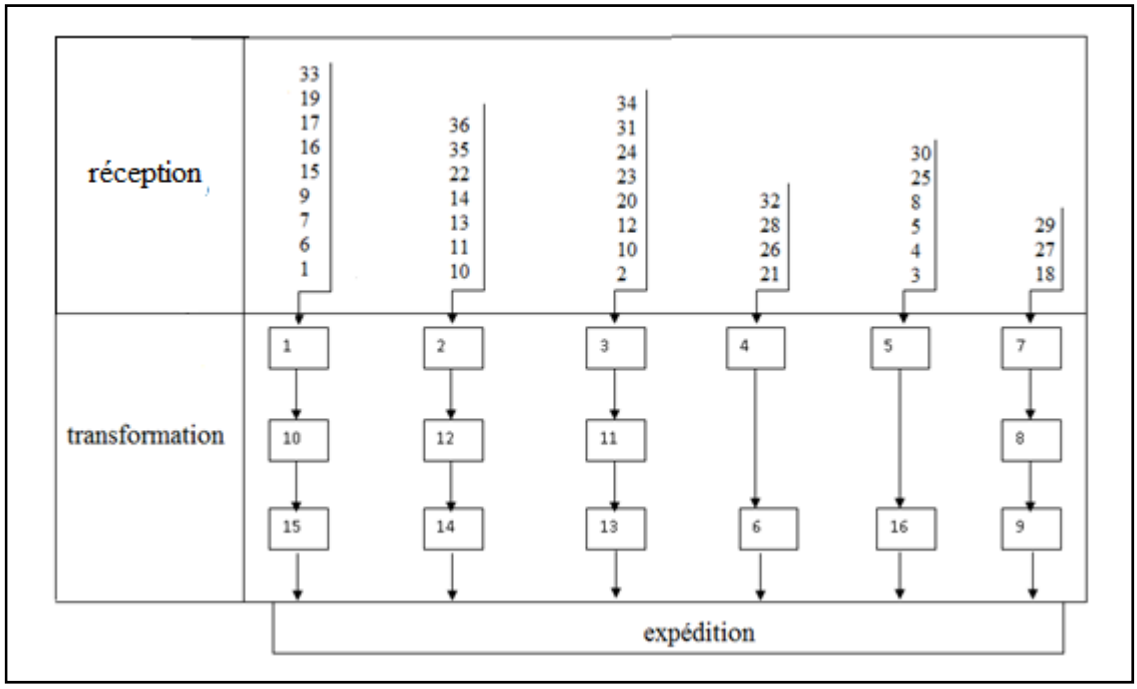


Figure 1.2 La production dans un atelier *flow shop.*

1.3.3 Le système de production cellulaire

Il est préférable d'opter pour le système de production cellulaire si le volume de production est raisonnable et s'il y a une variété de pièce moyenne. Le système manufacturier cellulaire (SMC) est un système industriel basé sur la technologie de groupe (TG). Ce concept bénéficie des similitudes des composants industriels pour améliorer la productivité.

La fabrication cellulaire implique la transformation d'une famille de pièces similaires sur un groupe de machines différentes. Cette méthode procède par la division physique des équipements de fabrication dans des cellules de production en s'inspirant du regroupement de l'organisation des concepts communs, des principes et des opérations afin d'améliorer la productivité [10]. Afin d'utiliser au mieux les avantages de la technologie de groupe, tels que la réduction des coûts de manutention et les délais de production, la transformation de toute la famille de pièces doit se faire à l'intérieur d'une seule cellule. Toutefois, il existe, en pratique, des opérations qui nécessitent un transfert dans une autre cellule : ce sont des éléments exceptionnels.

Le système manufacturier cellulaire est un compromis entre le *flow shop* et le *job shop* visant une plus grande flexibilité et efficacité. Dans un système de production cellulaire (figure 1.3), les pièces sont classées par familles selon la similarité de processus, et les machines sont regroupées en cellules suivant les exigences des opérations de ces familles. Dans la pratique, chaque cellule de machines est spécialisée pour fabriquer une ou plusieurs familles de pièces.

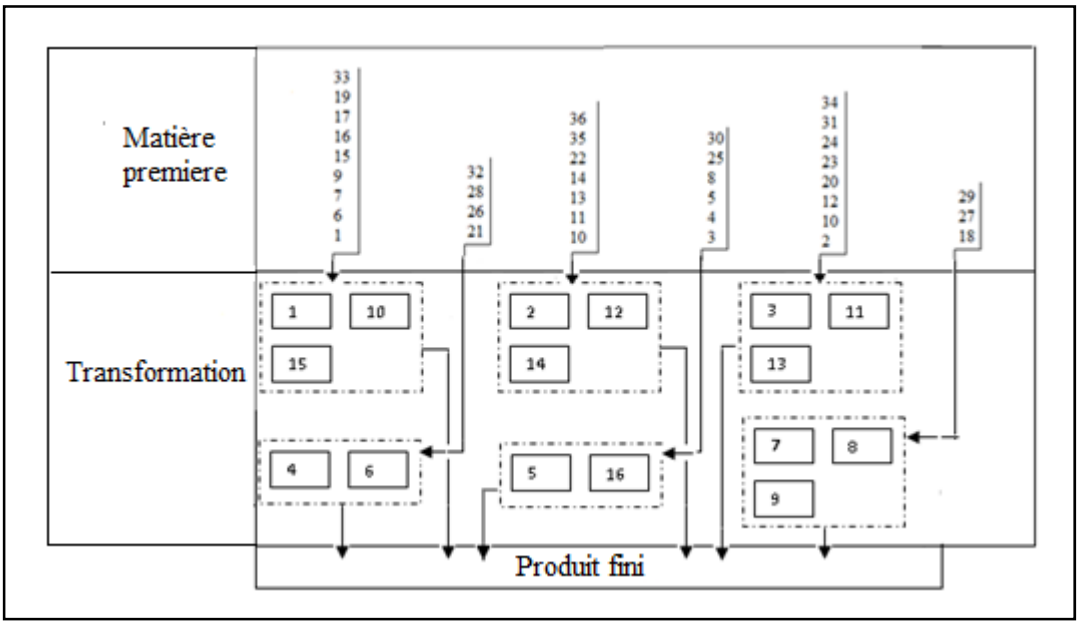


Figure 1.3 La production dans un système de production cellulaire.

1.4 Configuration d'un système de production cellulaire

Les cellules de production sont idéalement conçues de telle manière que les matières premières entrent dans une cellule et sortent en produits finis. Une cellule de production peut être complètement utilisée, indépendante ou connectée.

Cellule complètement utilisée

Chaque pièce d'une famille affectée à une cellule donnée doit être transformée par toutes les machines constituant cette cellule.

Cellules indépendantes

Chaque pièce de la famille est complètement transformée dans la cellule, indépendamment des autres cellules. Néanmoins, un tel regroupement n'est jamais atteint étant donné que les différentes pièces exigent généralement des processus différents. En d'autres termes, les

mouvements intercellulaires et les sous-utilisassions des cellules sont souvent inévitables, mais ils peuvent quand même être réduits [5].

Cellules connectées

Contrairement aux cellules indépendantes, les cellules connectées sont liées à d'autres cellules. Deux cellules dites « connectées » si la sortie de l'une est celle de l'entrée de l'autre. Un produit est complètement transformé à condition qu'il passe par les deux cellules. Dans un tel cas, la prise de décision et le chargement des cellules doivent se faire pour que les cellules puissent être connectées ensemble.

Les cellules peuvent être connectées en série ou encore en série-parallèle [64]. Les cellules connectées en série sont des cellules où le produit est transformé dans toutes les cellules de manière séquentielle (un saut de cellule peut alors être effectué). Dans les cellules connectées en série-parallèle, les familles de produits sont en mesure de suivre des chemins différents en fonction des opérations nécessaires et de se rencontrer dans la même cellule pour amorcer d'autres opérations.

1.5 Système de contrôle de la production cellulaire

Un système ne peut fonctionner correctement en l'absence d'un système de contrôle, cela est appliqué aussi pour les systèmes manufacturiers. Ainsi, toute bonne mise en œuvre des cellules de production exige un système de planification et de contrôle. Le chargement des cellules et l'ordonnancement cellulaire sont les deux principales activités qui ont un intérêt capital dans la fabrication cellulaire.

1.5.1 Chargement des cellules

Le chargement des cellules est une activité de planification de production qui se traduit par l'affectation des produits à l'intérieur des cellules. Cette activité permet aussi de déterminer le

type et les quantités de produits dans chaque cellule. Les objectifs généraux de cette étape sont la minimisation des retards, la maximisation de l'utilisation des cellules et l'équilibre de la charge entre les cellules [65].

1.5.2 Ordonnancement cellulaire

Dans un système de production cellulaire, il faut organiser et déterminer les délais de fabrication dans chaque cellule, C'est l'ordonnancement cellulaire. L'ordonnancement est un processus d'allocation des ressources au fil du temps pour réaliser un ensemble de tâches [66]. L'ordonnancement cellulaire est également un processus qui sert à déterminer le temps de début et d'achèvement sur des machines différentes une fois qu'un produit est affecté à une cellule. L'ordonnancement cellulaire peut dépendre de la configuration des cellules, des délais de traitement des produits sur des machines différentes ainsi que des délais de livraison.

Dans un environnement de production cellulaire, le problème d'ordonnancement est plus facile à résoudre en ce qui à trait aux environnements traditionnels *flowshop* et *jobshop*. Pour un problème de *m* machines et de *n* pièces, les solutions possibles des ordonnancements dans un milieu *jobshop* sont $(n!)^m$. Même pour un problème de petite taille, le nombre de solutions possibles est très important; par exemple, pour un problème de 6 pièces et 5 machines, on a $(6!)^5 = 1934917632 \times 10^5$. Ce nombre est beaucoup plus important pour des problèmes de moyenne et de grande taille. En revanche, si les pièces sont regroupées et traitées en famille, le problème d'ordonnancement devient beaucoup plus simple et réduit à $(n!)$ possibilités [32].

1.6 La conception des cellules de production

Dans cette partie, nous allons présenter les deux étapes nécessaires pour la conception des cellules de production : la sélection du routage de fabrication et le regroupement cellulaire.

1.6.1 Sélection du routage de fabrication

La sélection du routage de fabrication optimal est la première étape de la conception des cellules de production. Elle est formulée par un modèle de programmation linéaire qui tient compte des variables opérationnelles. Au début, des difficultés telles que le déséquilibre de la charge de travail et de la manutention intercellulaire sont présentes dans un système de production cellulaire conçu sans tenir compte des variables opérationnelles. Certaines procédures de conception cellulaire ont été développées tout en prenant en considération une ou plusieurs variables opérationnelles dans le but d'atteindre des objectifs tels que la maximisation de l'utilisation des ressources, la minimisation de l'investissement et de la manutention [11]. Dans la pratique, une pièce peut être exécutée selon plusieurs plans de production et chaque opération peut être effectuée sur des machines de remplacement [7]. Bien que Kusiak [12] ait montré que la prise en considération du plan de production donne une meilleure qualité de regroupement des pièces et des machines, il faut mentionner que dans le modèle qu'il propose le budget de l'entreprise et la capacité des machines n'ont pas été pris en compte.

Rajamani et al.[7], pour leur part, ont proposé trois modèles de programmation en nombres entiers pour étudier l'effet de routages alternatifs des pièces tout en tenant compte de l'utilisation des ressources dont l'objectif est de minimiser l'investissement. Liao [11], quant à lui, a proposé un modèle de programmation linéaire en nombres entiers pour déterminer le routage optimal qui minimise le coût de production en tenant compte du volume de production et de la capacité des machines. En ce qui concerne le chercheur Chang [16], il a présenté une recherche pour mesurer la flexibilité de routage; il a proposé, entre autres, des modèles mathématiques afin d'évaluer l'efficacité des routages, la polyvalence ainsi que la variété de ces routages. Pour ce qui est de Bilge et al. [15], ils ont présenté une étude qui porte sur la flexibilité de routage. Cette étude introduit l'approche de la logique floue en ce qui a trait au routage dynamique.

1.6.2 Regroupement cellulaire

La formation des cellules de production est généralement réalisée à partir d'une matrice binaire appelée *matrice incidente binaire initiale* de dimension *m*x*n* où *m* est le nombre de machines et *n* le nombre de pièces. Cette matrice composée des valeurs binaires indique la relation qui doit être établie entre les machines et les pièces. Une valeur non nulle (*i,j*) égale à 1 indique que la pièce *j* nécessite un traitement en utilisant la machine *i*. Nous présentons la méthode utilisée pour déterminer la matrice incidente binaire initiale ainsi que les approches existantes dans la littérature qui font ressortir l'utilisation de cette matrice dont l'objectif est d'obtenir la meilleure configuration de blocs regroupant les éléments non nuls.

Depuis longtemps, la formation des cellules de fabrication est reconnue comme un problème difficile à réaliser pour élaborer des concepts de la production cellulaire. Cette formation cellulaire peut être considérée comme l'un des problèmes d'optimisation difficile à résoudre, c'est-à-dire un problème d'optimisation combinatoire dit *NP-difficile*. L'augmentation de la taille du problème entraîne une augmentation exponentielle du temps de calcul pour toutes les techniques d'optimisation répandues [4]. Le nombre de solutions possibles sont déterminées à l'aide du nombre de Sterling [1], $S(M, C) = \frac{C^M}{C!}$, où M est le nombre d'éléments à regrouper en C cellules. Si le nombre de cellules est inconnu [2], ce nombre sera beaucoup plus grand, même pour des problèmes de petite taille; le nombre de cellules à obtenir est toujours compris entre 1 et le nombre d'éléments à regrouper M. Dans ce cas, le nombre de solutions faisables sera $\sum_{C=1}^{M} \frac{C^M}{C!}$. Plusieurs approches ont été utilisées pour résoudre ce problème. L'objectif de ces méthodes est de maximiser le coefficient de similarité entre les machines qui constituent une cellule ou de minimiser les mouvements intercellulaires. Ces approches sont fondées sur les hypothèses suivantes [3] :

- Le temps de fabrication des différentes pièces sur chacune des machines est identique.
- Les tailles des lots des produits à fabriquer sont égales.

Dans la pratique, ces hypothèses sont généralement invalides, car des paramètres pratiques doivent être pris en considération à condition que la demande de produits puisse être prévue avec précision. Ces paramètres sont le temps de fabrication et la capacité des machines. Pour faire face à ces limites, des solutions ont été proposées [5]:

- duplication des machines, qui constituent le goulot d'étranglement;
- sous-traitance des éléments exceptionnels;
- présence de plusieurs plans de production (routage);
- duplication des machines, qui sont liés à la duplication des lots.

Le problème de regroupement cellulaire à base binaire est largement étudié dans la littérature. Des approches ont été proposées par des chercheurs afin de résoudre ce problème. Puisque le problème de regroupement cellulaire est un problème d'optimisation de type *NP-difficile* où le temps de résolution augmente de manière exponentielle en fonction de la taille du problème, les méthodes utilisées doivent être capables de résoudre ces problèmes dans un temps de résolution acceptable, surtout pour des problèmes de grandes tailles.

Avec le grand nombre de publications sur ce sujet, seules les approches qui ont une influence sur notre recherche seront examinées.

Après avoir résolu le problème de regroupement et la formation des cellules, l'usine sera divisée en différentes cellules de production, formées par différents types de machines pour exécuter des fonctions multiples, de manière à réaliser des groupes de pièces de même type. Une fois ce problème résolu, nous nous retrouverons devant un deuxième problème qui est l'ordonnancement des pièces à l'intérieur des cellules. La disposition des machines dans une cellule pourrait être *flow shop* ou *job shop* selon le processus de fabrication. Le passage des systèmes traditionnels (*flow shop* et *job shop*) à un système de production cellulaire permet d'obtenir des avantages majeurs tels qu'une simplification de l'ordonnancement et un faible coût de manutention puisque la distance physique entre les machines est très réduite par rapport aux systèmes traditionnels, cela suppose moins de travail dans le processus de fabrication par la réduction des travaux de réglage et des délais d'installation. La garantie de ces avantages dépend de la manière dont les machines et les pièces sont groupées.

Généralement, les cellules complètement indépendantes sont les plus recherchées dans la production cellulaire : c'est le cas d'un regroupement idéal. Dans la pratique, il est rare d'obtenir un regroupement idéal, car il existe toujours des éléments exceptionnels où des transferts intercellulaires sont nécessaires. La sous-traitance ou la duplication des machines qui doivent exécuter des opérations pour plusieurs cellules seraient en mesure d'apporter des solutions afin d'éliminer les mouvements intercellulaires produis par ces éléments exceptionnels. Cependant, la duplication des machines peut être coûteuse par l'investissement de nouvelles machines, ce qui entraîne une augmentation du temps de repos, des coûts de la main-d'œuvre et de l'entretien. De même, la sous-traitance peut amener des retards de production incontrôlables ainsi que la dégradation de la qualité du produit.

1.7 Techniques de résolution

Différentes techniques et procédures de résolution ont été appliquées pour résoudre des problèmes de formation et d'ordonnancement cellulaire vu la complexité de ces deux problèmes. Parmi ces méthodes, les métaheuristiques ont été largement utilisées pour des problèmes de formation des cellules de production ainsi que pour des problèmes d'ordonnancement cellulaire. Dans le chapitre 2, nous présentons quelques techniques métaheuristiques utilisées.

CHAPITRE 2

TECHNIQUES METAHEURISTIQUES

2.1 Introduction

Résoudre un problème n'est pas toujours suffisant. La solution trouvée doit être la meilleure solution possible. En d'autres termes, il faut trouver la solution optimale du problème. Mais dans la pratique, la solution optimale n'est pas toujours faisable à cause de l'existence de certaines contraintes qui la rendent irréalisable dans le monde réel. Différentes techniques sont utilisées pour résoudre des problèmes d'optimisations complexes, parmi ces techniques on cite les approches métaheuristiques.

Les métaheuristiques sont de bonnes techniques utilisées pour résoudre des problèmes d'optimisation complexes qui sont difficiles à solutionner et qui nécessitent un temps de calcul important. Ce sont des techniques d'optimisation approchées qui produisent généralement des solutions très proches de l'optimum. Contrairement aux heuristiques spécifiques qui sont réservées à un type de problème donné, les métaheuristiques s'adaptent facilement à une large variété de problèmes combinatoires et ils convergent vers des solutions optimales qui ne sont pas connues par les méthodes traditionnelles.

Il existe aussi des heuristiques qui offrent la possibilité de trouver des optimums locaux si le problème est de trouver des optimums locaux. Dans ce cas-là, l'utilisation les métaheuristiques est inutile. Mais si le problème consiste à trouver un optimum global, les métaheuristiques ont la possibilité de ne pas tomber dans les optimums locaux. En outre, elles ont la capacité d'éviter un minimum local, mais la solution finale doit être la valeur optimale. Pour surmonter l'obstacle des minimums locaux, il s'agit d'autoriser de temps en temps, des mouvements de remonter, autrement dit d'accepter une dégradation temporaire de la situation lors du changement de la configuration courante.

Les métaheuristiques possèdent différentes caractéristiques; elles ont de nombreux avantages, mais aussi des limitations. Voici quelques points forts et limites des métaheuristiques :

- les métaheuristiques ont la possibilité de s'appliquer à une grande variété de problèmes;
- elles montrent une bonne efficacité;
- elles permettent d'obtenir une bonne performance en tenant compte de la qualité des solutions et du temps de calcul.

En revanche,

- les métaheuristiques ne garantissent pas une solution quasi optimale et il est difficile de prévoir la performance de la méthode utilisée;
- elles nécessitent une adaptation aux problèmes traités;
- elles nécessitent un bon réglage des paramètres afin d'obtenir des résultats de bonne qualité dans un temps de calcul acceptable.

Différentes métaheuristiques ont été développées dans le temps afin de résoudre les problèmes d'optimisation plus ou moins complexes. Elles sont fondées sur des procédés multiples, en particulier la biologie, le génie, la sociologie ou l'anthropologie. Dans ce chapitre, nous citons différentes algorithmes métaheuristiques qui sont utilisés dans les problèmes d'optimisation.

2.2 Le recuit simulé

L'objectif du recuit simulé est de conduire les matériaux à un état stable correspondant à un minimum absolu d'énergie. Pour cela, le matériau est porté à une température très élevée puis il est refroidi lentement. L'idée consiste à s'inspirer du mécanisme avec lequel le matériau trouve son état d'énergie minimal pour développer un algorithme d'optimisation. L'algorithme correspondant s'appelle le recuit simulé. Le recuit simule a été proposé par Kirkpatrick et al. [41] en s'inspirant de l'algorithme de Metropolis et al.[44] qui permet de décrire l'évolution d'un système thermodynamique. En utilisant, par analogie, ce processus physique, la fonction à minimiser deviendra l'énergie E du système.

2.2.1 Le mécanisme du recuit simulé

Le mécanisme de recuit simulé se présente comme suit :

- Il faut porter le matériau à une température initiale ($T=T_0$) élevée.
- Il est nécessaire de maintenir une température T=constante, le système évolue vers son état d'équilibre thermodynamique en subissant des modifications élémentaires de structure et en engendrant chaque une ΔE de l'énergie. Selon la théorie de Boltzmann et l'algorithme de métropolis, ces modifications sont telles que la probabilité que $\Delta E > 0$ soit de la forme :

$$P = e^{\left(\frac{-\Delta E}{k.T}\right)}$$, avec k : constante de Boltzmann

- Il convient de baisser légèrement la température une fois que l'équilibre thermodynamique soit atteint (à une température donnée T).
- Il faut continuer ainsi de manière à atteindre l'état final du solide cristallin.

Remarque :

L'opération de recuit simulé permet d'atteindre l'état d'énergie minimal (absolue); c'est grâce à son mécanisme que se déclenchent des modifications provisoires qui ont tendance à augmenter temporairement l'énergie.

À chaque étape, l'acceptation ou non d'un nouvel état dont l'énergie est supérieure à celle de l'état courant est déterminée de manière probabiliste, P' choisie d'une façon aléatoire compris entre 0 et 1 :

Si $P' \leq P$: le nouvel état est accepté pour remplacer l'état actuel.

Sinon : le nouvel état est rejeté et l'état actuel est maintenu.

- À haute température, $P = e^{\left(\frac{-\Delta E}{k.T}\right)} \longrightarrow 1$, la plupart des mouvements sont acceptés.

- À basse température, $P = e^{\left(\frac{-\Delta E}{k.T}\right)} \longrightarrow 0$, la plupart des mouvements qui augmentent l'énergie sont refusés, l'algorithme se ramène à une amélioration itérative classique.
- À température intermédiaire, l'algorithme autorise de temps en temps des transformations qui dégradent la fonction « objectif », il laisse ainsi au système la possibilité de s'extraire un minimum local.

2.2.2 Algorithme

La figure 2.1 montre la structure générale de l'algorithme de recuit simulé (RS). Le système initial est constitué de paramètres d'entrées pour l'algorithme, ces paramètres sont la température initiale T_0 et la vitesse de refroidissement (à chaque itération, la température est diminuée d'une valeur ΔT).

L'algorithme RS est une méthode simple et rapide à mettre en place et il est facile de l'adapter au problème traité; il est très général et facile à programmer. Également, cet algorithme a des inconvénients tels que le choix des paramètres initiaux (la température initiale T_0 et la modification élémentaire ΔT) qui sont fixés manuellement en testant diverses valeurs. L'algorithme est relativement lent, car un seul point est modifié à la fois. La vitesse de refroidissement ou de diminution de la température a une grande influence sur la performance de RS, car une fois l'algorithme piégé dans un minimum local à une basse température, la possibilité de s'en sortir tout seul est presque nulle. Il existe trois types de refroidissement ou schémas de refroidissement. Le premier est le refroidissement par paliers où la température est réduite après un certain nombre d'itérations et décroît par paliers. Le deuxième est le refroidissement continu où la température décroît à chaque itération [56]. Le troisième est la réduction non-monotone où la température est réduite à chaque itération mais avec des augmentations occasionnelles [57].

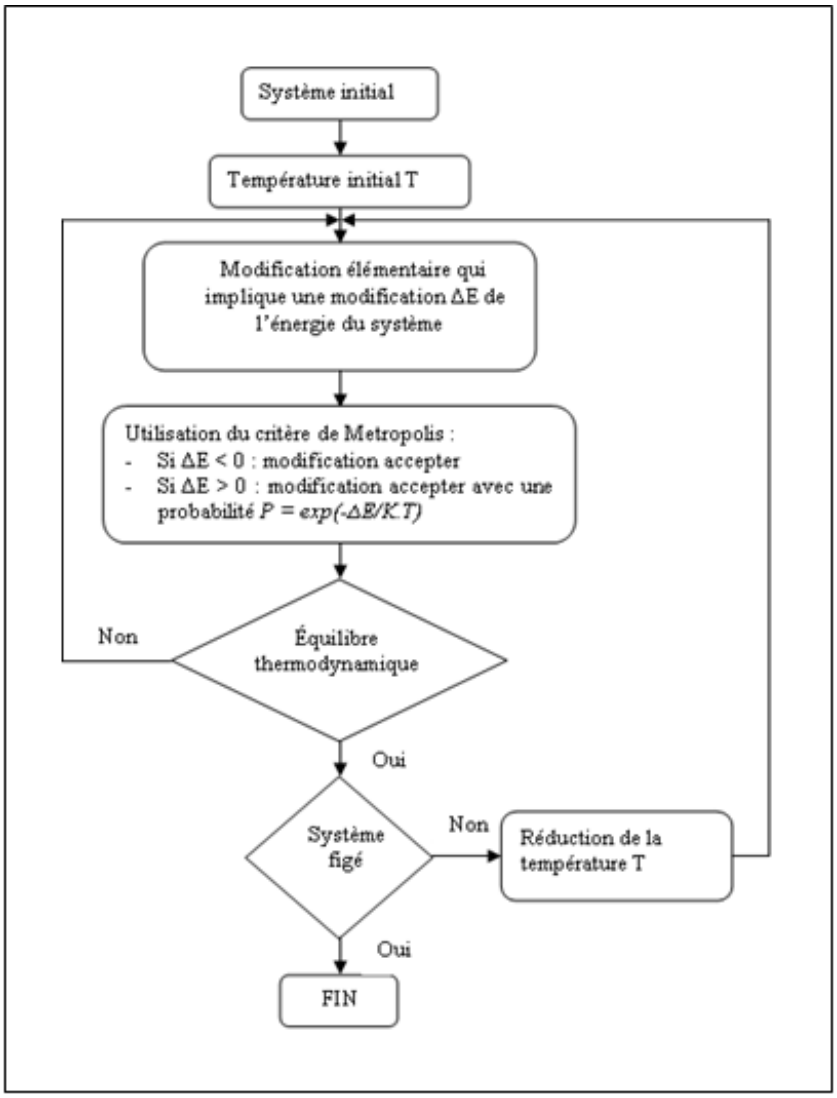


Figure 2.1 Structure générale de l'algorithme de RS.

2.2.3 Domaines d'applications

L'algorithme RS trouve son application dans de nombreux domaines dans lesquels on a à résoudre des problèmes d'optimisation difficiles. Parmi les problèmes traités, on cite les problèmes de regroupement cellulaire et les problèmes d'ordonnancement.

Vakharia et Chang [33] ont appliqué le RS dans l'ordonnancement cellulaire et ils ont comparé le RS avec un algorithme de *branch and bound* et deux autres heuristiques au niveau

de la minimisation du makespan dans une cellule *flowshop*. Les auteurs ont trouvé que pour des problèmes de petite taille, toutes les heuristiques étudiées donnent des résultats comparables à l'optimal. Par contre, le RS est supérieur aux autres heuristiques pour des problèmes de grande taille non seulement en ce qui a trait à la qualité des solutions obtenues, mais aussi en matière de temps de calcul. Une heuristique a été développée par Sridhar et Rajendran [34] [35] qui sert à trouver la bonne séquence d'opérations à l'intérieur de la cellule et qui minimise le makespan et le *flowtime*. L'heuristique proposée utilise la technique du RS et elle a été développée en deux étapes. Puisque le recuit simulé est une procédure de recherche aléatoire, les auteurs ont développé une première heuristique pour trouver une bonne solution initiale qui sera ensuite améliorée par le RS.

En raison de sa capacité de s'échapper des optimums locaux, l'algorithme du recuit simulé (RS) a été utilisé pour résoudre les problèmes de regroupement cellulaire. Cet algorithme est testé par Liu et Wu [26] pour ce type de problème et l'algorithme a montré une bonne performance en vue de trouver une solution optimale ou quasi optimale dans un temps acceptable. Une application du RS pour le problème de regroupement en introduisant les coûts de manutention et d'opérations a été développée par Boctor [27]. Il a proposé un modèle de programmation en nombres entiers pour concevoir les cellules de production; ce problème se décompose en deux : un problème d'assignation des pièces et un problème d'assignation des machines qui dépend des familles de pièces obtenues.

2.3 Algorithme génétique

Les algorithmes génétiques proviennent de la théorie darwinienne de l'évolution, ils expliquent la création des espèces en fonction de l'évolution de la vie sur terre. Darwin a introduit trois éléments fondamentaux de l'évolution : la réplication, la variation et la sélection naturelle.

La réplication est la formation d'un nouvel organisme à partir d'un précédent, au cours des processus de réplication, il se produit alors une série d'erreurs appelée variation qui

permettent un changement d'individus. En plus de la réplication et de la variation, l'évolution à besoin de la sélection naturelle, ce qui arrive lorsque les individus les plus forts réussissent à survivre en ce qui a trait à la nourriture ou lorsqu'ils vont trouver un partenaire pour se reproduire, ils vont générer un nombre plus important de progénitures. Les plus faibles vont avoir peu ou pas de descendants, ou mourir. Les gènes des individus les plus forts vont se transmettre dans plusieurs individus des générations suivantes, ce qui produit parfois des individus qui s'adaptent encore mieux à leur environnement que leurs parents.

2.3.1 Principe

Les algorithmes génétiques (AG) développés par Holland [28] s'appuient sur un codage binaire de longueur fixe et sur des opérateurs génétiques. La spécification des algorithmes génétiques est caractérisée par la façon de voir évoluer une population d'une génération à l'autre.

Un AG est constitué :

- D'individus : un individu est la représentation d'une solution du problème, il est caractérisé par une chaîne binaire appelé chromosome. Un chromosome est constitué d'une séquence de symbole dénommé gène.
- D'une population : ensemble d'individus utilisés par l'algorithme pour effectuer le processus d'évolution. La population initiale est générée au hasard. La suite est modifiée de façon itérative par des opérateurs dont le but est d'améliorer les individus.
- De l'aptitude de chacun des individus : elle montre comment un individu quelconque s'adapte à son environnement. Cette aptitude est la fonction à optimiser.
- Des opérateurs génétiques : ils sont appliqués sur les individus de la population pour les modifier afin d'obtenir une meilleure solution. Les trois opérateurs les plus utilisés dans un AG sont : la sélection, le croisement et la mutation [28].

Le croisement permet de produire deux nouveaux individus (descendants) à partir de deux autres individus (parents); il permet d'échanger deux segments ou plus des individus parents

déterminés par deux points de croisement choisis d'une façon aléatoire. La mutation est une perturbation aléatoire introduite sur la composante d'un même individu; elle consiste à changer la valeur de certains composants choisis aléatoirement dans un individu. La dernière étape, est la sélection : elle consiste à sélectionner les meilleurs individus de la population obtenue et de garder le même nombre d'individus que la population initiale [58]. La figure 2.2 présente une structure générale d'un AG.

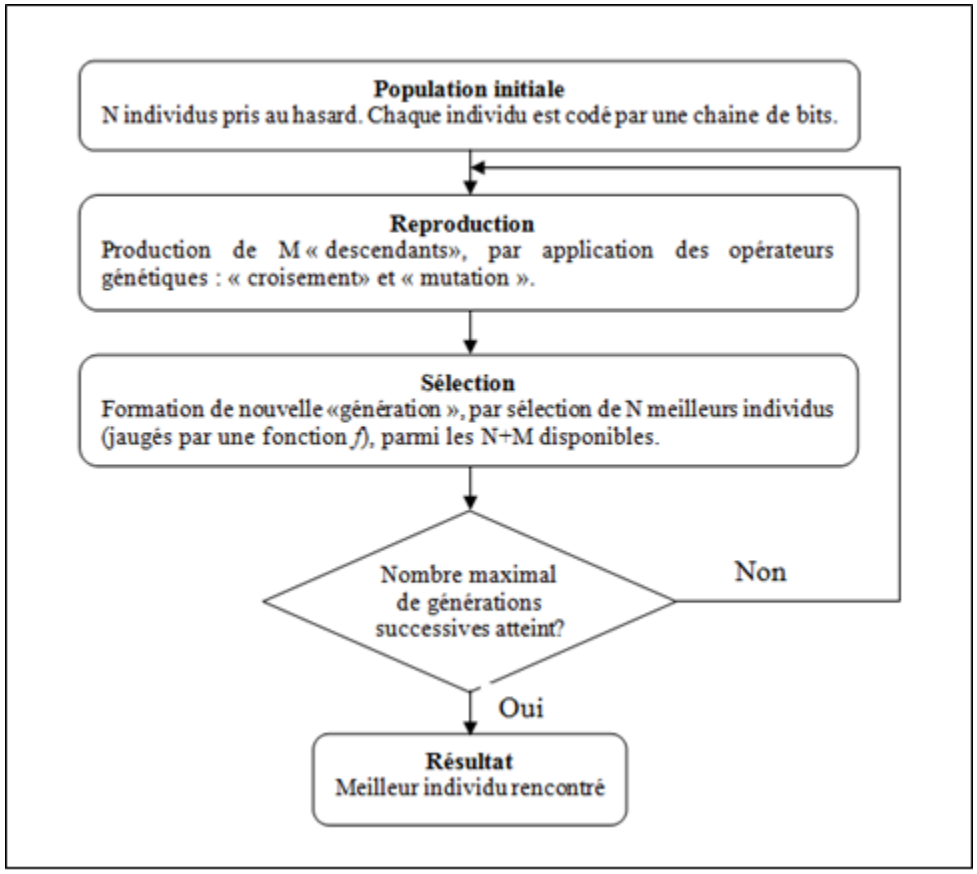


Figure 2.2 Structure générale d'un AG.

Parmi les avantages majeurs des AG, il permet d'être implanté simplement et facilement, de l'appliquer à une grande variété de problèmes incluant les problèmes combinatoires. Cependant, les AG possèdent des limites, ils sont moins efficaces qu'un algorithme spécifique dédié à un problème donné; le réglage des paramètres de contrôle est délicat tel

que la taille de la population ou le taux de mutation ou de croisement. Également, les AG ne peuvent pas garantir l'optimum.

2.3.2 Domaines d'applications

Les AG ont été utilisés pour résoudre des problèmes industriels. Ils ont été appliqués pour tester la performance des logiciels après chaque modification. Ils ont été utilisés aussi pour l'optimisation de la segmentation des traces des logiciels [59] ainsi que dans des problèmes de regroupement des cellules de production et des problèmes d'ordonnancement. Aussi, il y a eu des applications des AG à la reconnaissance et la classification des formes et d'apprentissage [29].

Venugopal et Narendan [30] ont été les premiers à appliquer les AG pour résoudre des problèmes de formation cellulaire. Ils ont développé une approche qui minimise le nombre de transferts intercellulaires ainsi que la variation de la charge des cellules. Aussi, Zolfaghari et Liang [31] ont utilisé les AG pour le même problème et ont proposé un modèle pour maximiser une fonction qui mesure le regroupement.

Les AG sont parmi les approches utilisées pour résoudre des problèmes d'ordonnancement dans un environnement de type *flowshop* avec l'objectif de minimiser le makespan. Rajkumar et shahabudeen [60] ont proposé un algorithme génétique amélioré (*Improved Genetic Algorithm « IGA »*) pour la résolution des problèmes d'ordonnacement *flowshop*. Les AG ont, aussi, été appliqués par Sridhar et Rajendran [35] pour résoudre un problème multi-objectif d'ordonnancement cellulaire dans un environnement *flowshop* dont l'objectif est de minimiser le makespan, le *flowtime* et le temps mort. L'algorithme proposé a été évalué et comparé avec des heuristiques d'ordonnancement cellulaire *flowshop* multicritères existants. D'après les auteurs, l'algorithme génétique donne de meilleurs résultats que ceux donnés par les heuristiques existantes.

2.4 La recherche tabou

La recherche Tabou (RT) est une technique d'optimisation proposée par Glover [62] en 1986. C'est une méthode de recherche locale utilisée pour résoudre des problèmes complexes.

2.4.1 Principe

Le processus de recherche se fait de façon itérative. Il débute par une solution initiale faisable qui sera améliorée progressivement par un mouvement dans son voisinage. Le principe de base de la RT est de guider le processus de recherche chaque fois qu'il rencontre un minimum local. Il lui permet de faire des mouvements qui dégradent la solution actuelle, c'est le cas lorsque toutes les solutions du voisinage sont pires que celles de la solution actuelle. Avec ce processus de recherche, il peut s'échapper d'un minimum local. Le risque est qu'à l'itération suivante, le processus de recherche retombe, dans le minimum local, auquel il vient de s'échapper. Pour cela, la RT l'empêche de retourner à des solutions visitées précédemment par l'utilisation d'une mémoire à court-terme, appelée liste tabou d'une taille donnée, qui enregistre l'historique récent de la recherche. La RT de base peut être considérée comme la combinaison d'un processus de recherche locale avec une mémoire à court terme.

La taille de la liste taboue est un paramètre très important et difficile à déterminer. Si sa taille est très grande, des solutions voisines seront inaccessibles, ce qui réduit la capacité de la méthode à visiter des solutions voisines. Si sa taille est trop faible, il y a risque que la méthode soit bloquée dans un minimum local.

L'utilisation du tabou n'améliore pas toujours la solution. Pour cela, il est nécessaire d'utiliser des dispositifs algorithmiques qui permettent de révoquer les tabous et d'accepter une solution, même si elle est taboue, à condition qu'elle aboutisse à une meilleure solution qui n'a jamais été rencontrée. C'est le principe d'aspiration.

La recherche avec les tabous est une méthode de recherche locale facile à implanter, elle s'adapte aux problèmes présentant des contraintes, elle permet d'avoir un paramétrage

simplifié par le choix d'une stratégie de mémorisation (la longueur de la liste tabou). De plus, le temps de calcul est relativement moindre par rapport aux autres méthodes.

En revanche, un réglage délicat de la liste tabou à une très grande influence sur la qualité des résultats obtenus, aussi la RT ne garantit pas une convergence vers l'optimum, pour cela il y a différentes manières de fixer la longueur de la liste tabou qui peut être aléatoire et bornée, statique, dynamique etc en fonction du problème traité. Pourtant, la recherche avec les tabous a montré une grande performance pour résoudre des problèmes complexes.

2.4.2 Applications

L'efficacité de la RT lui permet d'être largement utilisée dans des problèmes d'optimisation combinatoire. Parmi les domaines d'application, la RT est appliquée dans des problèmes de formation cellulaire, Logendran et Ramakrishna [63] ont étudié ce problème et ils ont proposé un modèle mathématique dont l'objectif est de minimiser la somme pondérée des mouvements intercellulaire et intracellulaire, ensuite ils ont utilisé la RT pour se rapprocher de la solution optimale.

La RT est appliquée aussi pour résoudre des problèmes d'ordonnancement. Une application de la recherche avec les tabous a été proposée par Skorin-Kapov et Vakharia [36] qui ont étudié la performance de la recherche avec les tabous dans les problèmes d'ordonnancement cellulaire *flowshop* pour minimiser le makespan et qui ont comparé leur méthode avec le RS proposé par Vakharia et Chang [33]. Ils ont conclu que pour des petits problèmes, les deux méthodes donnent presque les mêmes résultats, mais il est préférable d'utiliser la recherche avec les tabous pour problèmes plus compliqués.

Schaller [37] a proposé de nouvelles procédures pour l'ordonnancement d'une cellule de production *flowshop*. Il a comparé les résultats obtenus avec ces méthodes aux résultats obtenus en utilisant d'autres méthodes. Ces procédures ont été évaluées dans le contexte de minimisation du makespan. Schaller [37] a conclu que ces procédures peuvent donner de bons

résultats et que les résultats obtenus en utilisant la méthode de recherche avec les tabous développée par Skorin-Kapov et Vakharia [36] peuvent être améliorées si la solution initiale (la séquence de pièces au départ) est générée par l'une de ces procédures.

2.5 Les réseaux de neurones

Une autre approche qui a été largement appliquée au problème de regroupement cellulaire est les réseaux de neurones. Kao et Moon [13] ont proposé une approche de retro-propagation de réseaux des neurones pour le problème de formation des familles de pièces. Le processus d'apprentissage est à la base du succès de cette approche; il s'appuie sur la disponibilité des données d'entraînement. L'application de cette approche peut être limitée, car la formation des cellules de production et des familles de pièces est réalisée durant la phase de conception où généralement les données d'entraînement ne sont pas disponibles à l'avance [5].

Une des théories des réseaux de neurones nommée « *Adaptative Resonance Theory* » (ART1) est appliquée pour résoudre des problèmes de formation cellulaire dont l'avantage est de traiter des problèmes de grandes tailles. Les réseaux ART1 ont été appliqués dans ce type de problème par plusieurs chercheurs dont Kusiak et Chung [17], Kaparthi et Suresh [14], Liao et Chen [18], Venugopal and Narendren [19], Dagli et Huggahalli [20][21]. Une des approches proposées est celle développée par Suresh et Kaparthi [22] qui est une combinaison entre un réseau ART1 et une approche par logique floue. Elle est basée sur le choix des paramètres et le taux d'apprentissage. Une méthode similaire appelée « FACT » capable de résoudre des problèmes plus complexes est proposée par Kamal et Burke [23].

Également, des méthodes basées sur le réseau de Hopfield ont été développées pour résoudre le problème de regroupement cellulaire, Arizono et al. [24] ont développé un modèle stochastique basé sur le réseau de Hopfield. Zolfaghari [5] a conçu une structure basée sur le recuit simulé et les réseaux de neurones dénommée « Ortho-Synapse Hopfield Network » (OSHN) pour résoudre le problème de regroupement à base binaire; il a aussi appliqué les réseaux de neurones pour résoudre les problèmes d'ordonnancement. Une approche qui a été

développée pour guider le processus de recherche OSHN a réglé les paramètres du réseau et a permis de s'échapper des optimums locaux. De même, Ateme-Nguema et Dao [25][42] ont développé une heuristique de résolution hybride composée d'un réseau de Hopfield quantifié et fluctuant accompagnée d'une méthode de recherche avec les tabous pour résoudre des problèmes de formation cellulaire. Les auteurs ont utilisé un réseau de Hopfield quantifié et fluctuant afin de bénéficier des avantages de la quantification et de la fluctuation de neurones. La quantification permet de réduire la taille de réseau en gardant sa capacité de mémorisation alors que la fluctuation permet de s'échapper des minimums locaux [25][42].

2.6 L'algorithme de grand déluge étendu

Une procédure de recherche locale appelée « grand déluge étendu » (GDE) a été introduite par Dueck [61] en 1993. Il est considéré comme un algorithme de recherche locale où certaines mauvaises solutions dont la valeur ne dépasse pas une certaine limite B sont acceptées. Cette limite diminue (dans le cas de problèmes de minimisation) d'une façon monotone durant la recherche. La valeur initiale de B est égale à la fonction « objectif » initiale et à chaque itération sa valeur est diminuée d'un pas ΔB fixe pour des problèmes de minimisation; cette valeur augmente de la même valeur pour des problèmes de maximisation. Le pas ΔB, aussi appelé taux de décroissance/croissance, représente un paramètre d'entrée pour cette approche. Durant la recherche, la limite B représente une limite entre un espace de recherche réalisable et non réalisable, il sert à pousser la solution vers l'espace réalisable. En d'autres termes, le voisinage de la solution est coupé par la limite B et la recherche se fait d'un seul coté, au dessous ou au dessus de la limite B selon l'objectif. Le processus de décroissance/croissance de B peut être considéré comme un processus de contrôle qui conduit la recherche vers une solution souhaitable. Au début de la recherche, la solution actuelle a la possibilité de faire des mouvements dans les deux sens à l'intérieur de la partie faisable limitée par B. Autrement, il y a une grande possibilité d'accepter de mauvaises solutions puisque la limite B est située à une longue distance de la solution choisie et une petite partie du voisinage est coupée. Au cours de la recherche, la limite B rapproche de plus en plus à la valeur de la solution actuelle, l'espace

de recherche devient plus réduit et la possibilité d'améliorer la solution devient plus faible, ce qui conduit à la fin du processus de recherche [40].

La seule application réalisée avec cette approche est celle du problème d'optimisation de l'horaire des examens (*Exam Timetabling Problem*) traitée par Burke et al.[40]. Les résultats ont prouvé l'efficacité de l'algorithme [40]. Plusieurs résultats existants ont été améliorés où des approches comme les recherches avec les tabous avaient été utilisées. La figure 2.3 présente une description générale de l'algorithme tel que présentée par Burke et al.[40].

Définition d'une solution initiale $\boldsymbol{S}$
Calcul de coût initial $\boldsymbol{f(s)}$
Limite supérieure initiale $\boldsymbol{B=f(s)}$
Spécification de paramètre d'entrée $\Delta\boldsymbol{B=?}$
Tant que la condition d'arrêt n'est pas satisfaite, faire
    Définition de voisinage $\boldsymbol{N(s)}$
    Sélection au hasard d'une solution $\boldsymbol{S^*} \in \boldsymbol{N(s)}$
    Si $\boldsymbol{(f(s^*) \leq f(s))}$ ou $\boldsymbol{(f(s^*) \leq B)}$
        Accepter S^*
        Diminution de la limite supérieure $\boldsymbol{B = B - \Delta B}$
    Fin si.
    Fin tant que.

Figure 2.3 Le grand déluge étendu.

L'avantage de cet algorithme est de régler un seul paramètre, c'est le taux de croissance ou décroissance de B qui est noté ΔB. Les tests exécutés par Burke et al. [40] ont montré que le temps de convergence de l'algorithme dépend de la valeur du pas ΔB. En effet, en augmentant sa valeur, le temps de convergence diminue, mais la qualité des solutions peut se dégrader aussi, ce qui explique la nécessité de bien choisir ce paramètre afin d'avoir un bon compromis entre la qualité des résultats et le temps de calcul. L'algorithme du GDE est comme les autres métaheuristiques, il ne garantit pas l'optimum, car le processus de recherche est en fonction de la solution initiale sélectionnée aléatoirement qui représente aussi la limite initiale B et la valeur de ΔB choisie.

2.7 Conclusion

Après avoir étudié ces métaheuristiques, nous avons choisi l'algorithme de GDE pour notre recherche. Cet algorithme a montré une grande efficacité dans le problème d'optimisation de l'horaire des examens ainsi qu'une simplicité de paramétrage et de programmation. Toutes les autres techniques citées dans ce chapitre ont été utilisées pour des problèmes de regroupement et d'ordonnancement; pour cette raison, nous découvrirons la performance de cet algorithme en ce qui a trait aux deux problèmes soulevés ci-dessus.

CHAPITRE 3

LA CONCEPTION DES CELLULES DE PRODUCTION

3.1 Introduction

Le problème à résoudre consiste à former les cellules de production. Ce problème se divise en deux parties. La première partie est de sélectionner le cheminement optimal de la fabrication des produits. La deuxième est de regrouper les machines en cellules et les pièces en familles. Nous présentons dans ce chapitre les méthodes adoptées pour résoudre ces deux problèmes.

3.2 Sélection du routage de fabrication optimal

Le problème de sélection du routage de fabrication optimal est formulé comme un modèle de programmation linéaire. Il consiste à déterminer le meilleur routage tout en minimisant le coût de fabrication et en tenant compte du volume de production, les alternatives associées au processus de mise en production, ainsi que la disponibilité et la capacité des machines.

3.2.1 Modèle mathématique

Liao et al.[39] proposent un modèle de programmation linéaire qui permet de déterminer le meilleur routage, qui minimise le coût de fabrication du produit en respectant les différentes contraintes.

Comme il est expliqué par Liao et al.[39], le calcul du coût opérationnel pour une pièce *k* sur une machine *m* relative à une opération *o* en utilisant le processus *p* est calculé comme suit :

$$C_{mo}=c_o(T_{su}+Q(T_m+T_{th}))/(Q) \tag{3.1}$$

Avec :

C_o : les frais d'exploitation horaires.

T_{su} : le temps de préparation pour le traitement de la pièce k par la machine m.

Q : la taille du lot.

T_m : les temps d'usinage par pièce.

T_{th} : le temps moyen de fabrication par pièce.

Le modèle mathématique de programmation linéaire qui permet la sélection de meilleur routage de production en minimisant les coûts opérationnels est le suivant [39]:

Indices :

- k=1,2,…,K désigne les produits à fabriquer;
- m=1,2,…,M identifie les machines disponibles;
- p=1,2,…,P_k représente le cheminement de chaque produit k; chaque produit pourrait avoir un ou plusieurs cheminements possibles;
- o=1,2,…,$O(k,p)$ désigne une opération d'une pièce k selon le cheminement p; il est possible d'avoir une ou plusieurs opérations associées à chaque routage.

***Variables de décision* :**

- Y_{kp}= 1 si le produit k est usiné selon le routage p,
  0 sinon.
- $X_{mo}(k,p)$ représente la portion du lot de production pour que la machine m utilisée pour réaliser l'opération o relative à la pièce k selon le routage p.

Les coefficients :

- $C_{mo}(k,p)$ le coût opérationnel de la machine m pour réaliser l'opération o associée à la combinaison (k,p);
- $t_{mo}(k,p)$ le temps opérationnel de l'opération o associée à la combinaison (k,p) sur la machine m;
- $a_o(k,p)$ = 1 si l'opération o doit être faite pour la combinaison (k,p),
  0 sinon;
- α_{mo} = 1 si la machine m est capable de réaliser l'opération o,
  0 sinon;

- t_m = le temps opérationnel disponible sur la machine m;
- d_k la demande pour le produit k.

Fonction objectif

$$Minimiser \quad f = \sum_{k,p,m,o} d_k c_{mo}(k,p) X_{mo}(k,p) \tag{3.2}$$

Sujet à :

$$\sum_{p} Y_{kp} = 1 \; \forall \; k \tag{3.3}$$

$$\sum_{m} \alpha_{mo} X_{mo}(k,p) = a_o(k,p) Y_{kp} \qquad \forall o,k,p \tag{3.4}$$

$$\sum_{k,p,o} d_k X_{mo}(k,p) t_{mo}(k,p) \le t_m \qquad \forall \; m \tag{3.5}$$

$$Y_{kp} = (0,1) \qquad \forall \; k,p \tag{3.6}$$

$$X_{mo}(k,p) = (0,1) \qquad \forall \; m,o,k,p \tag{3.7}$$

- l'équation (3.2) représente la fonction « objectif » qui sert à minimiser les coûts de production afin d'identifier le meilleur routage selon les paramètres et les contraintes opérationnels;
- la contrainte (3.3) garantit qu'un et un seul routage de production sera choisi pour chacun des produits traités;
- la contrainte (3.4) assure que toutes les opérations du routage sélectionné sont réalisées sur les machines disponibles;
- la contrainte (3.5) est utilisée pour ne pas dépasser la capacité de chaque machine;
- les deux équations (3.6) et (3.7) représentent les variables décisionnelles Y_{kp} et X_{mo} qui sont des paramètres binaires.

3.2.2 Exemple

Dans cette partie, l'exemple de référence a été traité par Liao et al. [39] afin de tester les approches qu'ils proposent pour le regroupement cellulaire. Dans cet exemple, il y a neuf machines pour la fabrication de neuf pièces qui sont réalisées par trois opérations. Chacune de ces pièces possède un ou plusieurs cheminements de production. Ce problème est considéré comme un exemple de petite taille.

Dans cet exemple, les machines réalisent au plus trois opérations pour chaque produit. Les pièces 3, 6 et 9 peuvent être réalisées par deux cheminements possibles alors que les autres sont traitées selon un seul cheminement. Toutes les pièces sont fabriquées par deux ou trois opérations. Ainsi, la pièce 1 est fabriquée selon un seul cheminement par les deux opérations 1 et 3, la pièce 2 est fabriquée aussi selon un seul cheminement par les opérations 1 et 2 mais, la pièce 3 est fabriquée selon deux cheminements possibles : elle est fabriquée nécessairement par les opérations 1 et 3 comme premier cheminement et par les opérations 1 et 2 comme deuxième cheminement. Le reste des pièces à fabriquer suivent soit un ou deux cheminements. Le tableau 3.1 regroupe les opérations nécessaires pour la fabrication des différentes pièces en fonction des cheminements possibles ainsi que la demande pour chaque produit.

Tableau 3.1 Demande et opérations en fonction des cheminements possibles

pièces	1	2	3		4	5	6		7	8	9	
cheminement	1	1	1	2	1	1	1	2	1	1	1	2
opération 1	1	1	1	1	1		1	1	1	1	1	1
opération 2		1		1	1	1	1			1	1	1
opération 3	1		1		1	1	1	1	1	1	1	
demande	5	10	10		5	5	5		10	10	10	

Dans cet exemple, les machines ne sont pas toutes capables de réaliser les trois opérations. Les machines A, B et C sont capables de réaliser l'opération 1, les machines D, E et F sont capables de réaliser l'opération 2 et les machines G, H et I sont capables de réaliser

l'opération 3. Chaque machine possède une capacité maximale. Le tableau 3.2 regroupe ces données.

Tableau 3.2 Lien opérations/machines et capacité

machine	A	B	C	D	E	F	G	H	I
opération 1	1	1	1						
opération 2				1	1	1			
opération 3							1	1	1
capacité	80	80	80	50	50	50	50	50	50

Finalement, le tableau 3.3 présente les temps et les coûts opérationnels associés à chaque pièce sur les différentes machines selon les différents cheminements possibles.

Tableau 3.3 Temps et coûts opérationnels

	pièces		1	2	3		4	5	6		7	8	9	
	cheminement		1	1	1	2	1	1	1	2	1	1	1	2
Opération1	**Machines**	A	4, 2	M, M	4, 5	4, 4	2, 2		5, 5	6, 5	7, 6	M, M	10, 8	M, M
		B	6, 4	3, 2	5, 3	3, 3	M, M		6, 3	6, 5	M, M	2, 1	9, 8	6, 6
		C	M, M	3, 4	3, 2	4, 5	M, M		4, 3	5, 4	9, 8	3, 2	8, 7	6, 5
Opération2		D		5, 4		M, M	2, 3	4, 5	2, 2			2, 2	4, 5	5, 7
		E		2, 3		4, 4	M, M	4, 4	2, 2			2, 1	4, 5	4, 6
		F		4, 4		4, 3	M, M	5, 5	2, 1			2, 2	3, 4	3, 5
Opération3		G	3, 4		4, 4		2, 1	M, M	4, 4	4, 4	2, 1	3, 3	3, 3	
		H	4, 5		M, M		M, M	2, 1	3, 3	4, 3	3, 2	3, 2	4, 3	
		I	M, M		3, 2		M, M	3, 3	3, 2	3, 2	2, 2	4, 4	3, 3	

Dans le tableau 3.3, le premier chiffre représente le temps nécessaire pour réaliser une opération sur une machine donnée et le deuxième chiffre représente le coût de fabrication.
Par exemple, si la machine A est utilisée pour réaliser l'opération 1 de la pièce 1, elle prend 4 unités de temps pour un coût de 2 unité monétaire par heure (UM/h). Le temps et le coût opérationnel peuvent changer selon le cheminement choisi.

Notons que certaines valeurs associées aux temps et aux coûts de fabrication sont relativement importantes et sont désignées par *M* qui indique une valeur très grande. Elle indique que l'opération associée à la pièce correspondante soit techniquement ou économiquement non faisable. Dans notre cas, nous avons fixé M=1000.

En utilisant les données présentées dans les tableaux 3.1, 3.2 et 3.3, un modèle de programmation linéaire a été formulé et résolu pour sélectionner le routage de fabrication optimal pour chaque pièce. Le modèle de programmation linéaire a été résolu à l'aide du logiciel LINGO, mais aussi nous pouvons utiliser d'autres logiciels de programmation linéaire tels que LINDO, GAMS ou encore Matlab. Les résultats trouvés indiquent que toutes les pièces sont fabriquées selon le cheminement 1, sauf la pièce 9 qui est fabriquée selon le cheminement 2; ces résultats permettent d'avoir un coût de production minimal égal à 440. Le tableau 3.4 est la matrice incidente obtenue, elle représente les données initiales pour la formation des cellules de fabrication ou des cellules de machines qui correspondent à des ateliers de fabrication.

Tableau 3.4 La matrice incidente obtenue

	Pièces		1	2	3	4	5	6	7	8	9
	Cheminement		1	1	1	1	1	1	1	1	2
Opération1	**Machines**	A	0	0	0	1	0	0	1	0	0
		B	1	1	1	0	0	0	0	1	0
		C	0	0	0	0	0	1	0	0	1
Opération2		D	0	0	0	1	1	0	0	0	0
		E	0	1	0	0	0	0	0	1	0
		F	0	0	0	0	0	1	0	0	1
Opération3		G	1	0	0	1	0	0	1	0	0
		H	0	0	0	0	1	0	0	1	0
		I	0	0	1	0	0	1	0	0	0

3.3 Formation des cellules de fabrication

Dans cette partie, deux algorithmes sont proposés pour résoudre le problème de regroupement cellulaire basé sur une matrice binaire incidente. La première approche est basée sur l'algorithme de GDE; la deuxième est basée sur l'algorithme du RS. Pour l'évaluation de la qualité des solutions obtenues par les deux approches, une mesure de regroupement est nécessaire; c'est le calcul de l'efficacité du regroupement obtenu noté *e* qui sera expliquée en détails par la suite.

3.3.1 Solution proposée

La formation des cellules de fabrication est faite à partir d'une matrice incidente $A=[a_{mn}]$ de M machines et N pièces déjà obtenue dans la sélection du routage optimal. Avec a_{mn}, la charge du traitement de la pièce n sur la machine m. Le problème est équivalent à une décomposition en un certain nombre de blocs, où chaque bloc représente une cellule de fabrication. Tous les éléments non nuls à l'intérieur de ces blocs représentent le mouvement intracellulaire, tandis que ceux qui sont en dehors des blocs sont les éléments exceptionnels qui représentent les flux de matières intercellulaires. Pour former les cellules des machines et les familles de pièces, toutes les entrées non nulles de la matrice incidente A devraient être proches les unes aux autres. Pour cela, plusieurs techniques sont proposées dans la littérature pour mesurer le degré de rapprochement des entrées non nulles de la matrice A. Dans notre recherche, nous avons utilisé une technique proposée par Miltenburg et Zhang [45] caractérisée par une énergie de liaison des éléments les uns aux autres, notée α, calculée comme suit :

$$\alpha = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N-1} a_{mn}a_{m,n+1} + \sum_{m=1}^{M-1}\sum_{n=1}^{N} a_{mn}a_{m+1,n}}{\sum_{m=1}^{M}\sum_{n=1}^{N} a_{mn}} \tag{3.8}$$

Plus la valeur de α augmente, plus les éléments non nuls de la matrice incidente se rapprochent. Par conséquent, nous cherchons à maximiser la valeur de α pour faciliter la formation des cellules. Afin de mesurer l'efficacité du regroupement des cellules, une technique très utilisée dans la littérature a été proposée dans [46]; où e_1 mesure la densité cellulaire en fonction de n_1 qui est le nombre de valeurs non nulles à l'intérieur des cellules. Par conséquent, une valeur élevée de e_1 signifie que les machines et les pièces dans chaque cellule de fabrication sont très semblables les unes aux autres. Le paramètre K représente le nombre de cellules de fabrication formées; M_k et N_k représentent le nombre de machines et de pièces formant la cellule k, avec k=1, 2,…, K.

$$e_1 = \frac{n_1}{\sum_{k=1}^{k} M_k N_k} \tag{3.9}$$

Aussi, n_2 est le nombre d'éléments exceptionnels dans la matrice incidence machines/pièces. Le paramètre e_2 permet de mesurer le flux intercellulaire. Une faible valeur de e_2 (proche ou égale à 0) indique qu'il y a moins d'éléments exceptionnels qui se trouvent dans la matrice incidente.

$$e_2 = 1 - \frac{n_1}{n_1 + n_2} \tag{3.10}$$

L'efficacité du regroupement *e* est caractérisée par des valeurs qui varient entre -1 et 1. Plus les machines et les pièces sont bien regroupées, plus l'efficacité du regroupement augmente. L'efficacité *e* est donc donnée par la relation suivante:

$$e = e_1 - e_2 \tag{3.11}$$

Le tableau 3.5 est un exemple de regroupement : le tableau 3.5. a est la matrice initiale incidente et le tableau 3.5. b est le résultat de regroupement associé.

Tableau 3.5 Problème de regroupement idéal

		Pièces						
		1	2	3	4	5	6	7
Machines	1	1	1		1		1	
	2			1		1		1
	3	1	1		1		1	
	4			1		1		1
	5			1		1		1
	6	1	1		1		1	

(a)

		Pièces						
		3	5	7	1	2	4	6
Machines	2	1	1	1				
	4	1	1	1				
	5	1	1	1				
	1				1	1	1	1
	3				1	1	1	1
	6				1	1	1	1

(b)

Pour ces deux configurations présentées au Tableau 3.5, l'énergie de liaison α ainsi que l'efficacité de regroupement sont calculées comme suit:

Configuration (a) : $\alpha_a = (3+3)/21 = 0.2857$

$e_1 = 11 / (9+12) = 0.5238$

$e_2 = 1 - 11 / (11 + 10) = 0.4762$

D'où l'efficacité e_a est égale à $e_a = e_1 - e_2 = 0.0476$

Configuration (b): $\alpha_b = (15+14)/21 = 1.381$

$e_1 = 21 / (9+12) = 1$

$e_2 = 1 - 21 / (21 + 0) = 0$

D'où l'efficacité e_b est égale à $e_b = e_1 - e_2 = 1$

Le regroupement est alors parfait : la première cellule est composée des machines 2, 4 et 5 alors que la deuxième est composée des machines 1, 3 et 6.

La qualité du regroupement dans la configuration (a) est très mauvaise par rapport à la configuration (b) ($e_a << e_b$). Plus α augmente, plus la qualité de regroupement est meilleure et l'efficacité e tend vers 1.

En réalité, des groupes complètement indépendants, comme c'est le cas de l'exemple précédent, ne peuvent pratiquement pas être obtenus. Il est probable que quelques éléments non nuls soient à l'extérieur des groupes. Ces éléments sont appelés des éléments exceptionnels qui représentent des mouvements intercellulaires. La performance du regroupement est d'habitude mesurée par la quantité de mouvements intercellulaires et la densité à l'intérieur des groupes. L'exemple suivant illustre un problème de regroupement avec plusieurs éléments exceptionnels (Tableau 3.6).

Tableau 3.6 Problème de regroupement avec des éléments exceptionnels

		Pièces						
		1	2	3	4	5	6	7
Machines	1	1	1		1			
	2			1	1	1		1
	3	1	1	1	1		1	
	4		1			1	1	1
	5			1				1
	6	1	1				1	

(a)

		Pièces						
		3	5	7	1	2	4	6
Machines	2	1	1	1			1	
	4		1	1		1		1
	5	1		1				
	1				1	1	1	
	3	1			1	1	1	1
	6				1	1		1

(b)

Pour les configurations présentées au Tableau 3.6, l'énergie de liaison ainsi que l'efficacité sont calculées comme suit;

Configuration (a) : $\alpha_a = (9+6)/21 = 0.7149$

$e_1 = 11 / (9+12) = 0.5238$

$e_2 = 1 - 11 / (11 + 10) = 0.4762$

D'où l'efficacité e_a est égale, $e_a = e_1 - e_2 = 0.0476$

Configuration (b): $\alpha_b = (9+9)/21 = 0.8571$

$e_1 = 17 / (9+12) = 0.8095$

$e_2 = 1 - 17 / (17 + 4) = 0.1905$

D'où $e_b = e_1 - e_2 = 0.619$

Même avec la présence des éléments exceptionnels, la configuration (b) est supérieure que la configuration (a) ($e_a << e_b$).

3.3.2 Méthodologie proposée

3.3.2.1 Le grand déluge étendu

Application de GDE sur le regroupement cellulaire

Afin d'adapter le GDE au problème du regroupement cellulaire, il est nécessaire de spécifier le type de voisinage adopté. La recherche procède d'une façon itérative d'une solution faisable à une autre en effectuant des mouvements dans le voisinage. En ce qui concerne ce problème, le mouvement dans le voisinage qui a été adopté se fait de la façon suivante :

Étape 1. Générer un nombre aléatoire p entre 0 et 1;

Étape 2. Si $p > 0.5$

i. sélectionner aléatoirement deux colonnes d'indices i et j de la matrice incidente;
ii. permuter les deux colonnes d'indices i et j;
iii. aller à l'étape 4;

Étape 3. Si $p \leq 0.5$

i. sélectionner aléatoirement deux lignes d'indices i et j de la matrice incidente;
ii. permuter les deux lignes d'indices i et j;
iii. aller à l'étape 4;

Étape 4. Fin.

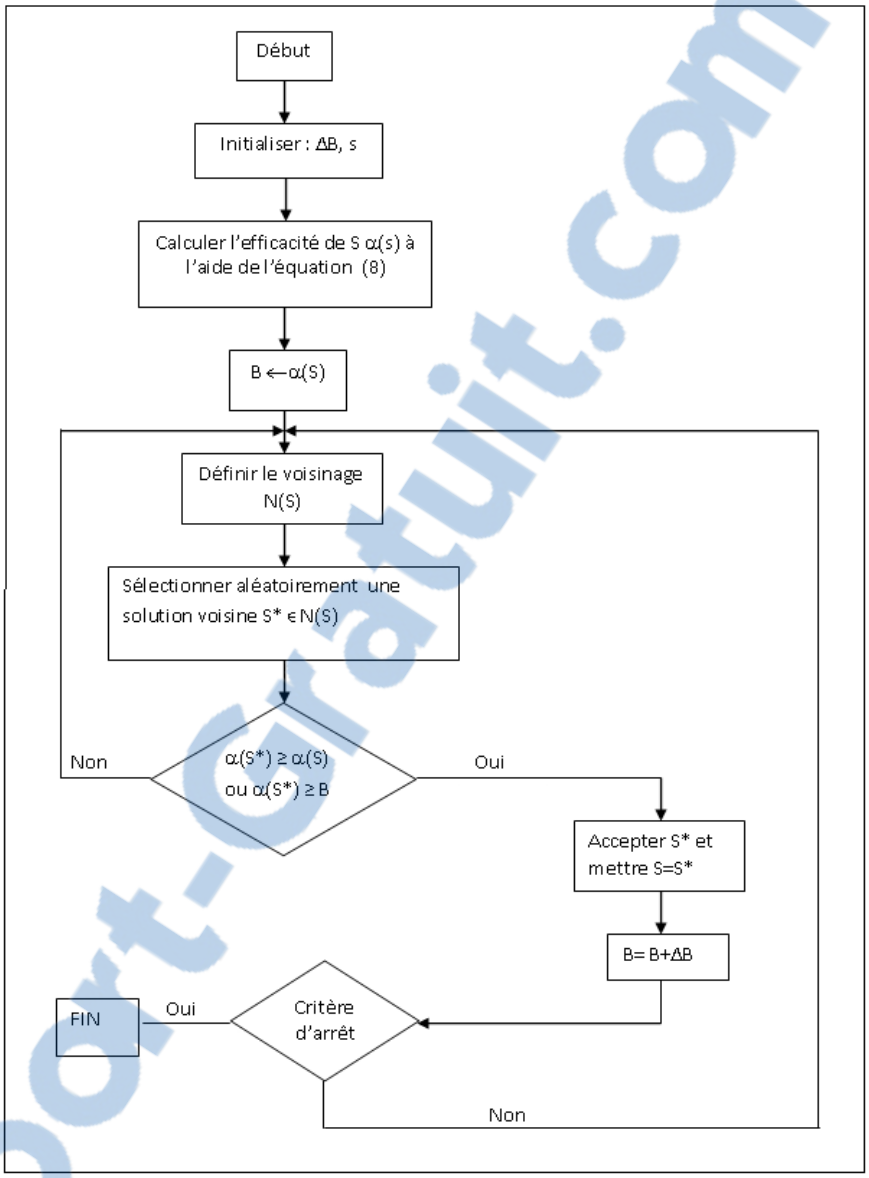


Figure 3.1 Algorithme pour le regroupement cellulaire basé sur le GDE.

La figure 3.1 présente les étapes du *GDE* appliquées au problème du regroupement cellulaire. Au début, il faut initialiser le taux de croissance ΔB afin d'obtenir une bonne limite B à chaque itération. Il faut initialiser aussi la solution S qui représente la séquence de pièces initiale. Dans notre cas, la séquence initiale est sélectionnée d'une façon aléatoire. Puis, une limite B est fixé en fonction de la solution initiale S, de là, l'énergie de liaison $\alpha(s)$ est calculé à l'aide de l'équation (3.8) présentée dans le paragraphe 3.3.1. Puis, la valeur de $\alpha(s)$ trouvée

est affecté à *B*. Ensuite, une solution voisine S^* est sélectionnée d'une façon aléatoire du voisinage *N(S)* (la sélection de la solution voisine est décrite ci-dessus), le voisinage *N(S)* est l'ensemble de solutions obtenues par la combinaison de la série de pièces et de machines. Pour qu'une solution soit acceptée ou pas, son énergie de liaison $\alpha(S^*)$ doit être comparée avec l'énergie de liaison de la solution précédente $\alpha(S)$ et la valeur de la limite *B*. Si $\alpha(S^*)$ est supérieure ou égale à $\alpha(S)$ ou bien supérieure ou égale à *B,* alors cette solution voisine *S** est acceptée. Sinon, il faut sélectionner une nouvelle solution voisine tout en reprenant le calcul et la comparaison de son énergie de liaison avec l'énergie de liaison de la solution précédente et *B*. Si la solution voisine *S** est acceptée, alors elle sera gardée et la limite B est augmentée par une valeur de ΔB. Si le critère d'arrêt est atteint, qui est le nombre d'itérations, alors le processus est fini et la solution optimale est la dernière solution acceptée. Sinon, une nouvelle solution voisine *S** est sélectionnée, et tout le processus de recherche se répète.

Exemple

Dans cette partie nous allons poursuivre le même exemple traité dans la sélection du routage de fabrication optimal (paragraphe 3.2.2) proposé par Liao et al. [39]. Après la résolution du problème de sélection du meilleur routage avec le logiciel LINGO, et après l'obtention d'une matrice binaire qui présente l'affectation des opérations de chaque pièce sur les machines, nous devons déterminer les cellules de fabrication. Pour cet objectif, nous avons utilisé l'algorithme GDE. La matrice incidente obtenue après sélection du meilleur routage de cet exemple est présentée dans le Tableau 3.7. Dans cette configuration, l'énergie de liaison α calculée par l'équation (3.8) est égale à 0.14. Cette faible valeur est une représentation de la mauvaise configuration de l'atelier de fabrication où les machines sont loin les unes des autres et les pièces ne sont pas regroupées par familles. Ce problème majeur dans les ateliers de fabrication augmente le prix et le temps de manutention.

Tableau 3.7 Matrice incidente initiale

		Pièces								
		1	2	3	4	5	6	7	8	9
Machines	A	0	0	0	1	0	0	1	0	0
	B	1	1	1	0	0	0	0	1	0
	C	0	0	0	0	0	1	0	0	1
	D	0	0	0	1	1	0	0	0	0
	E	0	1	0	0	0	0	0	1	0
	F	0	0	0	0	0	1	0	0	1
	G	1	0	0	1	0	0	1	0	0
	H	0	0	0	0	1	0	0	1	0
	I	0	0	1	0	0	1	0	0	0

La solution obtenue par l'algorithme GDE, tableau 3.8, est la même solution proposée par Liao et al.[39] où nous avant obtenus 3 cellules de fabrication : la première est formée par les machines A, D et G pour la fabrication des pièces 1, 4 et 7; la deuxième cellule est formée par les machines B, E et H avec une famille de pièces formée par 2, 5 et 8; et finalement, les machines C, F et I forment la troisième cellule de fabrication spécialisée pour la réalisation des pièces 3, 6 et 9. Cette configuration contient des éléments exceptionnels qui nécessitent un transfert intercellulaire, ces éléments sont les pièces 1, 3 et 5. La matrice diagonale de temps opératoire correspondante à ce regroupement est présentée dans le tableau 3.9 Ces résultats sont obtenus avec un taux de croissance $\Delta B=0,1$ et un nombre d'itérations égale à 10^5 itérations qui caractérisent ainsi notre critère d'arrêt.

La qualité du regroupement obtenu est évaluée par l'efficacité e; dans notre exemple, la densité cellulaire e_1 est égale à 0.6667 et le flux intercellulaire e_2 est égale à 0.1429 où $e = e_1 - e_2 = 0.5238$. Cette importante valeur e est due à une bonne énergie de liaison α qui est égale à 1,047.

Tableau 3.8 Le résultat final de regroupement par l'algorithme GDE

		Pièces								
		2	8	5	4	7	1	3	6	9
Machines	C	0	0	0	0	0	0	0	1	1
	F	0	0	0	0	0	0	0	1	1
	I	0	0	0	0	0	0	1	1	0
	B	1	1	0	0	0	1	1	0	0
	E	1	1	0	0	0	0	0	0	0
	H	0	1	1	0	0	0	0	0	0
	D	0	0	1	1	0	0	0	0	0
	A	0	0	0	1	1	0	0	0	0
	G	0	0	0	1	1	1	0	0	0

Tableau 3.9 Matrice diagonale de temps opératoires après regroupement avec le GDE

		Pièces								
		3	6	9	2	8	5	4	7	1
Machines	C	0	4	6	0	0	0	0	0	0
	F	0	2	3	0	0	0	0	0	0
	I	3	3	0	0	0	0	0	0	0
	B	5	0	0	3	2	0	0	0	6
	E	0	0	0	2	2	0	0	0	0
	H	0	0	0	0	3	2	0	0	0
	D	0	0	0	0	0	4	2	0	0
	A	0	0	0	0	0	0	2	7	0
	G	0	0	0	0	0	0	2	2	3

3.3.2.2 La méthode du recuit simulé

Description générale du recuit simulé

D'abord, un paramètre fictif est introduit, qui représente la température T du système, le pas ΔT et la solution initiale. Cette dernière est définie comme une séquence de pièces choisie aléatoirement. Puis, une solution voisine à la solution précédente est sélectionnée de façon

aléatoire avec une modification élémentaire. Si la solution sélectionnée améliore l'énergie de liaison $\alpha(s)$, alors elle sera acceptée. Si la solution sélectionnée dégrade l'énergie de liaison $\alpha(s)$, soit elle sera acceptée avec une probabilité p générée aléatoirement, soit elle sera rejetée. Ensuite, la température est diminuée et une nouvelle itération commence jusqu'à la fin de l'exécution.

Application du recuit simulé

Le type de voisinage adopté pour le RS est exactement celui utilisé pour le GDE.
L'algorithme du RS appliqué au problème de regroupement cellulaire peut se résumer de la façon suivante:

Étape 1. Fixer une température initiale $T=T_0$, ΔT et la solution initiale s;

Étape 2. Calculer l'énergie de liaison $\alpha(s)$ à l'aide de l'équation (3.8);

Étape 3. Définir le voisinage $N(s)$;

Étape 4. Sélectionner aléatoirement une solution voisine s^* de $N(s)$;

Étape 5. Calculer $\alpha(s^*)$

i. si $\alpha(s^*) \geq \alpha(s)$ alors accepter la solution et $s=s^*$;

ii. si $\alpha(s^*) < \alpha(s)$ alors générer au hasard un nombre p dans [0,1];

- ✓ si p > $exp(-(\alpha(s)- \alpha(s^*))/T)$ alors rejeter la solution et aller à étape 7;
- ✓ Sinon accepter la solution et $s = s^*$;

Étape 6. Diminuer $T=T - \Delta T$ et aller à étape 3;

Étape 7. Si le nombre maximal d'itérations n'est pas atteint, aller à Étape 3.

Exemple illustratif

La solution générée par le RS est présentée au tableau 3.10. La matrice diagonale de temps opératoire correspondante à ce regroupement est présentée dans le tableau 3.11. Le RS donne les mêmes familles de machines que celle obtenue avec l'algorithme GDE et celle proposée

par Liao et al.[39], mais avec des familles de pièces différentes. Nous obtenons toujours 3 cellules de fabrication, la première est formée par les machines A, D et G pour la fabrication des pièces 4, 5 et 7; la deuxième cellule est formée par les machines B, E et H avec une famille de pièces formée par les pièces 1, 2 et 8; et finalement, les machines C, F et I forment la troisième cellule de fabrication spécialisée pour la réalisation des pièces 3, 6 et 9. Avec cette configuration, les éléments exceptionnels qui nécessitent un transfert intercellulaire sont les même que ceux de la solution obtenue avec l'algorithme GDE et proposée par Liao et al. [39]. Ces éléments sont les pièces 1, 3 et 5. Ces résultats sont obtenus avec une température T=4, un pas $\Delta T=2x10^{-5}$ et un nombre d'itérations de $2x10^{5}$ itérations qui caractérisent toujours notre critère d'arrêt.

Tableau 3.10 Le résultat final de regroupement par le RS

		Pièces								
		1	2	8	7	4	5	3	6	9
Machines	I	0	0	0	0	0	0	1	1	0
	C	0	0	0	0	0	0	0	1	1
	F	0	0	0	0	0	0	0	1	1
	E	0	1	1	0	0	0	0	0	0
	B	1	1	1	0	0	0	1	0	0
	H	0	0	1	0	0	1	0	0	0
	D	0	0	0	0	1	1	0	0	0
	A	0	0	0	1	1	0	0	0	0
	G	1	0	0	1	1	0	0	0	0

Tableau 3.11 Matrice diagonale de temps opératoire après regroupement avec le RS

		Pièces								
		3	6	9	1	2	8	7	4	5
Machines	I	3	3	0	0	0	0	0	0	0
	C	0	4	6	0	0	0	0	0	0
	F	0	2	3	0	0	0	0	0	0
	E	0	0	0	0	2	2	0	0	0
	B	3	0	0	6	3	2	0	0	0
	H	0	0	0	0	0	3	0	0	2
	D	0	0	0	0	0	0	0	2	4
	A	0	0	0	0	0	0	7	2	0
	G	0	0	0	3	0	0	2	2	0

Le regroupement donné par le recuit simulé est évalué par l'efficacité *e*. La densité cellulaire e_1 est égale à 0.6667 et le flux intercellulaire e_2 est égal à 0.1429. L'efficacité du regroupement obtenu est égale à $e = e_1 - e_2 = 0.5238$ et elle est la même que celle obtenue avec l'algorithme GDE, mais l'énergie de liaison α qui est égale à 0.904 est inférieure à celle donnée par l'algorithme GDE qui est égale à 1.047. Cela montre une faible capacité de regroupement de l'algorithme de RS par rapport à celle de l'algorithme GDE.

3.4 Conclusion

Dans ce travail, nous avons présenté les deux étapes à suivre dans la formation des cellules de production. Premièrement, nous avons présenté un modèle linaire [39] pour la sélection du routage de fabrication optimal afin d'obtenir la matrice initiale binaire incidente. Un exemple constitué de 9 machines/9 pièces a été traité. Deuxièmement, nous avons appliqué l'algorithme de GDE et le RS pour un problème de regroupement. Les algorithmes ont été testés sur la même matrice incidente binaire obtenue à la première étape. Pour cet exemple (9 machines/9 pièces), l'algorithme de GDE a montré une meilleure efficacité de regroupement par rapport à l'algorithme de RS. Malgré que nous avons eu la même qualité de regroupement, le GDE a donné une énergie de liaison entre les éléments regroupés de la matrice supérieure à celle obtenue par le RS. Ce résultat permet de supposer que le GDE sera

plus efficace pour des problèmes de regroupement plus grands. Dans le chapitre 5, le GDE est appliqué et comparé avec d'autres méthodes de regroupement pour des problèmes beaucoup plus complexes.

CHAPITRE 4

ORDONNANCEMENT CELLULAIRE

4.1 Introduction

Une fois les machines et les pièces regroupées, le problème restant est de savoir comment déterminer les séquences de pièces, de sorte que certains objectifs puissent être atteints. Ce type de problème est appelé ordonnancement cellulaire et il sera analysé dans ce chapitre. La maximisation de l'utilisation des machines, l'équilibrage de charge et la minimisation des temps de retard font partie des objectifs les plus importants dans l'ordonnancement cellulaire. Dans ce chapitre, l'accent est mis sur la minimisation du temps de retard. Un système manufacturier cellulaire se compose d'un nombre de cellules de machines. Chaque cellule est principalement réservée pour réaliser la famille de pièces qui lui a été assignée dans la phase de formation. Les pièces provenant d'autres cellules sont également autorisées à être traitées dans cette cellule. Avant le traitement de chaque famille de pièces, une installation majeure sur chaque machine doit avoir lieu et pour chaque opération d'une pièce, de même qu'un temps d'installation mineur est requis. L'objectif est de trouver la meilleure séquence pour toutes les familles de pièces afin de réduire le makespan.

4.2 Solution proposée

Dans cette section, nous allons présenter la solution proposée pour résoudre les problèmes d'ordonnancement en tenant compte des éléments exceptionnels.

4.2.1 Ordonnancement

Ce problème consiste à trouver un ordonnancement optimal pour chacune des cellules obtenues d'une matrice incidente. Dans le cas où il n'existerait pas d'éléments exceptionnels (éléments à l'extérieur des cellules), l'ordonnancement se fera sur chacune des cellules

indépendamment les unes des autres. Dans le cas où il existerait des éléments exceptionnels, il faudrait tenir compte de toutes les opérations non incluses dans les cellules.
Pour résoudre ce problème, nous proposons les étapes suivantes :
Étape 1 : trouver l'ordonnancement optimal pour chacune des cellules indépendamment des autres en se basant sur les travaux de Sridhar et Rajendran [34].
Étape 2 : introduire la durée de tous les éléments exceptionnels dans le calcul du makespan.
Étape 3 : améliorer la solution obtenue en éliminant les temps morts.

4.2.2 Méthode de recherche itérative

Le processus de recherche itératif commence toujours par une solution initiale réalisable. Une solution initiale est une séquence de pièces définie aléatoirement d'une cellule où nous calculons le makespan correspondant. Une solution voisine est recherchée à chaque itération afin de raffiner la solution initiale tout en minimisant le makespan. À chaque itération, deux étapes sont réalisées, la recherche de voisinage, ensuite le calcul du makespan correspondant. Le voisinage est une modification élémentaire par rapport à la solution (la séquence de pièces pour chaque cellule) précédente; la solution voisine est obtenue en permutant les positions de deux (2) pièces choisies aléatoirement d'une séquence de pièces appartenant à une cellule. Ces deux étapes sont expliquées ci-dessous.

4.2.3 Solution voisine

À chaque itération, la solution voisine est déterminée par les étapes suivantes:

- Étape 1 : sélectionner aléatoirement une position i dans la séquence de pièces qui l'a précédée.
- Étape 2 : choisir aléatoirement une position j ($i \neq j$) dans la même séquence.
- Étape 3 : permuter la pièce située à la $i^{ème}$ position avec la quelle du $j^{ème}$ position.

Exemple : soit une solution caractérisée par une séquence de cinq (5) pièces :

s= 1 2 3 4 5

La solution voisine (s') est la suivante : soit i=2 et j=5 choisies aléatoirement, la solution devient alors :

s'=1 5 3 4 2

La recherche du voisinage est un problème de type NP-difficile, car elle provoque une explosion combinatoire en fonction de la longueur de la séquence.

La recherche du voisinage par cette méthode appartient à la classe des problèmes NP-difficile, ce qui signifie que l'augmentation de la longueur de la séquence entraînera une augmentation exponentielle du temps de calcul. Pour cela, nous avons utilisé le GDE pour la recherche de la meilleure solution qui minimise le makespan pour chaque cellule de fabrication.

4.2.4 Calcul du makespan

Pour trouver l'ordonnancement optimal pour chacune des cellules indépendamment les unes des autres, nous avons utilisé une heuristique proposée par Sridhar et Rajendran [34]. Les paramètres fondamentaux utilisés dans cette heuristique sont les suivants :

n	nombre de pièces ;
m	nombre de machines;
t_{ij}	le temps de fabrication de la pièce i sur la machine j;
s	l'ensemble des pièces réalisées;
a	représente les pièces non réalisées;
$q(s, j)$	délai d'achèvement de l'ensemble s sur la machine j;
$q(sa, j)$	délai d'achèvement de la pièce a sur la machine j quand a est ajoutée à l'ensemble s;
F_s	représente le *flowtime* de production de toutes les pièces de l'ensemble s;

Le délai d'achèvement *q(sa, j)* du plan partiel *sa* sur la machine *j* est déterminé par l'équation suivante :

$$q(sa, j)=\max(q(s, j);\ q(sa, j\text{-}1))+ t_{ij} \tag{4.12}$$

Le *flowtime* des pièces dans le plan partiel *sa* est déterminé comme suit :

$$F_{sa}= F_s+ q(sa, m) \tag{4.13}$$

À la suite de l'identification des paramètres, l'heuristique proposée par Sridhar et Rajendran [34] permet de déterminer le makespan et le *flowtime* d'une séquence donnée. L'heuristique est la suivante :

Initialiser T, F_s et M_s à 0.

Pour j=1 à m faire

Si $t_{aj} > 0$

Alors

Calculer les délais d'achèvement du plan partiel *sa* pour l'opération actuelle

$$q(sa, j)=\max[q(s, j);\ T]+ t_{aj}$$

Ensuite, mise à jour de T

$$T= q(sa, j)$$

Sinon

Faire

$$q(sa, j)= q(s, j)$$

Le *flowtime* de fabrication des pièces du plan *sa* est :

$$F_{sa}=F_s+T$$

Puisque T est le délai d'achèvement de la dernière opération de la pièce *a*. Le makespan du plan partiel *sa* est le suivant :

$$M_{sa}=\max(M_s,T)$$

4.2.5 Adaptation de la méthode proposée

En ce qui a trait aux cellules, la sélection de la meilleure séquence minimisant le makespan est effectuée par l'algorithme GDE. Nous nous sommes servi des mêmes étapes que celles qui ont été utilisées pour le regroupement, mais en tenant compte d'une autre méthode de recherche de voisinage (s^*) ainsi que la méthode utilisée pour le calcul de la fonction « objectif » (M_s). La figure 4.1 décrit l'algorithme GDE qui minimise le makespan.

Il est nécessaire d'effectuer des tests afin de bien choisir le paramètre de décroissance ΔB (la valeur de B diminue au cours de la recherche car nous sommes devant un problème de minimisation de makespan). Au départ, il faut initialiser la valeur ΔB, le nombre de cellules C et le nombre d'itérations *iter*. L'étape initiale consiste à sélectionner aussi une solution faisable s caractérisée par une séquence de pièces (des pièces affectées à une seule cellule) généré aléatoirement. Ensuite, il est nécessaire de calculer la fonction « objectif » (le makespan noté M_s) liée à la séquence initiale s à l'aide de la méthode proposée dans 4.2.4. La valeur de la limite supérieure B prend la même valeur que M_s. Ces donnés sont considérés comme des entrés pour le processus de recherche. Par la suite, une solution voisine est sélectionnée aléatoirement (paragraphe 4.2.3), son makespan sera calculé et comparé avec le makespan de la solution précédente et avec la limite B. Cette solution sera acceptée sauf si l'une des deux conditions est vraie, il faut alors que M_s^* soit inferieur à M_s ou à B, sinon une nouvelle solution voisine sera sélectionnée. Une fois que la solution actuelle vérifie l'une de ces conditions, elle sera gardée comme une solution optimale et la valeur de B sera réduite

d'une valeur ΔB. Si le nombre d'itération n'est pas encore atteint, le processus de recherche pour la cellule actuelle se reproduit. Si le nombre d'itérations est atteint, la recherche de la meilleure séquence est finie pour la cellule actuelle et la dernière solution acceptée est la solution optimale. Une nouvelle procédure est alors lancée pour la cellule suivante. Si le nombre de cellule est atteint, c'est la fin du programme.

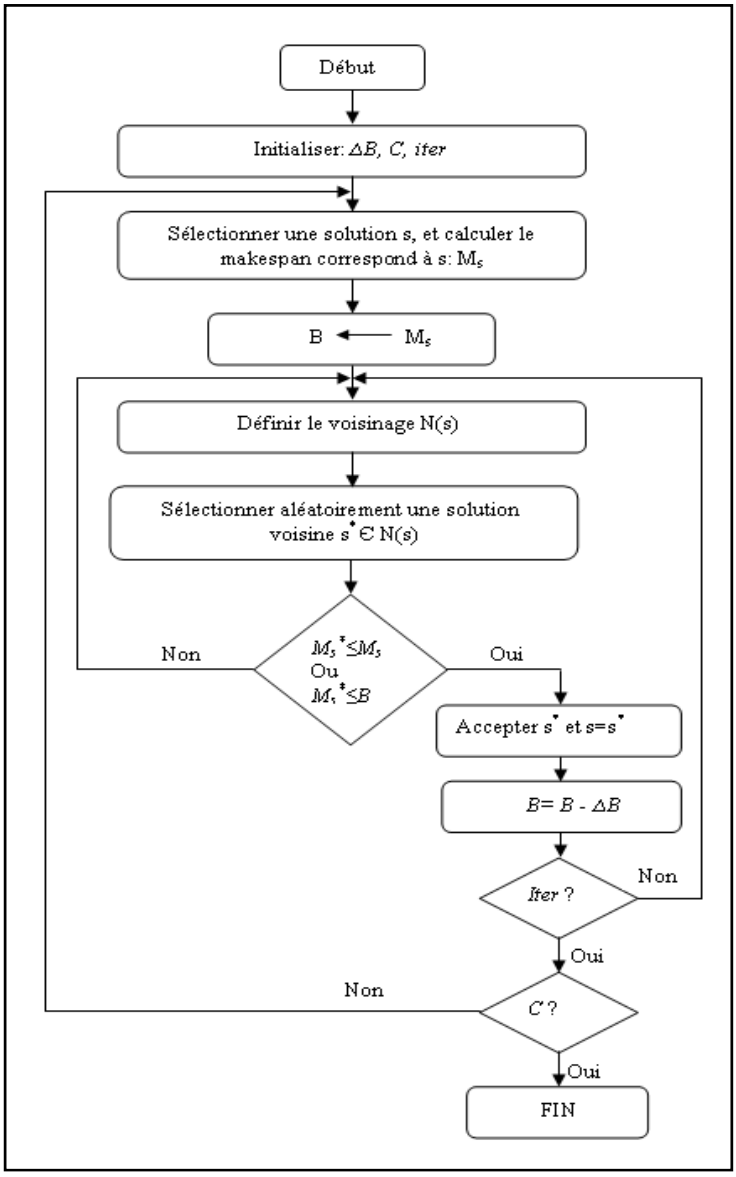


Figure 4.1 Algorithme pour l'ordonnancement cellulaire avec le GDE.

4.2.6 Traitement des éléments exceptionnels

Nous proposons une méthode pour traiter les éléments exceptionnels ainsi qu'une méthode d'amélioration de la solution obtenue.

4.2.6.1 Première solution faisable

En premier lieu, nous avons traité le problème d'ordonnancement des cellules avec des éléments exceptionnels dans le but de traiter toutes les opérations en utilisant les machines disponibles dans l'entreprise par un transfert intercellulaire. Après l'étape de regroupement des cellules, deux ensembles d'éléments exceptionnels peuvent être sélectionnés comme il est expliqué par l'exemple ci-après. La figure 4.2 décrit l'algorithme proposé avec toutes les étapes qui minimisent le makespan tout en introduisant les éléments exceptionnels.

L'exemple présenté à la figure 4.3 sert à expliquer comment nous choisissons les éléments exceptionnels pour chaque ensemble. *P* et *P'* sont inclus dans les deux ensembles *D* et *F;* où; *D*= {1, 3, 6, 2, 5, 7} et *F*={4, 8, 9, 1, 3, 6}. Parmi les éléments de *D* et *F,* il y a des pièces qui ne nécessitent pas un transfert vers d'autres cellules pour réaliser des opérations. Les deux ensembles *P* et *P'* incluant seulement les éléments de D et F qui nécessitent ce transfert, alors *P*= {3, 6, 7} et *P'*= {4, 8, 1}. La pièce 3 est réalisée dans la cellule 2 sur les machines C, E et G mais elle doit être envoyée vers la cellule 1 pour réaliser une opération sur la machine A d'une durée de 8 unités de temps. De même, la pièce 4 qui appartient à la cellule 1 doit être transférée vers la cellule 2 pour réaliser une opération sur la machine G d'une durée de 3 unités de temps; de même pour toutes les pièces qui appartiennent à *P* et *P'*.

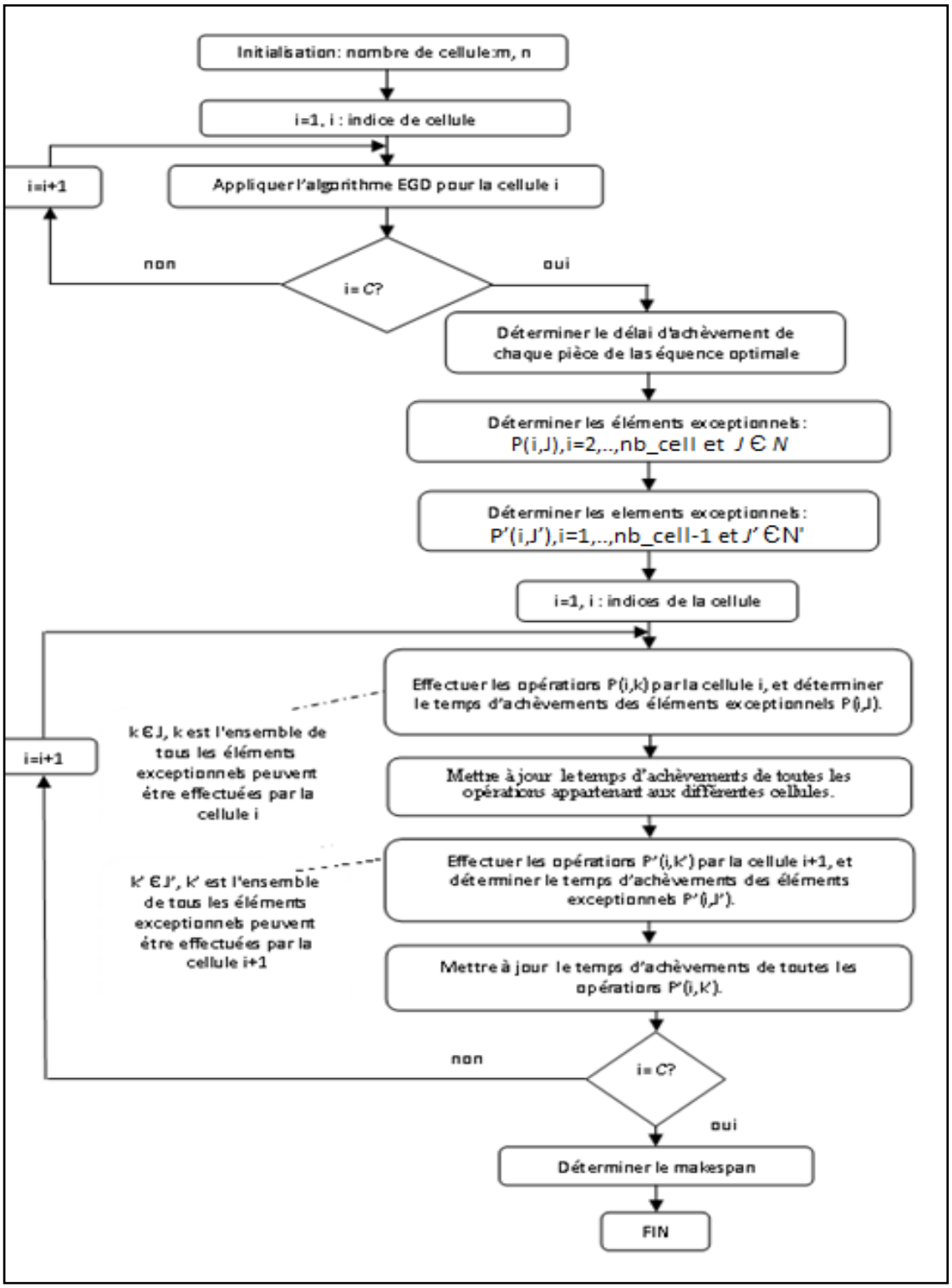


Figure 4.2 Algorithme d'ordonnancement cellulaire avec des éléments exceptionnels

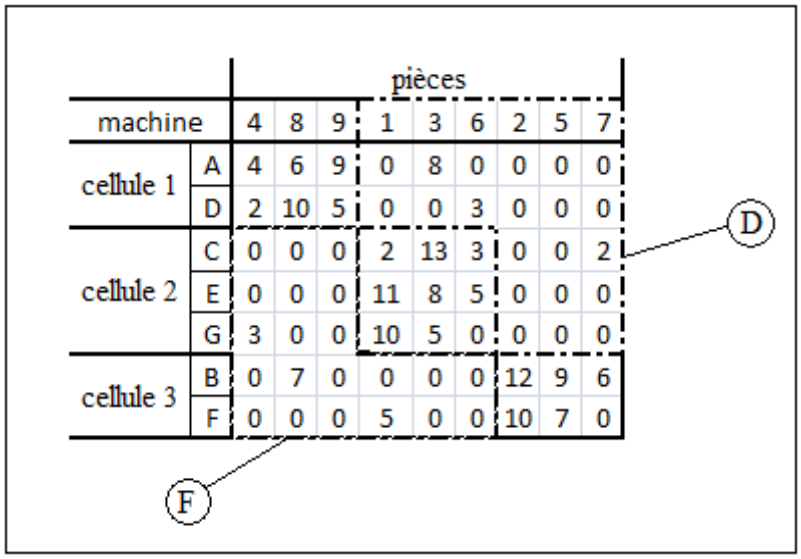


Figure 4.3 Schématisations des ensembles des éléments exceptionnels.

Trois étapes doivent être suivies afin de réaliser toutes les pièces. La première étape est de réaliser les éléments exceptionnels des pièces appartenant à l'ensemble *P*. Le makespan représente le temps d'achèvement de la dernière pièce de l'ensemble *P* : c'est le temps de retard avant le lancement de la production dans les cellules. Ce premier temps de retard sera ajouté au makespan trouvé après l'application de l'algorithme GDE sur chacune des cellules. La troisième étape est de finaliser les pièces qui appartiennent à l'ensemble *P'* où nous ajoutons ce deuxième temps de retard au makespan. La figure 4.4 est une représentation des étapes proposées.

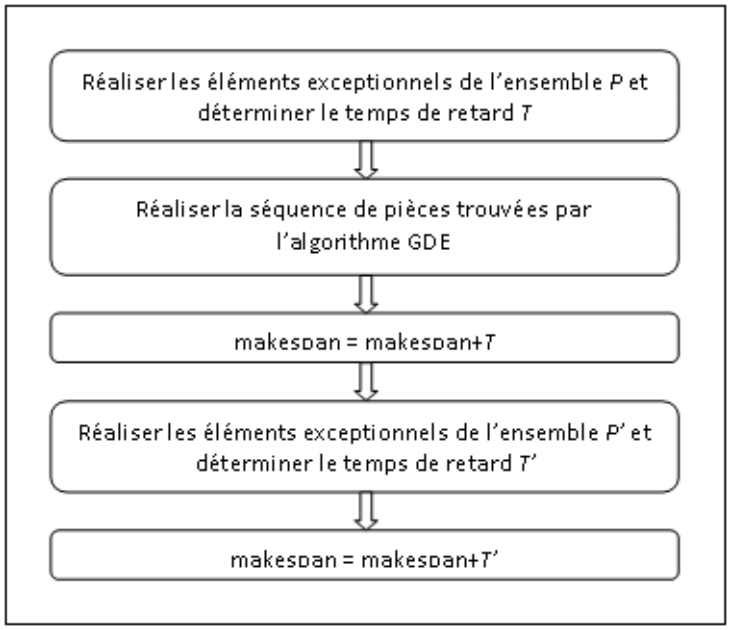


Figure 4.4 Étapes à suivre lors de l'ordonnancement

4.2.6.2 Procédure d'amélioration

La procédure d'amélioration permet de réduire le temps de repos des machines au maximum. Une machine commence le traitement d'une pièce dès qu'elle est libre à condition que la pièce soit présente. C'est le principe de la méthode. Dans certains cas, les machines restent au repos en attendant l'arrivée des pièces qui sont en cours de traitement sur d'autres machines et malheureusement nous ne pouvons rien faire pour cette situation.

L'approche qui optimise la séquence de pièces à l'intérieur des cellules en tenant compte des éléments exceptionnels avec la procédure d'amélioration se présente comme suit :

1) Introduire les données :
   - Le temps opératoire des familles de pièces T_{emps} ;
   - Le temps opératoire des éléments exceptionnels de P, T_{excep} ;
   - Le temps opératoire des éléments exceptionnels de P', T'_{excep} ;

- Le nombre de cellules *C*.

Les matrices T_{emps}, T_{excep} *et* T'_{excep} sont de même dimension (*m, n*). La matrice temps d'achèvement de toutes les pièces est notée par T_{ach}.
m_c et n_c sont le nombre de machines et le nombre pièces pour chaque cellule.

2) Appliquer l'algorithme de GDE et déterminer la matrice temps d'achèvement des opérations des familles de pièces (paragraphe 4.2.4, équation 4.12), nous la notons par:

$T_{ach\_cellule}(i, j)$ avec $i = 1 \ldots m$; indices des machines ;
$j = 1 \ldots n$; indices des pièces.

3) Réaliser les éléments exceptionnels de P et déterminer la matrice d'achèvement des opérations de P (paragraphe 3.2.4, équation 4.12), nous la notons par :

$T(i, j)$ avec $i = 1 \ldots m$; indices des machines ;
$j = 1 \ldots n$; indices des pièces.

et $T_{ach}(i, j) = T(i, j)$

4) Effectuer la séquence de pièces donnée par le GDE et mettre à jour la matrice temps d'achèvement des opérations T_{ach}:

- Pour la première cellule : c=1

  Pour la première machine: i=1 , j=1 à n_1 et $k = n_1 + 1$ à n

  $$T_{ach}(1, j) = T_{ach\_cellule}(1, j) + max\,(T_{ach}(1, k))$$

  Pour i=2 à m_1

  - Pour la première pièce de chaque séquence: j=1 et $k = n_1 + 1$ à n

    $$T_{ach}(i, 1) = T_{emps}(i,1) + max\,[T_{ach}(i\text{-}1, 1)\,,\, max(T_{ach}(i,k))]$$

  - Pour le reste des pièces : j=2 à n_1

    $$T_{ach}(i, j) = T_{emps}(i, j) + max\,[T_{ach}(i\text{-}1, j)\,,\, T_{ach}(i, j\text{-}1)]$$

- Pour c = 2 à C-1

  Pour la première machine: $i=m_{c-1}+1$

  - Pour $j= n_{c-1}+1$, $k= n_c +1$ à n et $l=1$ *à* m_{c-1}

    $$T_{ach}(i, j) = T_{emps}(i,j) + max\,[max(T_{ach}(i, k)) , max(T_{ach}(l,j))]$$

  - Pour $j= n_{c-1}+2$ à n_c et $l=1$ *à* m_{c-1}

    $$T_{ach}(i, j) = T_{emps}(i,j) + max\,[T_{ach}(i, j\text{-}1), max(T_{ach}(l,j))]$$

  Pour $i= m_{c-1}+2$ à m_c

  - Pour $j= n_{c-1}+1$ et $k= n_c +1$ à n

    $$T_{ach}(i, j) = T_{emps}(i, j) + max\,[T_{ach}(i\text{-}1, j) , max\ T_{ach}(i,k)]$$

  - Pour $j= n_{c-1}+2$ à n_c

    $$T_{ach}(i, j) = T_{emps}(i, j) + max\,[T_{ach}(i\text{-}1, j) , T_{ach}(i, j\text{-}1)]$$

- Pour $c = C$

  Pour $i= m_{C-1}+1$

  - Pour $j= n_{C-1}+1$ et $l=1$ à m_{C-1}

    $$T_{ach}(i, j) = T_{ach\_cellule}(i, j) + max(T(l,j))$$

  - Pour $j= n_{C-1}+2$ à n et $l=1$ à m_{C-1}

    $$T_{ach}(i, j) = T_{emps}(i, j) + max\,[T_{ach}(i, j\text{-}1) , max(T(l,j))]$$

  Pour $i= m_{C-1}+2$ à m

  - Pour $j= n_{C-1}+1$

    $$T_{ach}(i, j) = T_{emps}(i, j) + T_{ach}(i\text{-}1, j)$$

  - Pour $j= n_{C-1}+2$ à n_c

    $$T_{ach}(i, j) = T_{emps}(i, j) + max\,[T_{ach}(i\text{-}1, j) , T_{ach}(i, j\text{-}1)]$$

5) Réaliser les éléments exceptionnels de P' et mettre à jour la matrice temps d'achèvement : la réalisation des éléments de P' se fait toujours selon la séquence optimale sélectionnée par le GDE.

   Notons que les opérations qui nécessitent des transferts intercellulaires sont identifiées par leurs coordonnées dans la matrice temps ; soit :

- r l'indice de la machine qui va réaliser l'élément exceptionnel k.
- $f = 1 \ldots r-1$: pour assurer que la pièce k est déjà libérée des autres machines.
- $w = 1 \ldots k-1$ et $w = w+1 \ldots n$: pour assurer que la machine r est libre pour réaliser l'opération (r, k) de la pièce k.

$$T_{ach}(r, k) = T'_{excep}(r,k) + max\,[T_{ach}(f, k), T_{ach}(r, w)]$$

6) Calculer le makespan et le *flowtime*

$$Makespan = max(T_{ach})$$

$$flowtime = \sum_{j=1}^{n} max[T_{ach}(i, j)], \text{ avec } i = 1 \ldots m$$

4.2.6.3 Exemple illustratif

Dans cette partie, nous avons expérimenté la méthode par l'exemple traité dans le chapitre précédent. La matrice diagonale obtenue par le GDE, présenté dans le Tableau 3.9, représente le temps opératoire de chaque pièce sur chaque machine. Nous avons appliqué l'approche proposée pour déterminer la meilleure séquence qui minimise le makespan avec les éléments exceptionnels 3, 5 et 1.

1) Donnés du problème :

- Le temps opératoire des familles de pièces T_{emps} est présenté dans le tableau 4.1.

Tableau 4.1 Temps opératoire T_{emps}

		Pièces								
		3	6	9	2	8	5	4	7	1
Machines	C	0	4	6	0	0	0	0	0	0
	F	0	2	3	0	0	0	0	0	0
	I	3	3	0	0	0	0	0	0	0
	B	0	0	0	3	2	0	0	0	0
	E	0	0	0	2	2	0	0	0	0
	H	0	0	0	0	3	2	0	0	0
	D	0	0	0	0	0	0	2	0	0
	A	0	0	0	0	0	0	2	7	0
	G	0	0	0	0	0	0	2	2	3

- Le temps opératoire des éléments exceptionnels de P, T_{excep} est présenté dans le tableau 4.2

Tableau 4.2 Temps opératoires T_{excep}

		Pièces								
		3	6	9	2	8	5	4	7	1
Machines	C	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	0	0	0	0
	I	0	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	0	0	0	6
	E	0	0	0	0	0	0	0	0	0
	H	0	0	0	0	0	0	0	0	0
	D	0	0	0	0	0	0	0	0	0
	A	0	0	0	0	0	0	0	0	0
	G	0	0	0	0	0	0	0	0	0

- Le temps opératoire des éléments exceptionnels de P', T'_{excep} est présenté dans le tableau 4.3

Tableau 4.3 Temps opératoires T'_{excep}

		Pièces								
		3	6	9	2	8	5	4	7	1
Machines	C	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	0	0	0	0
	I	0	0	0	0	0	0	0	0	0
	B	5	0	0	0	0	0	0	0	0
	E	0	0	0	0	0	0	0	0	0
	H	0	0	0	0	0	0	0	0	0
	D	0	0	0	0	0	4	0	0	0
	A	0	0	0	0	0	0	0	0	0
	G	0	0	0	0	0	0	0	0	0

2) Après l'application de l'algorithme du GDE, nous obtenons une matrice temps d'achèvement avec les séquences de pièces optimales pour les trois cellules, $T_{ach\_cellule}$, la matrice $T_{ach\_cellule}$ obtenue est présentée dans le tableau 4.4 et le diagramme de Gantt correspondant à l'ordonnancement des séquences optimales est présenté à la figure 4.5.

Tableau 4.4 Matrice temps d'achèvement $T_{ach\_cellule}$

		pièces								
		3	6	9	5	8	2	1	7	4
Machines	C	0	4	10	0	0	0	0	0	0
	F	0	6	13	0	0	0	0	0	0
	I	3	9	13	0	0	0	0	0	0
	B	0	0	0	0	2	5	0	0	0
	E	0	0	0	0	4	7	0	0	0
	H	0	0	0	2	7	7	0	0	0
	D	0	0	0	0	0	0	0	0	2
	A	0	0	0	0	0	0	0	7	9
	G	0	0	0	0	0	0	3	9	11

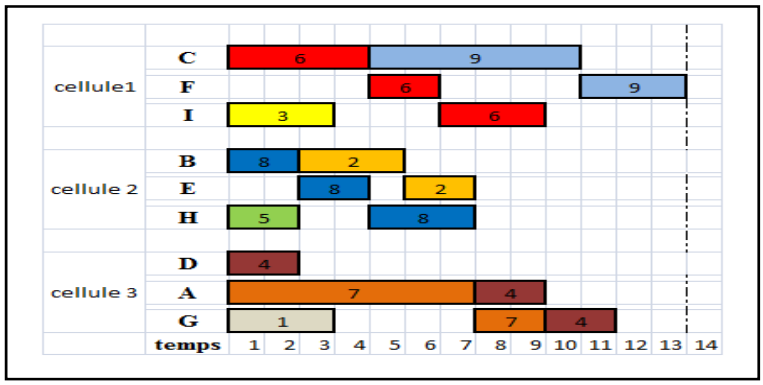


Figure 4.5 Ordonnancement des séquences optimales.

Les séquences optimales qui ont été obtenues par le GDE et qui minimisent le makespan à l'intérieur des cellules sont les suivantes : pour la cellule 1, la séquence de pièces optimale est [3, 6, 9], pour la cellule 2 la séquence optimale est [5, 8, 2], et pour la cellule 3 la séquence optimale est [1, 7, 4]. Le makespan obtenu sans les éléments exceptionnels est égal à 13.

3) Réalisation des éléments exceptionnels de P, et la matrice temps d'achèvement *T* est présentée au le tableau 4.5.

Tableau 4.5 Matrice temps d'achèvement *T*

		pièces								
		3	6	9	5	8	2	1	7	4
Machines	C	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	0	0	0	0
	I	0	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	0	(6)	0	0
	E	0	0	0	0	0	0	0	0	0
	H	0	0	0	0	0	0	0	0	0
	D	0	0	0	0	0	0	0	0	0
	A	0	0	0	0	0	0	0	0	0
	G	0	0	0	0	0	0	0	0	0

4) Réalisation des séquences optimales de pièces obtenues par le GDE et mise à jour de la matrice temps d'achèvement (T_{ach}) des opérations, la matrice T_{ach} est présentée au tableau 4.6 et l'ordonnancement des familles de pièces et des éléments exceptionnels de

P (pour notre cas, nous avons un seul élément, c'est la pièce 1) est présenté par le diagramme de Gantt figure 4.6, le makespan est à 17, ce qui explique que nous pouvons réaliser l'élément exceptionnel 1 avec un retard de 4 unités.

Tableau 4.6 Matrice temps d'achèvement T_{ach}

		Pièces								
		3	6	9	5	8	2	1	7	4
Machines	C	0	4	10	0	0	0	0	0	0
	F	0	6	13	0	0	0	0	0	0
	I	3	9	13	0	0	0	0	0	0
	B	0	0	0	6	8	11	6	0	0
	E	0	0	0	6	10	13	0	0	0
	H	0	0	0	8	13	13	0	0	0
	D	0	0	0	0	0	0	6	6	8
	A	0	0	0	0	0	0	6	13	15
	G	0	0	0	0	0	0	9	15	17

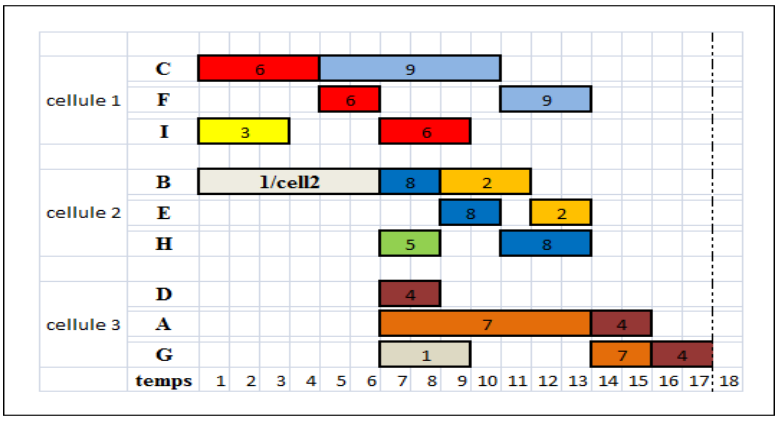


Figure 4.6 Ordonnancement des familles de pièces et des éléments exceptionnels de P.

5) Réalisation des éléments exceptionnels de P' et mise à jour de la matrice temps d'achèvement T_{ach}. À cette étape, le temps d'achèvement des opérations de l'ensemble P' est calculé en fonction de la disponibilité des machines et de la présence des pièces manquantes pour ces opérations. La matrice T_{ach} finale est présentée au tableau 4.7.

L'ordonnancement de toutes les pièces est représenté par le diagramme de Gantt figure 4.7.

Tableau 4.7 Matrice finale T_{ach}

		Pièces								
		3	6	9	5	8	2	1	7	4
Machines	C	0	4	10	0	0	0	0	0	0
	F	0	6	13	0	0	0	0	0	0
	I	3	9	13	0	0	0	0	0	0
	B	16	0	0	6	8	11	6	0	0
	E	0	0	0	6	10	13	0	0	0
	H	0	0	0	8	13	13	0	0	0
	D	0	0	0	12	0	0	6	6	8
	A	0	0	0	0	0	0	6	13	15
	G	0	0	0	0	0	0	9	15	17

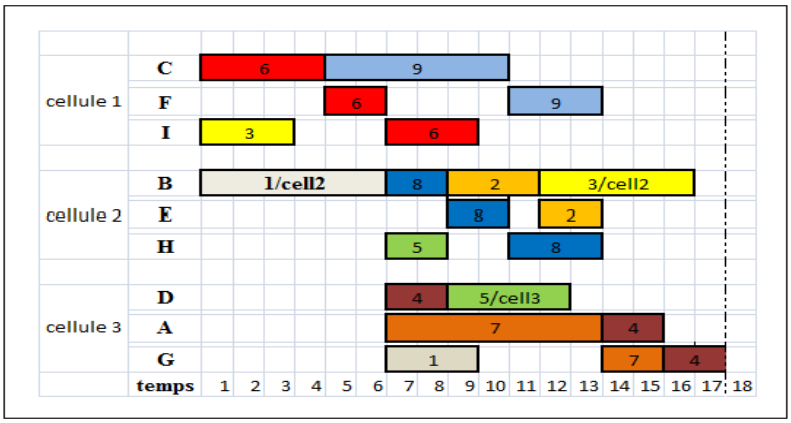


Figure 4.7 Solution améliorée.

6) Calcul de makespan et de *flowtime :* Après l'application de la procédure d'amélioration, les nouveaux makespan et *flowtime* sont les suivants :

$$Makespan = max(T_{ach}) = 17$$

$$flowtime = \sum_{j=1}^{9} \max[T_{ach}(i,j)], \text{ avec } i=1...9$$

$$=117$$

Le makespan calculé pour les familles de pièces obtenues par le GDE est égal à 13 si les éléments exceptionnels ne sont pas fabriqués dans les cellules de production. Par exemple, ils seront fabriqués en sous-traitance. Mais, dans notre cas, l'objectif essentiel est de fabriquer toutes les pièces en utilisant les machines qui constituent les cellules de fabrication. Nous pouvons avoir comme première solution faisable celle qui est présentée par le diagramme de Gantt figure 4.8. Pour cette solution, le temps de retard provoqué par les éléments exceptionnels est assez important, il est égal à 11, et le makespan s'élève à 24. Le temps d'achèvement de la première solution faisable est simple à déterminer, il est calculé par l'équation suivante :

$$T_{ach} = T_{ach\_cellule} + Temps\ de\ retard$$

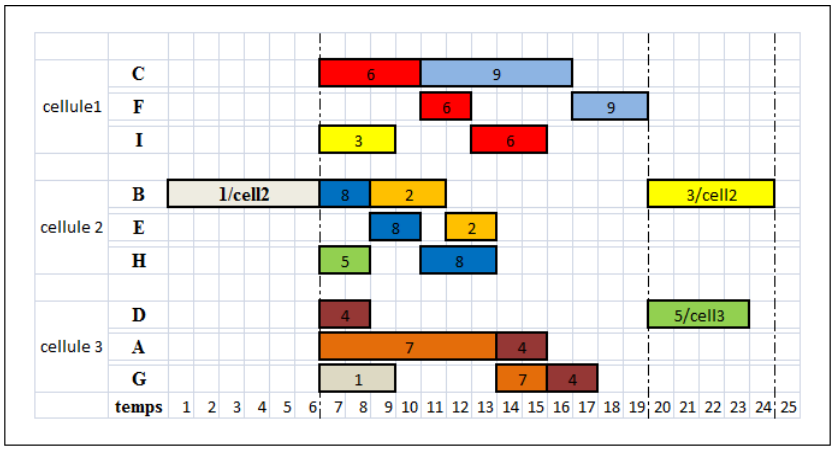


Figure 4.8 Première solution.

Après cette procédure d'amélioration, le makespan est réduit de 7 unités de temps pour cet exemple. Mais il a été augmenté à 17 par rapport à celui obtenu sans les éléments exceptionnels. Alors le prix à payer pour réaliser toutes les pièces à l'intérieur de l'usine est de 4 unités de temps de retard, ce qui pourrait être rentable pour l'entreprise sans prise de

risque de retard imprévisible de la sous-traitance qui peut provoquer des perturbations sur la chaîne de production.

4.3 Conclusion

Ce chapitre présente une approche développée à la base de l'algorithme de GDE pour des problèmes d'ordonnancement cellulaire en proposant une heuristique en vue d'améliorer la qualité de la solution obtenue. L'environnent de travail dans les cellules de production étudiées est de type *flowshop*.

Les résultats de l'exemple traité (9 machines/9 pièces) obtenus par simulation montre l'efficacité de l'algorithme de GDE et l'heuristique d'amélioration en ce qui a trait à la qualité de la solution et au temps de calcul puisque nous avons eu une solution acceptable en tenant compte des éléments exceptionnels dans un temps de résolution qui ne dépasse pas 15 s.

Vu la complexité des problèmes d'ordonnancement, l'approche proposée sera testée sur des problèmes beaucoup plus complexes dans le chapitre 5.

CHAPITRE 5

VALIDATION ET SYNTHÈSE

5.1 Introduction

Dans ce chapitre, nous présentons les résultats obtenus pour des problèmes de regroupement cellulaire en utilisant l'algorithme du RS et l'algorithme du GDE. Ensuite, nous présentons une partie de validation de l'algorithme du GDE ainsi que l'approche proposée pour l'ordonnancement cellulaire en tenant compte des éléments exceptionnels.

5.2 Regroupement cellulaire

Nous avons testé la performance des algorithmes GDE et RS dans des problèmes de regroupement cellulaire. Sept (7) problèmes ont été sélectionnés de la littérature, ils ont été traités pour tester la capacité de ces algorithmes à regrouper les éléments non nuls des matrices incidentes binaires correspondantes à ces problèmes.

Deux exemples de taille moyenne (pb1 et pb2) sont initialement proposés par Srinivasan et al.[67] et repris dans Mak et al.[46]. Le premier est composé de 10 machines et 20 pièces et le deuxième exemple est formé de 16 machines et 30 pièces. Les cinq autres exemples ont été proposés par Chandrasekharan et Rajagopalan [43], et repris par Mak et al.[46]; ils sont composés de 24 machines et 40 pièces. Ces exemples sont considérés comme des matrices de grande taille. Pour déterminer les bons paramètres pour les deux algorithmes, nous avons utilisé l'exemple composé de 10 machines et 20 pièces puisque la solution finale de cet exemple est un regroupement idéal où il n'y a pas d'éléments exceptionnels. Après plusieurs tests pour déterminer les bons paramètres, nous avons trouvé que les algorithmes atteignent leur meilleure performance et donnent de bons résultats avec les paramètres suivants : pour le GDE, la meilleure valeur du taux de croissance ΔB est égale à 0,1 avec un nombre d'itérations égal à 10^5. Pour le RS, les paramètres choisis sont : une température T égale à 4, un pas ΔT

égal à $2x10^{-5}$ et un nombre d'itérations limité à 10^6. Les algorithmes GDE et RS sont codés avec Matlab et exécutés avec un ordinateur Dual Core avec une vitesse du processeur de 1,86 GHz et 3 GB de mémoire vive (RAM).

Les problèmes ont été exécutés 10 fois avec les mêmes paramètres. Nous avons calculé la valeur moyenne ainsi que l'écart type de l'énergie de liaison α obtenus par les deux algorithmes. Les valeurs maximales, minimales de l'énergie de liaison et le temps de calcul sont présentées dans le tableau 5.1 et le tableau 5.2.

Tableau 5.1 Résultats obtenus par le recuit simulé

		RS				
Problèmes	**size**	α_{min}	α_{moy}	α_{max}	**écart type**	t_{moy}**(s)**
Pb1[67]*	10x20	1,2041	1,2571	1,3265	0,0399	7,48
Pb2[67]*	16x30	1,1379	1,1966	1,2414	0,0314	12,81
Pb3_dataset1[43]*	24x40	1,3435	1,387	1,4427	0,0286	22,43
Pb4_dataset2[43]*	24x40	1,2308	1,2738	1,3462	0,0321	23,08
Pb5_dataset3[43]*	24x40	1,0458	1,0801	1,1069	0,0201	22,36
Pb6_dataset4[43]*	24x40	1,0763	1,1107	1,145	0,0213	22,25
Pb7_dataset5[43]*	24x40	0,8168	0,8412	0,8626	0,0147	21,86

Tableau 5.2 Résultats obtenus par l'algorithme GDE

		GDE				
problèmes	**size**	α_{min}	α_{moy}	α_{max}	**écart type**	t_{moy}**(s)**
Pb1[67]	10x20	1,388	1,388	**1,388**	0	**6,51**
Pb2[67]	16x30	1,2672	1,2913	**1,3017**	0,0106	**8,86**
Pb3_dataset1[43]	24x40	1,4275	1,4825	**1,5115**	0,0227	**19,61**
Pb4_dataset2[43]*	24x40	1,3692	1,3985	**1,4231**	0,0553	28,45
Pb5_dataset3[43]*	24x40	1,1679	1,1809	**1,2061**	0,013	25,94
Pb6_dataset4[43]*	24x40	1,1832	1,1931	**1,2061**	0,0073	24,74
Pb7_dataset5[43]*	24x40	0,8626	0,884	**0,9084**	0,0143	31,13

(*) Nécessite une modification manuelle élémentaire afin de trouver une solution faisable.

La matrice des regroupements cellulaires du problème 1 donnée par l'algorithme du recuit simulé est celle présentée au tableau 5.3.

Nous présentons les solutions obtenus (les matrices binaires regroupées) par le GDE dans l'annexe.

Tableau 5.3 Matrice finale des regroupements cellulaires du Pb1 par le recuit simulé

		Pièces																			
		7	1	2	8	5	6	9	12	11	16	19	18	14	15	20	17	13	3	10	4
Machines	6	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	9	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
	10	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	2	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
	3	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

La solution présentée au tableau 5.3 peut être améliorée. En effet, en déplaçant les deux colonnes correspondantes aux pièces trois (3) et dix (10) vers la colonne correspondante à la pièce cinq (5), nous formons alors une cellule de deux (2) machines 2 et 3 pour réaliser la famille de pièces 2, 3, 5, 8 et 10. En déplaçant la colonne quatre (4) vers la colonne sept (7) ou un (1), nous obtenons une cellule de trois (3) machines 1, 4 et 6 pour la fabrication de trois (3) pièces 1, 4 et 7. Le tableau 5.4 représente la matrice finale des regroupements cellulaires après modifications.

Tableau 5.4 Matrice modifiée des regroupements cellulaires du Pb1

		Pièces																			
		7	1	4	2	8	5	3	10	6	9	12	11	16	19	18	14	15	20	17	13
Machines	6	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
	10	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0
	2	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Nous calculons la qualité des résultats obtenus avant et après modifications :

- L'efficacité du regroupement donné par le RS (tableau 5.3):

n_1=42

n_2=7

e_1=(6+6+18+12)/ (6+6+18+12) = 1

e_2= 1- (6+6+18+12)/(42+7) = 0.143

D'où $e = e_1 - e_2 = 0.857$

- L'efficacité du regroupement après amélioration (le tableau 5.4):

n_1=49

n_2=0

e_1= (9+10+18+12)/ (9+10+18+12) = 1

e_2= 1- (9+10+18+12)/(42+7) = 0

D'où $e = e_1 - e_2 = 1$

La qualité du regroupement est améliorée en passant de la première configuration (tableau 5.3) à la deuxième configuration (tableau 5.4). Alors, la modification améliore la qualité de regroupement obtenu.

À partir des tableaux 5.1 et 5.2, nous pouvons déduire que l'algorithme de GDE donne les meilleurs résultats par rapport au RS. Il est aussi le plus rapide pour résoudre les trois premiers problèmes mais pour les problèmes de grande taille (pb4, pb5, pb6 et pb7), le RS est plus rapide. De plus, sur les sept problèmes étudiés, l'algorithme de GDE a donné trois solutions qui ne nécessitent pas une modification manuelle (Pb1, Pb2 et Pb3) tandis que le RS n'a donné aucune solution qui ne nécessite pas cette modification.

Après ces modifications, nous avons calculé l'efficacité du regroupement trouvé pour chaque problème, en présentant les valeurs trouvées de l'efficacité minimale (e_{min}), maximale (e_{max}) et moyenne (e_{moy}) ainsi que l'écart type et la meilleure valeur de l'énergie de liaison (α_{best}) dans le tableau 5.5.

Ensuite, la qualité du regroupement obtenu par l'algorithme de GDE et celle qui a été obtenue par le recuit simulé sont toujours évaluées par l'efficacité e où nous obtenons les mêmes

résultats pour les problèmes sauf le problème Pb7_dataset5 proposé par Chandrasekharan et Rajagopalan [43]. L'algorithme du GDE donne un meilleur regroupement caractérisé par une efficacité *e* égale à 0,3536 qui est supérieure à celle qui a été obtenue par le recuit simulé qui est égale à 0,3486. La valeur de l'efficacité *e* trouvée à la suite de la configuration donnée par l'algorithme de GDE reflète une bonne énergie de liaison α par rapport à celle de la configuration donnée par le RS.

Nous avons comparé ces résultats avec ceux qui ont été obtenus par l'algorithme génétique [46] et l'algorithme ZODIAC [43]. Les résultats obtenus en utilisant le RS, l'algorithme du GDE, ZODIAC et les AG sont présentés dans le tableau 5.6.

Tableau 5.5 Qualité de regroupement après amélioration

		RS					GDE				
Pbs	**size**	e_{min}	e_{moy}	e_{max}	**écart type**	α_{best}	e_{min}	e_{moy}	e_{max}	**écart type**	α_{best}
Pb1[67]	10x20	1	1	**1**	0	1,388	1	1	**1**	0	1,388
Pb2[67]	16x30	0,6185	0,6185	**0,6185**	0	1,2328	0,566	0,59	**0,6185**	0,0106	1,2931
Pb3_dataset1[43]	24x40	1	1	**1**	0	1,5115	1	1	**1**	0	1,5115
Pb4_dataset2[43]	24x40	0,8391	0,8391	**0,8391**	0	1,3615	0,8391	0,8391	**0,8391**	0	1,4231
Pb5_dataset3[43]	24x40	0,6946	0,6946	**0,6946**	0	1,145	0,6946	0,6946	**0,6946**	0	1,1908
Pb6_dataset4[43]	24x40	0,6946	0,6946	**0,6946**	0	1,1374	0,6946	0,6946	**0,6946**	0	1,1985
Pb7_dataset5[43]	24x40	0,1915	0,303	0,3486	0,046	0,7863	0,2517	0,3134	**0,3536**	0,0292	0,8244

Tableau 5.6 Tableau de comparaison avec les autres travaux

		algorithme ZODIAC [43]			**AG[46]**				**RS**				**GDE**			
Pbs	**size**	**e1**	**e2**	**e**	**e1**	**e2**	**e**	**α**	**e1**	**e2**	**e**	**α**	**e1**	**e2**	**e**	**α**
Pb1[67]	10x20	-	-	-	1	0	**1**	1,388	1	0	**1**	1,388	1	0	**1**	1,388
Pb2[67]	16x30	-	-	-	0,7823	0,1638	**0,6185**	1,25	0,7823	0,1638	**0,6185**	1,2328	0,7823	0,1638	**0,6185**	1,2931
Pb3_dataset1[43]	24x40	1	0	1	1	0	**1**	1,511	1	0	**1**	1,511	1	0	**1**	1,511
Pb4_dataset2[43]	24x40	0,916(1)	0,0769(1)	**0,839(1)**	0,916(1)	0,0769(1)	**0,839(1)**	1,225	0,916	0,077	**0,839**	1,3615	0,916	0,077	**0,839**	1,4154
Pb5_dataset3[43]	24x40	0,841(2)	0,153(2)	0,688(2)	0,841(2)	0,153(2)	0,688(2)	1,162	0,847	0,153	**0,695**	1,145	0,847	0,153	**0,695**	1,1908
Pb6_dataset4[43]	24x40	0,847	0,153	**0,694**	-	-	-	-	0,8473	0,1527	**0,694**	1,1374	0,8473	0,1527	**0,694**	1,1985
Pb7_dataset5[43]	24x40	0,5466	0,374	0,1726	-	-	-	-	0,6692	0,3206	0,3486	0,7863	0,6742	0,3206	**0,3536**	0,8321

(1) et (2) : il y avait une erreur de calcul dans ces valeurs, une explication est présentée dans l'annexe.

Nous calculons la qualité des regroupements donnés par le RS et l'algorithme de GDE pour chaque problème traité et nous comparons les résultats trouvés avec celles qui ont été données par ZODIAC et le GA (tableau 5.6). Deux solutions ont été modifiées parmi les sept problèmes traités. La qualité du regroupement trouvée pour le problème cinq (pb5) est améliorée où *e* augmente de 0.688 à 0.6946. La deuxième solution modifiée est celle du septième (pb7) problème où nous avons trouvé une qualité de regroupement supérieure à celle qu'a donnée Chandrasekharan et Rajagopalan [43] où *e* augmente de 0,1726 à 0,3536.

La figure 5.1 est une représentation des solutions acceptées par l'algorithme GDE durant le processus de recherche pour le problème (Pb2).

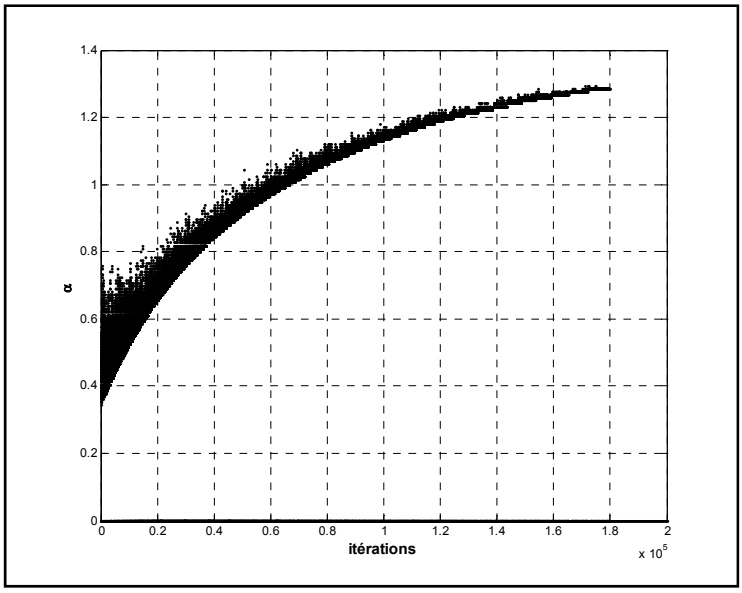


Figure 5.1 Ensemble des solutions acceptées par le GDE.

5.3 Ordonnancement

5.3.1 Évaluation de l'algorithme de GDE

Pour évaluer la performance de l'algorithme du GDE, nous avons appliqué l'algorithme proposé sur des problèmes d'ordonnancement de type *flowshop*. Vingt (20) problèmes de 4 dimensions différentes (100 opérations/5 machines, 100 opérations /20 machines, 200 opérations /20 machines et 500 opérations /20 machines) ont été étudiés en utilisant l'algorithme du GDE ; ces problèmes étaient initialement proposés par Taillard [47]. L'algorithme du GDE est codé avec Matlab et exécuté avec un ordinateur Dual Core avec une vitesse du processeur de 1,86 GHz et 3 GB de mémoire vive (RAM). La valeur du pas ΔB a été fixée à 0.05 avec le nombre d'itérations à 5.10^5.

Afin de mieux évaluer la performance de l'algorithme du GDE, nous avons comparé les résultats obtenus avec deux méthodes existantes dans la littérature. Nous avons comparé nos résultats avec celles qui ont été obtenus par les AG proposés par Reeves [48] et avec le RS-hybride proposé par Nearchou [49]. Nous avons présenté les résultats obtenus dans le tableau 5.7, ainsi que les variations des résultats (*%offset*) obtenus par les AG, RS-hybride et l'algorithme du GDE par rapport aux meilleurs résultats présentés par Taillard [47]; les paramètres *%offset* et sa moyenne sont calculés comme suit :

$$\%offset = ((Makespan - UB) / UB) \times 100$$

$$moyenne = (1 / n\_Pb) \times soumme(\%offset)$$

Avec *n_Pb* est le nombre de problèmes de même dimension.

En examinant les résultats obtenus (tableau 5.7), nous trouvons que l'algorithme du GDE a donné des résultats supérieurs aux AG et au SA-hybride pour 12 problèmes parmi les 20 problèmes, surtout pour des problèmes de grande taille. De plus, le GDE a montré une bonne performance par rapport aux AG. Il a donné un makespan inferieur à celui donné par le AG pour 14 problèmes, comme il est représenté graphiquement dans la figure 5.2.

D'après ces résultats, nous pouvons déduire que le GDE est capable de résoudre des problèmes d'ordonnancement dans un environnement *flowshop*. Cependant, il pourrait être utilisé pour des problèmes d'ordonnancement cellulaire afin de déterminer la bonne séquence de pièces qui minimise le makespan à l'intérieur de chaque cellule.

Tableau 5.7 Résultats obtenus pour 20 problèmes pris de la littérature [47]

	AG[48]	**RS-hybride[49]**	**GDE**
Problèmes	**Makespan**	**Makespan**	**Makespan**
100x5	5523	5502	**5495**
	5283	**5274**	5290
	5213	**5185**	5193
	5042	5029	**5023**
	5297	5257	**5253**
100x20	6450	6434	**6396**
	6441	6397	**6333**
	6488	**6451**	6496
	6422	6425	6440
	6519	**6497**	6524
200x20	11529	11483	**11437**
	11708	**11600**	11601
	11754	11721	**11536**
	11696	11597	**11530**
	11617	11576	**11492**
500x20	27338	26814	**26537**
	26843	26891	27028
	27334	26994	**26819**
	26973	26969	**26861**
	26298	26424	26792

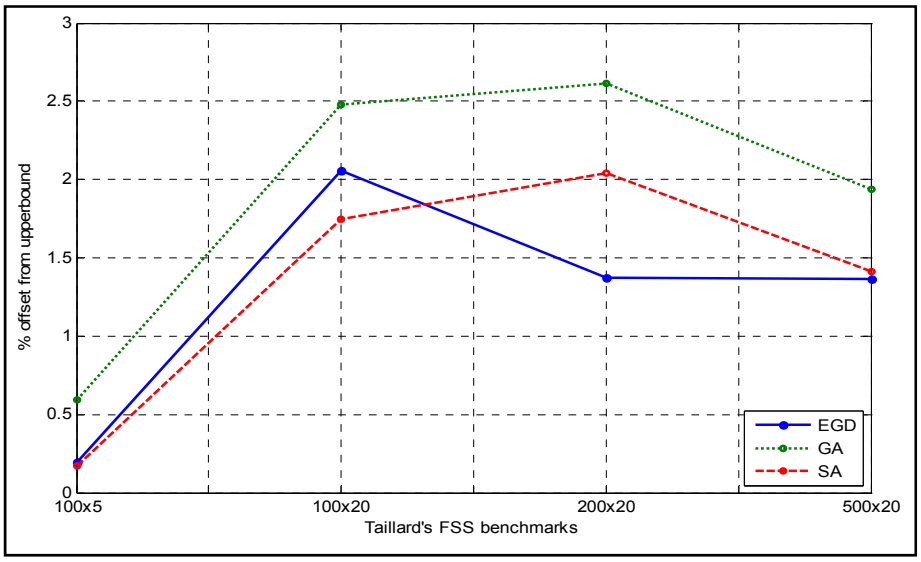


Figure 5.2 La variation des résultats obtenus par les 3 méthodes.

5.3.2 Ordonnancement cellulaire

Dans cette section, nous allons appliquer l'approche proposée sur un exemple illustratif pour faire l'ordonnancement cellulaire en tenant compte des éléments exceptionnels et par la suite nous appliquons la méthode d'amélioration proposée.

5.3.2.1 Exemple illustratif

Nous avons appliqué l'algorithme du GDE sur un problème d'ordonnancement cellulaire initialement traité par Solimanpur et al. [38]. L'objectif est de déterminer la meilleure séquence de pièces dans chaque cellule en tenant compte des éléments exceptionnels qui minimisent le makespan. Les résultats ont été comparés avec ceux qui ont été obtenus par la méthode *SVS-algorithm* proposée par Solimanpur et al. [38]. Cet exemple a été traité pour évaluer la performance de l'approche proposée dans le chapitre 4. Le problème traité est un exemple formé de 10 pièces/8 machines. Les machines sont déjà regroupées pour former les cellules de fabrication. Les données prises en considération sont : le temps d'exécution des opérations de chaque pièce, présenté au tableau 5.8 et le temps d'installation des machines

dans chacune des cellules présenté au tableau 5.9. Dans cet exemple, il y a 4 éléments exceptionnels : il y a 3 pièces qui nécessitent une opération ou plus dans d'autres cellules que celle à laquelle elles appartiennent. La pièce 1 est fabriquée dans la cellule 2 mais elle doit être envoyée à la cellule 1 pour réaliser une opération sur la machine E d'une durée de 5 unités de temps. De même pour la pièce 7, fabriquée dans la cellule 3, mais elle lui manque une opération sur la machine C de la cellule 2. La troisième est la pièce 5 qui doit être fabriquée dans la cellule 1, mais elle passe par les deux autres cellules pour réaliser une opération sur la machine C et une autre sur la machine F.

Tableau 5.8 Temps d'exécution des opérations sur chaque machine

		Pièces									
Machines		5	6	9	10	1	2	4	3	7	8
cellule 1	A	8	5	3	7						
	D	12	15		3						
	E	3	6	9		5					
cellule 2	C	11				4	6	8		8	
	G					10	12	4			
cellule 3	B								2	11	3
	F	4							14	8	6
	H								10	5	

Tableau 5.9 Temps d'installation des machines dans chaque cellule

	Machines							
Cellules	A	B	C	D	E	F	G	H
1	3		2	4	4	1		
2			2		3		6	
3		3	3			7		2

Solimanpur et al.[38] ont proposé une méthode pour résoudre des problèmes d'ordonnancement cellulaire avec des éléments exceptionnels et ils ont appelé cette approche *SVS-algorithm*. La solution obtenue par le *SVS-algorithm* est représentée par le diagramme de Gantt (figure 5.3). Cette solution propose une séquence de pièces pour chaque cellule qui a donné un makespan égal à 78. Dans les diagrammes de Gantt des figures 5.3, 5.4 et 5.7, la lettre *s* représente le temps d'installation.

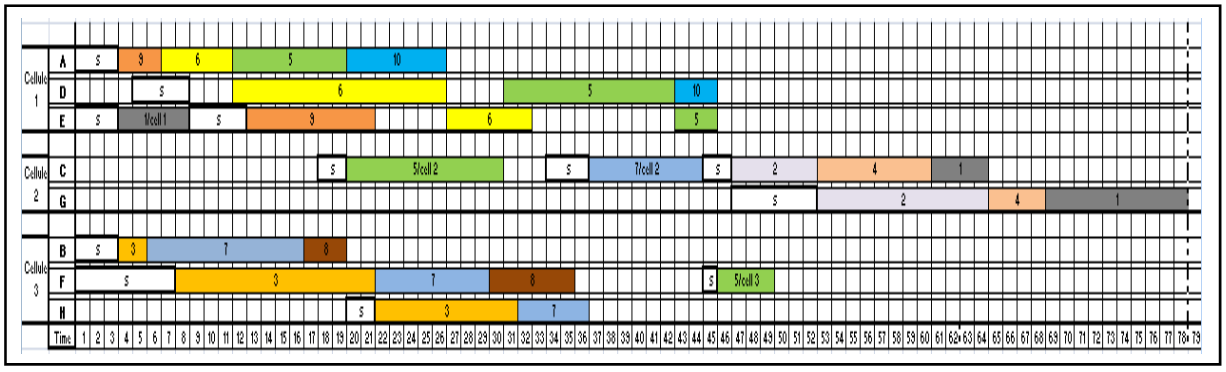


Figure 5.3 Solution proposée par Solimanpur et al.[38].

Le tableau 5.10 représente les délais d'achèvement de chaque opération, y compris des éléments exceptionnels en tenant compte des contraintes appliquées sur les pièces et les machines. Ce tableau représente les délais d'achèvement sans amélioration (avec un temps de repos important pour la plupart des machines) de chaque opération sur les machines associées. Cette première solution que nous nous proposons a donné un makespan qui est égal à 69, moins que celui trouvé par *SVS-algorithm* malgré que cette solution puisse être améliorée.

Le pourcentage de réduction (*E* %) du makespan par rapport à la méthode *SVS-algorithm*, sans amélioration ; ce pourcentage de réduction est 11,53 % et il est calculé comme suit :
E%=((le makespan obtenu par la méthode *SVS-algorithm*)-(le makespan obtenu par la méthode proposée))/ le makespan obtenu par la méthode *SVS-algorithm*.

Tableau 5.10 Délais d'achèvement de chaque opération

		Pièces									
Machines		6	9	5	10	1	4	2	3	7	8
cellule 1	A	19	22	30	37						
	D	34	34	46	49						
	E	40	49	52	52	8					
cellule 2	C			65		17	25	31		11	
	G					27	31	43			
cellule 3	B								16	27	30
	F			69					32	40	46
	H								42	47	47

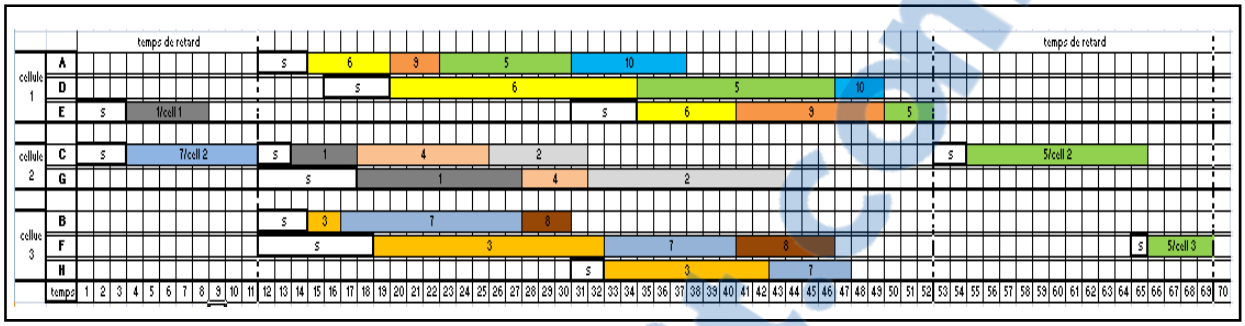


Figure 5.4 Notre première solution proposée.

L'approche proposée ne permet pas seulement de diminuer le makespan, mais permet aussi de minimiser le temps de déplacement des pièces entre les cellules par la minimisation du nombre de transferts intercellulaires. Par la méthode *SVS-algorithm*, des pièces peuvent être envoyées plus qu'une fois vers la même cellule, ce qui provoque un coût de manutention supplémentaire ainsi qu'une perte de temps. Dans cet exemple, la pièce 5 est fabriquée dans les trois cellules. Au début, la machine A qui appartient à la cellule 1 réalise une opération sur la pièce 5, ensuite la pièce 5 est transférée vers la cellule 2 pour réaliser une opération sur la machine C, après, elle est ré-envoyée de nouveau vers la cellule 1 pour compléter des opérations sur les machines D et E. Enfin, elle est transférée vers la cellule 3 pour réaliser la dernière opération sur la machine F. En fin de compte, la pièce 5 a fait 3 mouvements intercellulaires afin d'être fabriquée et qui sont représentés par la figure 5.5.

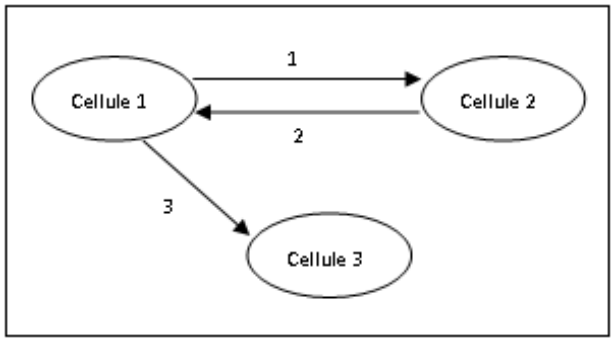


Figure 5.5 Transferts intercellulaires de la pièce 5 selon la méthode SVS-algorithme.

Avec la méthode que nous avons proposée pour résoudre des problèmes d'ordonnancement cellulaire, nous pouvons diminuer le nombre de transfert intercellulaire pour la pièce 5 à deux au lieu de trois avec la méthode *SVS-algorithm*. La figure 5.6 est une représentation des transferts intercellulaires de la pièce 5 suivant notre méthode. Tout d'abord, la pièce 5 est fabriquée dans la cellule 1 mais il reste à réaliser deux autres opérations ; alors, elle est transférée vers la cellule 2 pour réaliser une opération sur la machine C ; ensuite, elle est transférée vers la cellule 3 pour terminer la fabrication de la pièce par la réalisation d'une dernière opération sur la machine F.

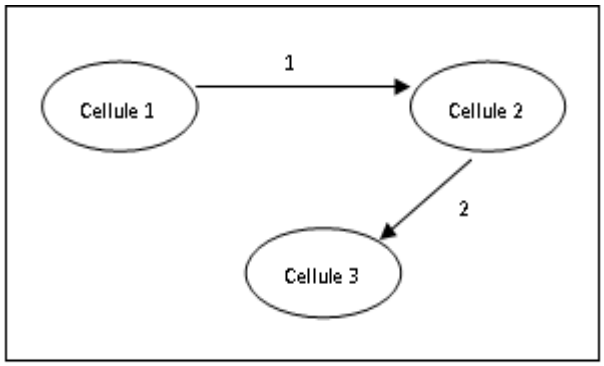


Figure 5.6 Transferts intercellulaires de la pièce 5 selon notre méthode

5.3.2.2 Solution améliorée

Pour avoir une meilleure solution plus performante, il faut diminuer le temps mort de chaque machine. Par conséquent, nous avons fait une modification de la première solution proposée (figure 5.4). Cette amélioration conserve toujours la même séquence de pièces sur chacune des machines. Dès qu'une machine termine de réaliser ses tâches sur les pièces appartenant à l'ensemble *P,* elle commence la réalisation des pièces affectées à la cellule à laquelle elle appartient en tenant compte des contraintes de temps de réalisation de chaque opération sur chaque machine. En observant le diagramme de Gantt de notre solution (figure 5.4), nous remarquons que les machines A, D, G, B, F et H sont en état de repos en attendant que les machines E et C finissent leurs tâches sur les pièces 1 et 7, respectivement. La machine A est

en repos pendant 11 unités de temps, alors nous pouvons réduire ce temps mort à 0 en commençant le traitement de la pièce 6 à *temps* = 0. De ce fait, la machine A termine toutes ses tâches à 26 au lieu de 37 et la machine D aussi terminera ses tâches à 38 au lieu de 49. Pour la machine E, nous avons une contrainte que nous devons respecter ; elle doit compléter sa tâche sur la pièce 1, donc elle sera libérée après 8 unités de temps. Cependant, elle sera mise au repos jusqu'à ce que la machine D finisse son opération sur la pièce 6 à *temps* = 23, donc la machine E finit toutes ses opérations à 41 au lieu de 52. Pour la cellule 2, la machine C ne sera libre pour la première fois qu'à *temps* = 31 lorsqu'elle finit ses opérations sur les pièces 7, 1, 4 et 2. Ensuite, elle sera mise en repos jusqu'à l'arrivée de la pièce 5 à *temps* = 41. La préparation de la machine C pour recevoir la pièce 5 peut être réalisée à *temps* = 39 avant l'arrivée de la pièce afin de minimiser le temps total de 2 unités car la machine 5 est au repos durant ce temps. La pièce 5 sera finie à 52 au lieu de 65. De même pour la cellule 3, toutes ses machines sont au repos durant tout le temps de retard qui est égal à 11 et qui peut être réduit tout en respectant les contraintes. La fabrication de la pièce 3 peut être lancée à *temps*=0, mais la machine B sera mise au repos jusqu'à l'achèvement de son opération sur la pièce 3 (à *temps* = 5) pour attendre l'arrivé de la pièce 7 à *temps* = 11. De même, les machines F et H doivent respecter cette contrainte imposée sur la pièce 7. De plus, la machine F sera au repos juste après avoir terminé la pièce 8 pour attendre la réalisation de la pièce 5 sur la machine C. Avec cette amélioration, nous avons diminué le makespan pour la cellule 1 à 41 au lieu de 52, pour la cellule 2 à 52 au lieu de 65, et pour la cellule 3 à 56 au lieu de 69. Par conséquent, le makespan total est de 56. La nouvelle solution optimale est présentée à la figure 5.7.

Le pourcentage de réduction (*E* %) du makespan par rapport à la méthode *SVS-algorithm*, après amélioration, est égale à 28,2 %.

Le tableau 5.11 représente les délais d'achèvement après amélioration (réduction du temps de repos des machines) de chaque opération sur les machines associées.

Tableau 5.11 Délais d'achèvement des opérations après amélioration

		Pièces									
Machines		6	9	5	10	1	4	2	3	7	8
cellule 1	A	8	11	19	26						
	D	23	23	35	38						
	E	29	38	41	41	8					
cellule 2	C			52		17	25	31		11	
	G					27	31	43			
cellule 3	B								5	22	25
	F			56					21	30	36
	H								31	36	36

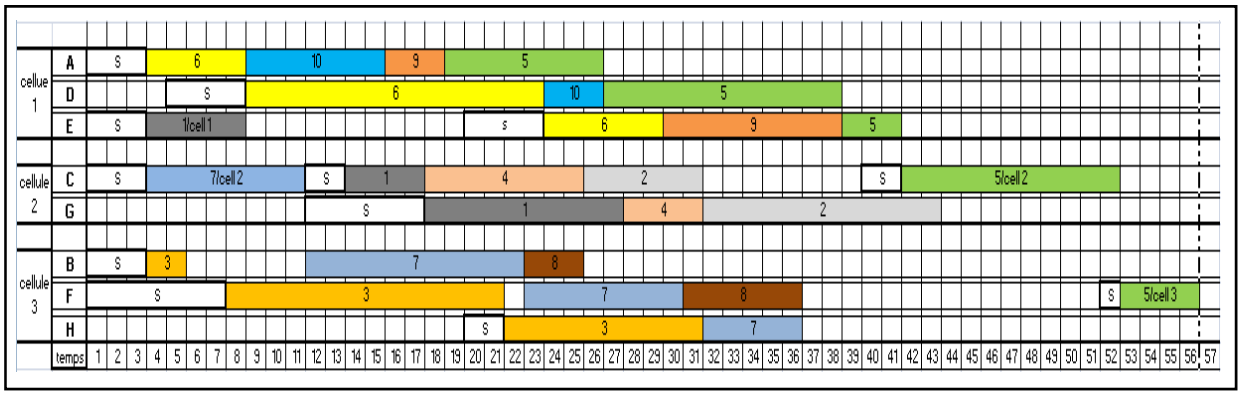


Figure 5.7 Solution améliorée.

Pour mieux tester la méthode proposée, nous avons sélectionné 13 problèmes de la littérature de petite, moyenne et grande taille. L'approche est codée avec Matlab et exécutés avec un ordinateur Dual Core avec une vitesse du processeur de 1,86 GHz et 3 GB de mémoire vive (RAM). La valeur du pas ΔB a été fixée à 10^{-3} avec un nombre d'itérations égal à 2.10^4. Les donnés des problèmes ont été générés d'une façon aléatoire suivant une distribution uniforme entre 0 et 100 et chaque problème a été exécuté 100 fois. En premier lieu, nous proposons des résultats sans application de l'heuristique d'amélioration. Dans ces solutions, le temps de repos des machines est élevé, et ces solutions seront améliorées par la suite. Le but visé est la

minimisation du makespan comme premier objectif et la minimisation du makespan et du *flowtime* comme deuxième objectif dans un contexte d'un problème de multi-objectif.

Pour le multi-objectif, nous avons donné un poids à chaque objectif. Pour la minimisation du makspan, le poids est de 0.9 et pour le *flowtime* le poids est de 0.1. La fonction « objectif » à minimiser α devient alors:

$$\alpha = 0.9*M_{sa}+0.1*F_{sa}$$

La moyenne du makespan, du *flowtime* et du temps de retard sont présenté dans le tableau 5.12. Malgré que les solutions trouvées soient initiales et que les résultats obtenus puissent être améliorées, nous avons trouvé de bons résultats et l'approche proposée a donnée de meilleurs résultats que le *SVS-algorithm* pour 5 problèmes (tableau 5.13).

Tableau 5.12 Résultats obtenus sans procédure d'amélioration

Problèmes	dimension			mono-objectif			multi-objectif		
	m	n	Nb de cellules	Temps de retard	*Flowtime*	Makespan moyen	Temps de retard	*Flowtime*	Makespan moyen
1. Kumar et Vannelli [50]	30	41	2	112.84	20502	731.77	112.84	14471	795.29
2. chandrasekharan et Rajagopalan (fig. 1) [43]	24	40	7	0	12786	555.21	0	11035	557.02
3. chandrasekharan et Rajagopalan (fig. 2) [43]	24	40	7	166.59	12499	703.99	166.59	11650	711.9
4. Carrie [51]	20	35	4	104.45	17033	791.42	104.45	13720	800.55
5. Harhalakis et al. [52]	20	20	5	271.13	5470	696.07	271.13	5305.9	712.15
6. seifoddini [53]	11	22	3	312.5	8035.1	891.04	312.5	7258.3	903.41
7. seifoddini [54]	5	18	2	255.6	6771	927.4	255.6	6005.8	931.43
8. Kusiak and Chow [55]	7	8	3	0	1073.2	205.08	0	978.58	206.24
9. King and Nakornchai [9]	5	7	2	0	1265.1	245.18	0	1148.7	247.37
10. waghodera and sahu (fig.2) [17]	5	7	2	64.26	1236.1	300.48	64.26	1214.6	303.79
11. waghodera and sahu (fig.3) [17]	5	7	2	67.69	1396.8	302.46	67.69	1247.7	312.99
12. waghodera and sahu (fig.4) [17]	5	7	2	145.86	1817	449.48	145.86	1470.7	460.75
13. waghodera and sahu (fig.5) [17]	5	7	2	149.94	1790.1	469.64	149.94	1530,2	471.51

Tableau 5.13 Comparaison avec *SVS-algorithm* sans procédure d'amélioration

	Dimension			Makespan moyen		
Problèmes	**m**	**n**	**Nb cells**	**Mono-objective**	**Multi-objective**	**SVS-algorithme**
1. Kumar et Vannelli [50]	30	41	2	731.77	795.29	**727,2**
2. chandrasekharan et Rajagopalan (fig. 1) [43]	24	40	7	555.21	557.02	**353,8**
3. chandrasekharan et Rajagopalan (fig. 2) [43]	24	40	7	**703.99**	**711.9**	1015,8
4. Carrie [51]	20	35	4	**791.42**	**800.55**	801,8
5. Harhalakis et al. [52]	20	20	5	**696.07**	712.15	711,5
6. seifoddini [53]	11	22	3	**891.04**	**903.41**	1019,2
7. seifoddini [54]	5	18	2	927.4	931.43	**897,1**
8. Kusiak and Chow [55]	7	8	3	205.08	206.24	**150**
9. King and Nakornchai [9]	5	7	2	245.18	247.37	**226,4**
10. waghodera and sahu (fig.2) [17]	5	7	2	**300.48**	**303.79**	408,3
11. waghodera and sahu (fig.3) [17]	5	7	2	**302.46**	**312.99**	372,3
12. waghodera and sahu (fig.4) [17]	5	7	2	449.48	460.75	**425,8**
13. waghodera and sahu (fig.5) [17]	5	7	2	469.64	471.51	**383,7**

Avec la procédure d'amélioration, nous avons atteint le minimimum de temps de repos des machines qui offre la possibilité de minimiser le makespan ainsi que le *flowtime*. Avec cette heuristique, le minimal makespan a été obtenu pour les 13 cas qui traitent les problèmes de mono et multi-objectif. Les résultats obtenus sont présentés dans le tableau 5.14 et tel que indiqué il y a 9 cas sur 13 les résultats sont meilleurs en comparant avec ceux obtenus par *VS-algorithm*,

Tableau 5.14 Résultats obtenus avec procédure d'amélioration

	Dimension			Mono-objectif	Multi-objectif	SVS-algorithme	E %	
Problems	**m**	**n**	**Nb. cells**	**Makespan moyen**	**Makespan moyen**	**Makespan moyen**	**Mono-objectif**	**Multi-objectif**
1. Kumar et Vannelli [50]	30	41	2	729,43	789,29	**727,2**	-0,3%	-8,53%
2. chandrasekharan et Rajagopalan (fig. 1) [43]	24	40	7	555,21	557,02	**353,8**	-56,93%	-57,4%
3. chandrasekharan et Rajagopalan (fig. 2) [43]	24	40	7	**516,81**	**520,77**	1015,8	49,12%	48,73%
4. Carrie [51]	20	35	4	**633**	**638**	801,8	20,05%	20,43%
5. Harhalakis et al. [52]	20	20	5	**514,1**	**520,56**	711,5	27,74%	26,84%
6. seifoddini [53]	11	22	3	**602,99**	**614,57**	1019,2	40,84%	39,7%
7. seifoddini [54]	5	18	2	**675,85**	**677,99**	897,1	24,66%	24,42%
8. Kusiak and Chow [55]	7	8	3	205,08	206,24	**150**	-36,72%	-37,5%
9. King and Nakornchai [9]	5	7	2	245,18	247,37	**226,4**	-8,3%	-9,26%
10. waghodera and sahu (fig.2) [17]	5	7	2	**273,67**	**282,33**	408,3	32,97%	30,85%
11. waghodera and sahu (fig.3) [17]	5	7	2	**238,98**	**277,8**	372,3	35,81%	25,38%
12. waghodera and sahu (fig.4) [17]	5	7	2	**375,44**	**378,99**	425,8	11,83%	10,99%
13. waghodera and sahu (fig.5) [17]	5	7	2	**369,58**	**373,21**	383,7	3,68%	2,73%

CONCLUSION

Dans le présent travail, une étude de regroupement et d'ordonnancement cellulaire a été réalisée dans les chapitres 3 et 4. Dans le chapitre 3, deux problèmes sont traités :

- La première étape vise à déterminer le meilleur routage de production afin de minimiser le coût opérationnel en tenant compte des contraintes liées au processus de fabrication.
- La deuxième étape est une application de deux algorithmes pour résoudre le problème de regroupement cellulaire, le premier algorithme est le RS, le deuxième est l'algorithme du GDE qui est appliqué pour la première fois à ce type de problème. Une comparaison entre les deux algorithmes et d'autres approches par la résolution de 7 problèmes prises de la littérature où l'algorithme de grand déluge étendu a montré une meilleure performance par rapport aux autres en ce qui a trait à la qualité de regroupement.

Le chapitre 4 présente une application de l'algorithme du GDE pour résoudre des problèmes d'ordonnancement. Deux étapes sont présentées dans ce chapitre :

- Une étape d'évaluation de l'algorithme est présentée en premier. La performance de cet algorithme est évaluée et testée par 20 problèmes de moyenne et grande tailles d'un environnement *flow shop* sélectionnés de la littérature.
- Une approche basée sur l'algorithme du GDE est présentée pour les problèmes d'ordonnancement des pièces à l'intérieur des cellules avec une proposition d'une méthode pour traiter les éléments exceptionnels.
- Une heuristique d'amélioration est développée afin de minimiser le temps mort des machines. L'approche développée est testée sur 13 problèmes pris de la littérature.

L'originalité de ce travail réside dans le développement d'une approche qui est en mesure de résoudre les problèmes de formation et d'ordonnancement cellulaire avec des éléments exceptionnels en utilisant un outil (GDE) qui n'a jamais été utilisé pour ce type de problèmes.

Le test de l'approche proposée en ce qui a trait à des problèmes de regroupement et d'ordonnancement présenté dans les chapitres 3 et 4, nous permet d'évaluer et de mettre en évidence les points forts et les points faibles de l'approche. Nous pouvons conclure que :

- L'approche proposée peut donner de bons résultats pour des problèmes de petite, moyenne et grande tailles.
- L'approche a montré une très bonne performance dans des problèmes de regroupement cellulaire et permet d'obtenir de bonnes qualités de regroupement.
- L'algorithme à base du GDE offre la possibilité de fournir des bons résultats pour des problèmes d'ordonnancement dans un environnement *flow shop,* même pour des problèmes de grande taille.
- L'approche proposée pour l'ordonnancement cellulaire permet de donner de très bons résultats.
- L'approche donne la possibilité de faire un ordonnancement cellulaire avec l'introduction des éléments exceptionnels tout en minimisant le temps mort des machines afin de profiter des équipements disponibles en excluant la sous-traitance et la duplication des machines.
- La méthode proposée permet de réduire le nombre de mouvement intercellulaire des éléments exceptionnels.
- Les résultats donnés par l'approche pour des problèmes de regroupement nécessitent parfois une amélioration mineure manuelle en augmentant la taille du problème;
- Le nombre d'itérations augmente proportionnellement à la taille du problème;
- La méthode ne traite pas de l'ordre de priorités dans les cellules.

- La performance de la méthode est en fonction du choix des valeurs du pas ΔB et du nombre d'itérations.

Des sujets intéressants pourraient favoriser la poursuite de ce travail :

- La méthode développée à la base du GDE pour des problèmes de regroupement cellulaire présentée dans le chapitre 3 nécessite un réglage délicat du taux de croissance/décroissance pour obtenir de bonnes qualités de regroupement. Une méthode de réglage de ce paramètre pourrait être ajoutée à l'approche.
- Le processus de recherche de la solution commence par une solution aléatoire qui sera améliorée par la suite. Généralement, la solution initiale a une influence sur la qualité de la solution obtenue, en particulier pour les problèmes de grande taille. Alors, le développement d'une heuristique de sélection de la solution initiale pourrait améliorer l'approche.
- Une étude pourrait aussi être ajoutée à l'approche proposée pour introduire l'ordre de priorité dans l'ordonnancement cellulaire.

Ce travail a produit la naissance de trois articles :

- Le premier article accepté est: «*Optimization of group scheduling using simulation with the meta-heuristic extended great deluge (EGD) approach*», *IEEE International Conference on Industrial Engineering and Engineering Management*, 7-10 December 2010, Macao.

- Le deuxiéme article accepté est *: «Optimization of Group Scheduling Problem Using the Hybrid Meta-heuristic Extended Great Deluge (EGD) Approach: A Case Study», IEMS International Conference on Industry, Engineering, and Management Systems,*28-30 mars 2011, Florida.

- Le troisième article soumis sous le titre: *Optimization of Manufacturing Cell Formation with Extended Great Deluge Approach* », *International Conference on Industrial Engineering and Systems Management, 25 – 27 mai, 2011, METZ-FRANCE*. Une réponse est attendue pour janvier 2011.

La combinaison des trois a fait l'objet d'un article de journal qui sera envoyé plus tard.

ANNEXE I

REGROUPEMENT CELLULAIRE

Exemples traités et résultats obtenus avec le GDE

- Résultat obtenu du problème 10 machines et 20 pièces

Tableau I.1 Matrice binaire regroupée, 10 machines et 20 pièces

		Pièces																		
		18	13	15	14	20	17	7	1	4	6	11	12	19	16	9	3	5	10	2	8
Machines	10										1	1	1	1	1	1					
	7										1	1	1	1	1	1					
	8										1	1	1	1	1	1					
	2																1	1	1	1	1
	3																1	1	1	1	1
	9	1	1	1	1	1	1														
	5	1	1	1	1	1	1														
	6							1	1	1											
	4							1	1	1											
	1							1	1	1											

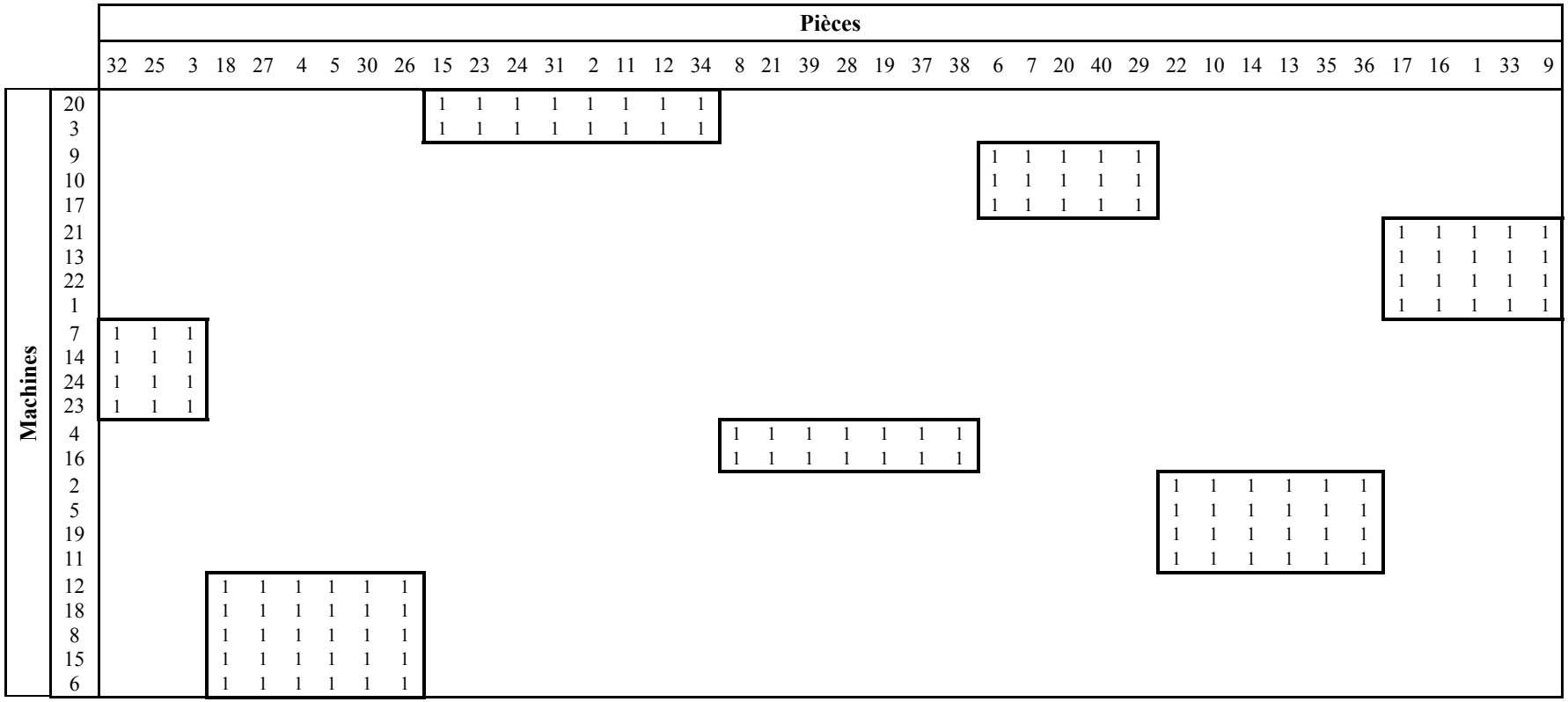
- Résultat obtenu du problème 16 machines et 30 pièces

Tableau I.2 Matrice binaire regroupée, 16 machines et 30 pièces

		Pièces																													
		17	11	21	14	15	8	6	24	26	3	13	20	10	16	1	28	19	5	25	23	27	29	30	7	12	2	18	4	22	9
Machines	11		1																					1	1	1	1	1	0	0	1
	8																1							0	1	1	1	1	1	1	1
	1																		1					1	1	1	0	1	1	1	1
	4																							1	1	0	1	1	1	1	1
	7																			1				1	1	1	1	1	1	1	0
	12												1					1						0	1	1	1	1	1	1	0
	2										0	0	1	1	1	1												1	1		
	13										1	1	1	1	0	0							1	1							
	3											1					1	0	0	1	1	1	1						1		
	6																0	0	0	1	1	1	1								
	15									1							0	0	1	1	1	1	1	1							
	9																1	1	1	1	1	1	0								
	14	1	1	1	1	1	1	1	0	0												1									
	10	0	1	1	1	1	1	1	1	1																		1			
	5	0	1	1	1	1	1	1	1	1	1																				
	16	0	0	0	1	1	1	1	1	1												1			1						

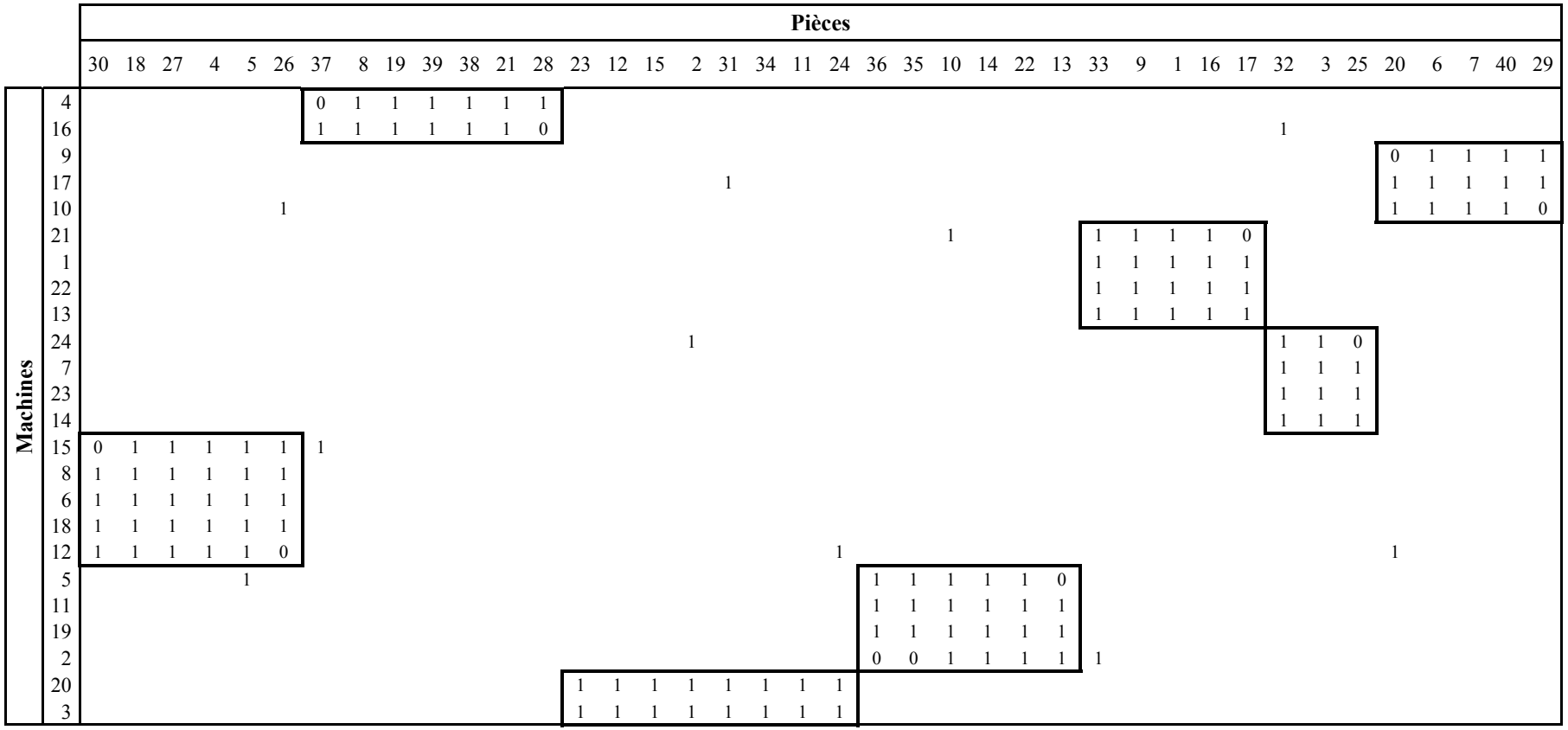
- Résultats du problème 24 machines/ 40 pièces : Pb3_dataset1

Tableau I.3 Matrice binaire regroupée, 24 machines et 40 pièces: pb3_dataset1



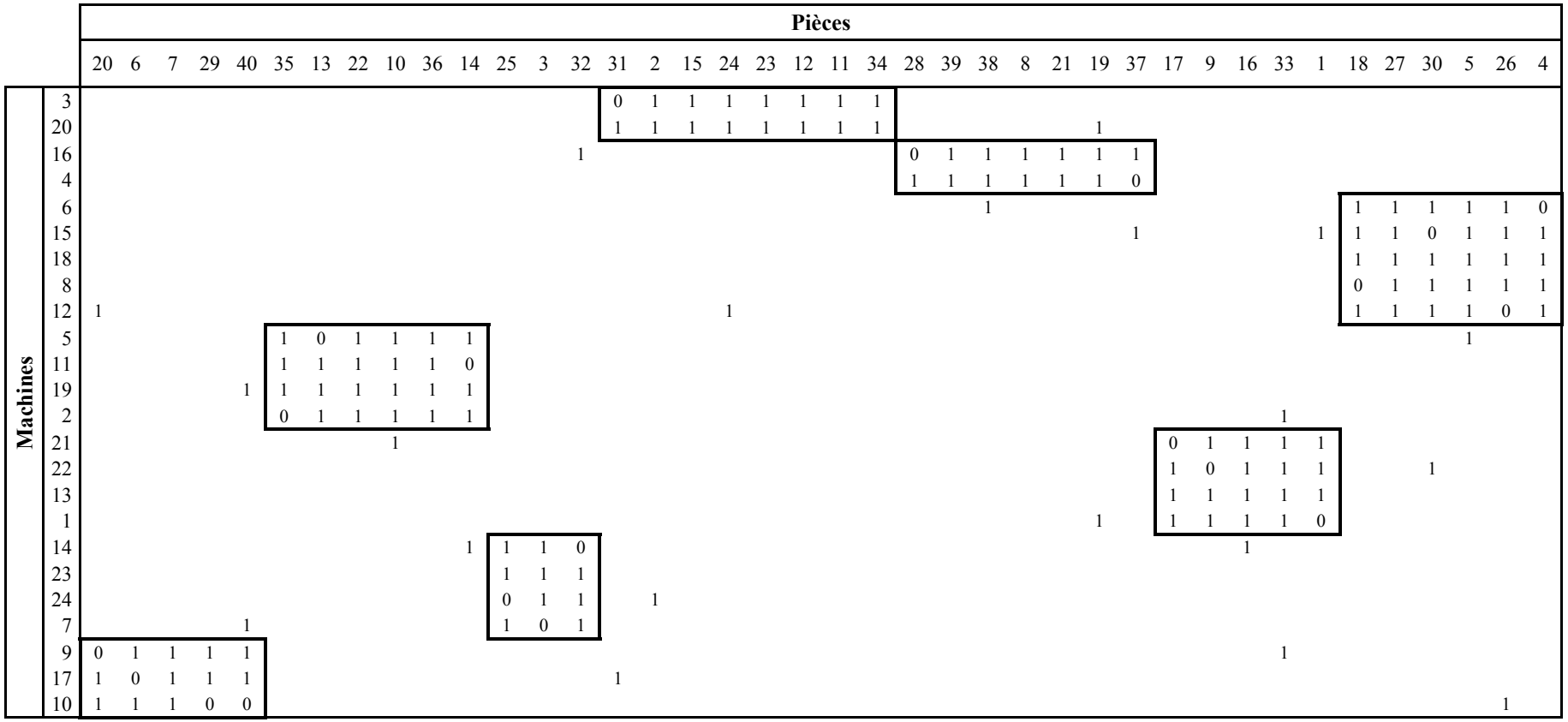
- Résultat du problème 24 machines/ 40 pièces : Pb4_dataset2

Tableau I.4 Matrice binaire regroupée, 24 machines et 40 pièces: pb3_dataset2



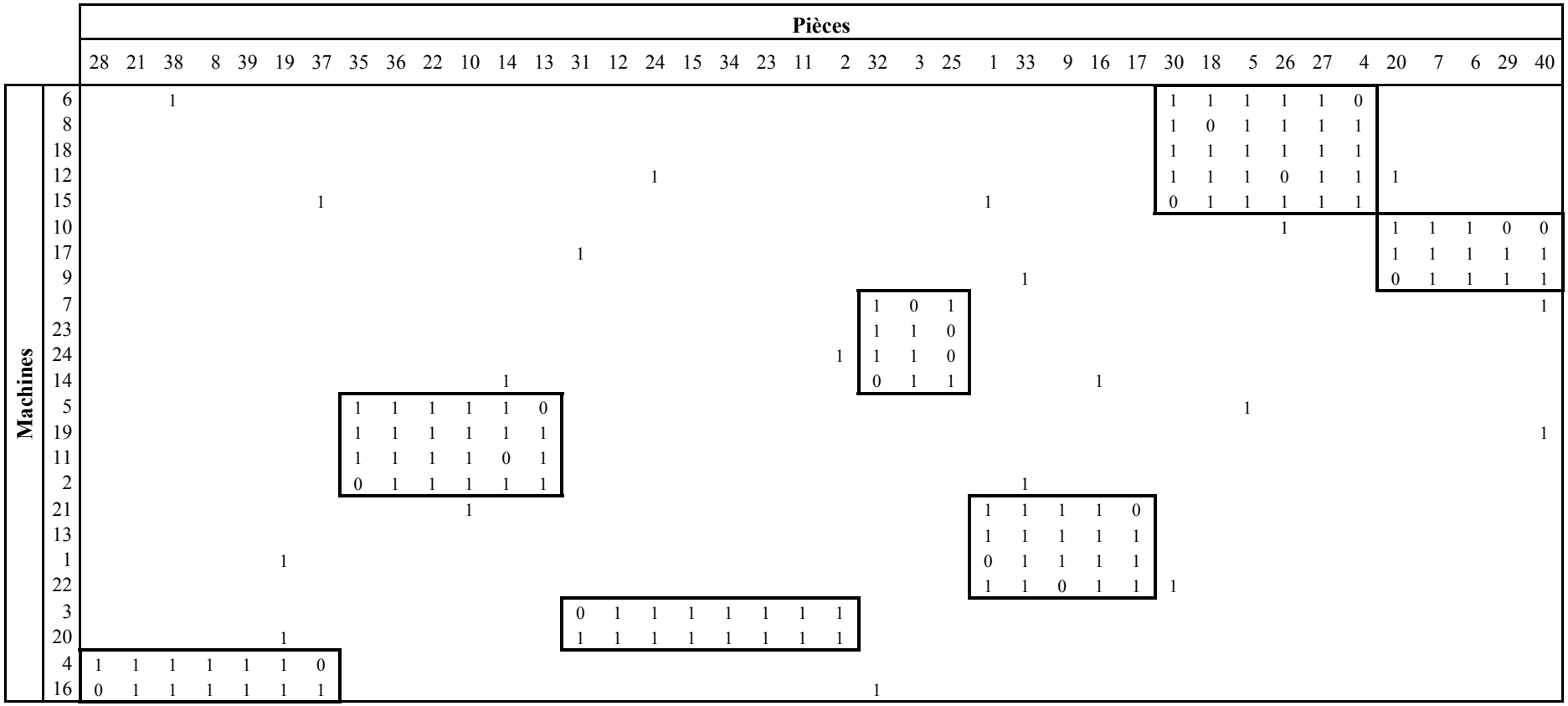
- Résultat obtenu du problème 24 machines/ 40 pièces : Pb5_dataset3

Tableau I.5 Matrice binaire regroupée, 24 machines et 40 pièces: pb3_dataset3



- Résultat du problème 24 machines/ 40 pièces : Pb6_dataset4

Tableau I.6 Matrice binaire regroupée, 24 machines et 40 pièces: pb3_dataset4



- Résultat du problème 24 machines/ 40 pièces : Pb7_dataset5

Tableau I.7 Matrice binaire regroupée, 24 machines et 40 pièces:pb3_dataset5

Machines	Pièces																																							
		20	6	39	40	7	29	24	34	2	23	15	12	31	9	33	17	1	16	25	3	32	13	36	35	10	22	14	5	4	27	18	30	26	19	8	28	11	37	21	38
	10	1	1	0	0	0	0	1																										1								
	17	1	0	0	0	1	1							1			1																									
	9	0	1	1	1	0	1									1																										
	16																					1		1	1									1	1	1	0	0	1	1	0	
	4																																		1	1	1	1	0	1	1	
	20							0	1	1	1	1	1	1						1															1				1			
	3							1	1	1	1	1	1	0																												
	11										1	1											1	0	1	1	1	0													
	5												1										0	1	1	1	1	1	1												
	2															1							1	1	0	0	1	1													
	19				1																		1	1	1	0	0	1			1										
	23																			0	1	1	1																		
	24									1										0	1	1																			
	7				1															1	0	1																			
	14																		1	1	1	0						1													
	1														1	1	1	0	0			1													1							
	21														1	1	0	1	1		1					1															
	13												1		0	0	1	1	1						1																
	22														0	0	1	1	1															1							
	6																									1			0	0	0	1	1	1							1
	18																												1	1	1	1	1	0				1			
	12	1						1																			1		1	1	1	1	0	0							
	15																	1										1	1	1	1	0	0	1					1		
	8			1					1																				1	1	0	0	1	1							

- **Erreur de calcul commise dans les problèmes 4 et 5**

(1) Dans le problème, il y une erreur dans les valeurs de e_1 et e_2 entre les résultats donnés par Mak et al. [46] et celles données par Chandrasekharan et al. [43] du problème 4 (dataset 2).

Les résultats donnés dans Mak et al. [46] sont les suivants :

$e_1 = 0.939$

$e_2 = 0.075$

$e = e_1 - e_2 =$ **0.864**

Or, on calcule la qualité de regroupement dans la matrice donnée par Chandrasekharan et al. [43] tableau I.8, on trouve :

$n_1 = 120$

$n_2 = 10$

$e_1 = 120/(15+12+30+24+14+16+20) = 0.916$

$e_2 = 1-(120/130) = 0,0769$

$e = e_1 - e_2 =$ **0,839**

(2) La matrice des regroupements cellulaires du problème 5 (dataset 3) donnée par Mak et al. [46] est présentée au tableau I.9. Dans ce problème, l'auteur a fait une erreur au niveau des opérations de la pièce 30. Dans cet exemple, la pièce 30 nécessite des opérations sur les machines 6, 8, 12, 15 et 18. Par contre, dans la matrice originale proposée par Chandrasekharan et al. [43] présenté au tableau I.10, cette pièce nécessite une opération sur la machine 22 et pas sur la machine 15. Ce changement a provoqué des fausses valeurs pour calculer la qualité du regroupement e, et plus de ca, la somme des valeurs non nul de la matrice incidente est mal calculé, il a utilisé une somme égale à 120 qui est en réalité (dans son exemple) égale à 121:

$n_1 = 112$

$n_2 = 18$

Ce qui donne :

$e_1 = 0.855$

e_2=0.138

$e = e_1 - e_2 =$ **0.717**

Or, dans la matrice donnée par Chandrasekharan et al. [43], tableau I.10, on trouve que n_2=20 et n_1 = 111, où :

e_1=0,841

e_2=0,153

$e = e_1 - e_2 =$ **0,688**

L'erreur se trouve au niveau de la pièce 30. Dans la matrice initiale donnée par Chandrasekharan et al. [43], les opérations de la pièce 30 sont réalisées sur les machines 6, 8, 12, 18 et 22. Or, dans la matrice donnée par Mak et al. [46] l'une de ces opérations est assignée à la machine 15 au lieu de la machine 22 ce qui donne une valeur de n_1 = 112. De plus, la valeur de n_2 donnée dans cet article est 18 alors que la valeur de n_2 de cette matrice est égale à 19.

Tableau I.8 Matrice proposée par Chandrasekharan et al. [43]

Machines	Pièces																																							
	1	9	16	17	33	2	11	12	15	23	24	31	34	8	19	21	28	37	38	39	10	13	14	22	35	36	4	5	18	26	27	30	3	25	32	6	7	20	29	40
1	1	1	1	1	1																																				
13	1	1	1	1	1																																				
21	1	1	1	0	1																**1**																			
22	1	1	1	1	1																																				
3						1	1	1	1	1	1	1	1																												
20						1	1	1	1	1	1	1	1																												
4														1	1	1	1	0	1	1																				
16														1	1	1	0	1	1	1															**1**					
2					**1**																1	1	1	1	0	0														
5																					1	0	1	1	1	1		**1**												
11																					1	1	1	1	1	1														
19																					1	1	1	1	1	1														
6																											1	1	1	1	1	1								
8																											1	1	1	1	1	1								
12											**1**																1	1	1	0	1	1						**1**		
15																		**1**									1	1	1	1	1	0								
18																											1	1	1	1	1	1								
7																																	1	1	1					
14																																	1	1	1					
23																																	1	1	1					
24						**1**																											1	0	1					
9																																				1	1	0	1	1
10																														**1**						1	1	1	0	1
17												**1**																								1	1	1	1	1

Tableau I.9 La matrice proposée par Mak et al. [46]

	Pièces																																							
Machines	6	7	29	20	40	31	12	34	15	23	11	2	24	17	33	9	16	1	10	13	36	22	35	14	26	5	4	30	27	18	3	25	32	37	19	38	8	21	39	28
11																			1	1	1	1	1	0																
19					1														1	1	1	1	1	1																
5																			1	0	1	1	1	1		1														
2															1				1	1	1	1	0	1																
12				1									1												0	1	1	1	1	1										
8																									1	1	1	1	1	0										
18																									1	1	1	1	1	1										
6																									1	1	0	1	1	1						1				
15																		1							1	1	1	1	1	1				1						
10	1	1	0	1	0																				1															
17	0	1	1	1	1	1																																		
9	1	1	1	0	1										1																									
3						0	1	1	1	1	1	1	1																											
20						1	1	1	1	1	1	1	1																						1					
4																																		0	1	1	1	1	1	1
16																																	1	1	1	1	1	1	1	0
21														0	1	1	1	1	1																					
1														1	1	1	1	0																	1					
13														1	1	1	1	1																						
22														1	1	0	1	1										0												
14																	1							1							1	1	0							
7					1																										0	1	1							
23																															1	1	1							
24											1																				1	0	1							

Tableau I.10 Matrice incidente originale proposée par Chandrasekharan et al. [43]

Machines \ Pièces	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1									1							1	1		1														1							
2										1			1	1								1											1			1				
3		1									1	1			1								1	1										1						
4								1											1		1							1									1	1	
5					1					1				1								1													1	1			
6					1													1								1	1			1						1		
7																									1							1							1
8				1	1																					1	1			1								
9						1	1																						1			1							1
10						1	1													1						1												
11										1			1									1													1	1			
12				1	1													1		1			1				1			1								
13	1								1							1	1															1							
14			1											1		1									1														
15	1			1	1													1								1	1			**0**						1		
16								1											1		1											1				1	1	1	
17							1													1									1		1								1
18				1	1													1								1	1			1								
19										1			1	1								1													1	1			1
20		1									1	1			1				1				1	1							1			1				
21	1								1	1						1																1						
22	1															1	1													**1**		1						
23			1																						1						1						
24		1	1																												1						

REFERENCES

[1] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis. New York: John Wiley & Sons, 1973.

[2] H. Lee and A. Garcia-Diaz, "A Network Flow Approach to Solve Clustering Problems in Group Technology", International Journal of Production research, vol. 31, pp. 603-612, 1993.

[3] V. Venugopal and T. Narendran, "A Genetic Algorithm Approach to the Machine-Component Grouping Problem with Multiple Objectives", Computers and Industrial Engineering, vol. 22, pp. 469-480, 1992.

[4] J. R. King, "Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm", *International Journal of Production Research*, vol. 18, pp. 213-232, 1980.

[5] Zolfaghari Saeed, "Design and Planning for Cellular Manufacturing: Application of Neural Networks and Advanced Search Techniques", *School of Graduate Studies and Research, University of Ottawa*, 1997.

[6] Kumar, K.R. and Vannelli, A., "Design of flexible manufacturing systems: capacity balancing and subcontracting strategies", *Proceedings of the 2nd ORSA/TIMS Conference on FMS*, Elsevier Science, Amsterdam, pp. 203-208, 1986.

[7] Rajamani D., Singh, N. and Aneja, Y.P., "Integrated design of cellular manufacturing systems in the presence of alternative process plans", *International Journal of Production Research*, Vol. 28 No. 8, pp. 1541-53, 1990.

[8] Co H.C., and Araar, A., "Configuring cellular manufacturing systems", *International Journal of Production Research*, Vol. 26 No. 9, pp. 1511-22, 1988.

[9] King, J. R., and Nakornchai, V., ''Machine-component group formation in group technology: review and extension'', *International Journal of Production Research,* Vol. 20, pp. 117-133, 1982.

[10] Timothy J. Greene and Randall P. Sadowski, "Cellular Manufacturing Control", *Journal of Manufacturing Systems*, Vol. 2 No. 2, pp. 137-145, 1983.

[11] T. Warren Liao, "Design of line-type cellular manufacturing systems for minimum operating and material-handling costs", *International Journal of Production Research,* Vol. 32 No. 2, pp. 387-397, 1994.

[12] Kusiak, A., "the generalized group technology concept", *International Journal of Production Research,* Vol. 25, pp. 561-569, 1987.

[13] Kao Y. and Moon Y. B., "A Unified Group Technology Implementation using the Back Propagation Learning Rule of Neural Networks", *Computers and Industrial Engineering*, Vol. 20, pp. 425-437, 1991.

[14] Kaparthi S. and Suresh N. C., "A neural Network System for Shape-Based Classification and Coding of Rotational Parts", *International Journal of Production Research*, Vol.29, pp. 1771 – 1784, 1991.

[15] Ümit Bilge, Murat Fırat, Erinç Albey, "A parametric fuzzy logic approach to dynamic part routing under full routing flexibility'', *Computers & Industrial Engineering*, Vol. 55, pp. 15–33, 2008.

[16] An-Yuan Chang, ''On the measurement of routing flexibility: A multiple attribute approach'', *International journal of production economics,* vol. 109, pp. 122-136, 2007.

[17] Kusiac A. and Chung Y., "GT/ART: Using Neural Networks to Form Machine Cells", *Manufacturing Review*, Vol. 4, pp. 293-301, 1991.

[18] Liao T. W. and Chen L. J., "An Evaluation of ART1 Neural Models for GT Part Family and Machine Cell Forming", *Journal of Manufacturing Systems*, Vol. 12, pp. 282-290, 1993.

[19] Venugopal V. and Narendran T., "Machine-Cell Formation Through Neural Network Models", *International Journal of Production Research*, Vol. 32, pp. 2105 – 2116, 1994.

[20] Dagli C. and Huggahalli R., "Configuring Cellular Manufacturing Systems", *International Journal of Production Research*, Vol. 26, pp. 1511 – 1522, 1991.

[21] Dagli C. and Huggahalli R., "Machine-Part Family Formation with the Adaptative Resonance Theory Paradigm", *International Journal of Production Research*, Vol. 33, pp. 893 – 913, 1995.

[22] Suresh N. C. and Kaparthi S., "Performance of fuzzy ART Neural Network for Group Technology", *International Journal of Production Research*, Vol. 32, pp. 1693 – 1713, 1994.

[23] Kamal S. and Burke L. I., "FACT: A New Neural Network-Based Clustering Algorithm for Group Technology", *International Journal of Production Research*, Vol. 34, pp. 919 – 946, 1996.

[24] Arizono I., Kato M., Yamamoto A., and Ohta H., "A New Stochastic Neural Network Model and its Application to Grouping and Tools in Flexible Manufacturing Systems", *International Journal of Production Research*, Vol. 33, pp. 1535 – 1548, 1995.

[25] Ateme-Nguema B. H. and Thiên-My Dao, "Quantized Hopfield networks and tabu search for manufacturing cell formation problems", *International Journal of Production Economics*, Vol. 121, pp. 88 – 98, 2009.

[26] Liu C. M. and Wu J. K., "Machine Cell Formation: using the Simulated Annealing Algorithm", *Journal of Computer Integrated Manufacturing*, Vol. 6, pp. 335-349, 1993.

[27] Boctor F. F., "The Minimum-Cost, Machine-Part Cell Formation Problem", *International Journal of Production Research*, Vol. 34, pp. 1045 – 1063, 1996.

[28] J. H. Holland, "Adaptation in Natural and Artificial Systems", *University of Michigan Press*, Ann Arbor, 1976.

[29] L. Davis (Ed.), "Genetic Algorithms and Simulated Annealing", *Pitman, London*, 1987.

[30] Venugopal V. and Narendran T., "A Genetic Algorithm Approach to the Machine-Component Grouping Problem with Multiple Objectives", *Computers and Industrial Engineering*, Vol. 22, pp. 469-480, 1992.

[31] Zolfaghari S. and Liang M., "Comprehensive machine cell/part family formation using genetic algorithms", *Journal of Manufacturing Technology Management*, Vol. 15 No. 6, pp. 433-444, 2004.

[32] Radharamanan R., "A Heuristic Algorithm for Group Scheduling", *Proceedings of the 8th Annual Conference on Computers and Industrial Engineering*, Vol. 11, pp. 204-208, 1986.

[33] Vakharia A. J. and Chang Y. L., "A simulated annealing approach to scheduling a manufacturing cell", *Naval Research Logistics*, Vol. 37, pp. 559-577, 1990.

[34] Sridhar J. and Rajendran C., ''Scheduling in a cellular manufacturing system: a simulated anneling approach'', *International Journal of Production Research*, vol. 31 No.12, pp. 2927-2945, 1993.

[35] Sridhar J. and Rajendran C., ''Scheduling in flowshop and cellular manufacturing systems with multiple objectives-A genetic algorithmic approach'', *Production Planning & Control*, Vol. 7 No. 4, pp. 374-382, 1996.

[36] Skorin-Kapov J. and Vakharia A. J., "Scheduling a flow-line manufacturing cell: a tabu search approach", *International Journal of Production Research*, Vol. 31, pp. 1721-1734, 1993.

[37] Schaller J., ''A comparison of heuristics for family and job scheduling in a flow-line manufacturing cell'', *International Journal of Production Research,* Vol. 38 No 2, pp. 287–308, 2000.

[38] Solimanpur M., Vrat P., Shankar R., " A heuristic to minimize makespan of cell scheduling problem'', *International Journal of Production Economics*, Vol. 88, pp. 231–241, 2004.

[39] T. Warren Liao, L.J. Chen, Z.H. Chen and E.R. Coates, "A comparison of two approaches for designing line type cellular manufacturing systems'', *Integrated Manufacturing Systems*, Vol. 7 No. 1, pp. 6–15, 1996.

[40] E. Burke, Y. Bykov, J. Newell and S. Petrovic, ''A time-predefined local search approach to exam timetabling problems'', *IIE Transactions*, Vol.36 No.6, pp. 509–528, 2004.

[41] S. Kirkpatrick, C. D. Gelatt and Jr. M. P. Vecchi, ''Optimization by Simulated Annealing'', *Science*, Vol. 220 No. 4598, pp. 671-680, 1983.

[42] Ateme-Nguema B. H., "Conception optimal des cellules de fabrication flexible basée sur l'approche par réseaux de neurones", *École de technologie supérieure, Université du Québec*, 2007.

[43] Chandrasekharan M. P. and Rajagopalan R., "GROUPABILITY: an analysis of the properties of binary data matrices for group technology", *International Journal of Production Research*, Vol. 27 No. 6, pp. 1035-1052, 1989.

[44] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A. and Teller E.Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, Vol. 21 No. 6, pp. 1087-1092, 1953.

[45] J. Miltenburg and W. Zhang, "A comparative evaluation of nine well-known algorithms for solving the cell formation problem in group technology", *Journal of Operations Management*, Vol. 10 No.1, pp. 44–69, 1991.

[46] K. L. Mak, Y. S. Wong and X. X. Wang, "An Adaptive Genetic Algorithm for Manufacturing Cell Formation", *The International Journal of Advanced Manufacturing Technology*, Vol. 16, pp. 491-497, 2000.

[47] Taillard, E., "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, Vol. 64, pp. 278-285, 1993.

[48] Reeves C.R., "A genetic algorithm for flowshop sequencing", *Computers and perations Research*, Vol. 22 No. 1, pp. 5-13, 1995.

[49] Nearchou A. C., "A novel metaheuristic approach for the flow shop scheduling problem", *Engineering Applications of Artificial Intelligence,* Vol. 17, pp. 289–300, 2004.

[50] Kumar K.R. and Vannelli A., "Strategic subcontracting for efficient disaggregated manufacturing", *International Journal of Production Research*, Vol. 25 No. 12, pp. 1715–1728, 1987.

[51] Carrie A.S., "Numerical taxonomy applied to GT and plant layout", International Journal of Production Research, Vol. 11 No. 4, pp. 399–416, 1973.

[52] Harhalakis G., Nagi R. and Proth J.M., "An efficient heuristic in manufacturing cell formation for group technology applications", International Journal of Production Research, Vol. 28 No. 1, pp. 185–198, 1990.

[53] Seifoddini, H., "Single linkage versus average linkage clustering in machine cells formation applications", *Computers & Industrial Engineering*, Vol. 16 No. 3, pp. 419–426, 1989a.

[54] Seifoddini, H., "A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications", *International Journal of Production Research*, Vol. 27 No. 7, pp. 1161–1165, 1989b.

[55] Kusiak A. and Chow W.S., "Efficient solving of the group technology problem", *Journal of Manuf acturing Systems*, Vol. 6 No. 2, pp. 117–124, 1987.

[56] Lundy S. and Mees A., "Convergence of an annealing algorithm", *Mathematical Programming,* Vol. 34, pp. 111-124, 1986.

[57] D. Connolly, "An improved annealing scheme for the QAP", *European Journal of Operational Research*, Vol. 46, pp. 93-100, 1990.

[58] Bäck T., Hoffmeister F. and Schwefel H-P., "A Survey of evolutionary Strategies", *Proc. of 4th Intl. Conference on Genetic Algorithms (ICGA'91), R. Belew, L. Booker (Eds.), Morgan Kaufmann*, pp. 2-9, 1991.

[59] Fatemeh A., Massimiliano D. P., Giuliano A. and Yann-Gaël G., "A Heuristic-based Approach to Identify Concepts in Execution Traces", *Proceedings of the 2nd*

International Symposium on Search Based Software Engineering (SSBSE '10), Benevento Italy, pp. 153-162, 7-9 September 2010.

[60] Rajkumar R. and Shahabudeen P., "An improved genetic algorithm for the flowshop scheduling problem", *International Journal of Production Research,* Vol. 47 No. 1, pp. 233-249, 2009.

[61] Dueck, G., "New optimization heuristics. The great deluge algorithm and the record-to-record travel", *Journal of Computational Physics*, Vol. 104, pp. 86–92, 1993.

[62] Glover F., "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research,* Vol. 13, pp. 533-549, 1986.

[63] Logendran R. and Ramakrishna P., "Manufacturing cell formation in the presence of lot splitting and multiple units of the same machine", *International Journal of Production Research,* Vol. 33, pp. 675-693, 1995.

[64] Süer G. A., Saiz M., Dagli C. and Gonzales W., "Manufacturing cell loading rules and algorithms for connected cells", *Planning, Design and Analysis of Cellular Manufacturing Systems,* Vol. 24, pp. 97-127, 1995.

[65] Süer G. A. and Saiz M.," Cell Loading in Cellular Manufacturing Systems", *Computers and Industrial Engineering,* Vol. 25 No. 1, pp 247-250, 1993.

[66] Baker K. R., "Introduction to Sequencing and Scheduling", John Wiley and Sons Inc., New York, 1974.

[67] Srinivasan G., Narendran T.T. and Mahadevan B., "An assignement model for part-families problem in group technology", *International Journal of Production Research*, Vol. 28 No. 1, pp. 145-152, 1990.