ROYAUME DU MAROC
Université Mohammed V - RABAT -
جامعة محمد الخامس – الرباط –
Faculté des sciences
كلية العلوم
CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

N° d'ordre 3077

THESE

En vue de l'obtention du : ***DOCTORAT***

Structure de Recherche : Laboratoire de Recherche en Informatique
Discipline : Sciences de l'ingénieur
Spécialité : Informatique

Présentée et soutenue le : 10 Mars 2018 par :

Bouchra KAROUM

APPROCHES INSPIRÉES DE LA NATURE APPLIQUÉES AU PROBLÈME DE FORMATION DE CELLULES MANUFACTURIER

JURY

Fouzia OMARY	*PES, Université Mohammed V Faculté des Sciences de Rabat*	*Président*
Belaïd AHIOD	*PH, Université Mohammed V Faculté des Sciences de Rabat*	*Rapporteur*
Youssef ELBENANI	*PH, Université Mohammed V Faculté des Sciences de Rabat*	*Directeur de thèse*
Mohamed OUZINEB	*PH, Institut National de Statistique et d'Economie Appliquée, Rabat*	*Rapporteur*
Mohamed EL HAZITI	*PH, Ecole Supérieure de Technologie, Salé*	*Examinateur*

Année Universitaire : 2017/2018

Dédicace

Je dédie ce travail :

A mes chers parents,

A ma sœur et mes frères

A tous ceux que j'aime et je respecte.

Avant-Propos

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire de Recherche en Informatique (LRI) de la Faculté des Sciences de Rabat (FSR), Université Mohammed V, sous la direction du Professeur **Youssef ELBENANI**.

Je commence par exprimer ma profonde gratitude à Monsieur **Youssef ELBENANI**, Professeur habilité à la Faculté des Sciences de Rabat, qui a bien voulu prendre en charge la direction de cette thèse. Grâce à sa pédagogie et ses précieux conseils, il a su me guider pour mener à bien mes travaux de recherche. Qu'il trouve ici l'expression de mes vifs remerciements et de mon profond respect.

Je tiens à remercier Madame **Fouzia OMARY**, Professeur de l'Enseignement Supérieur à la Faculté des Sciences de Rabat, pour l'intérêt qu'elle a accordé à ce travail et l'honneur qu'elle me fait de présider le jury de cette thèse.

Je tiens également à remercier Monsieur **Belaïd AHIOD**, Professeur habilité à la Faculté des Sciences de Rabat, pour avoir accepté de rapporter ce travail et de participer au jury.

Je suis très honoré que Monsieur **Mohamed El HAZITI**, Professeur habilité à l'École Supérieure de Technologie de Salé, ait accepté d'être Examinateur de ce travail et de participer au jury.

Je souhaite remercier profondément Monsieur **Mohamed OUZINEB**, Professeur habilité à l'Institut National de Statistique et d'Economie Appliquée de Rabat, pour avoir accepté de rapporter ce travail et de participer au jury.

Je remercie énormément tous les membres du jury pour leur disponibilité le jour de ma soutenance, pour le privilège qu'ils m'ont accordé par leur présence et pour l'intérêt qu'ils ont attribué à mon mémoire de thèse.

Résumé

Le travail de recherche présenté dans cette thèse porte sur la résolution du problème de formation de cellules, qui est considéré comme l'un des premiers problèmes rencontrés pour réaliser un système de production cellulaire. L'absence d'un algorithme déterministe capable de résoudre le problème de façon optimale et dans un temps de calcul raisonnable justifie souvent le recours aux algorithmes approximatifs. Dans ce contexte, nous proposons des approches basées sur deux métaheuristiques inspirées de la nature : l'algorithme de sélection clonale et l'algorithme de recherche du coucou.

Nous commençons par traiter la première variante du problème en considérant sa matrice d'incidence (produits/machines) et en fixant le nombre de cellules de production. Nous visons à maximiser l'efficacité du système en réduisant les mouvements intercellulaires des produits. Puis, nous étudions le cas où le nombre de cellules est variable comme une deuxième variante du problème de formation de cellules. Ensuite, nous abordons la troisième variante du problème qui prend en compte la possibilité d'avoir plusieurs routes de traitement pour chaque produit et d'intégrer différents facteurs de production ; ce qui rend le problème plus réaliste mais aussi plus complexe.

Nous avons développé plusieurs approches basées sur l'algorithme de sélection clonale et l'algorithme de recherche du coucou pour résoudre ces trois variantes. Chaque approche est adaptée différemment au problème pour satisfaire les contraintes de la variante étudiée. De plus, un bon réglage des paramètres est appliqué pour obtenir des meilleures solutions. Les performances des approches proposées sont évaluées sur un ensemble d'instances de référence et comparées à d'autres algorithmes collectés de la littérature. Les résultats obtenus indiquent l'efficacité de nos approches en termes de qualité des solutions et du temps de calcul. Il est à noter que nous sommes les premiers à proposer la résolution du problème et ses variantes avec ces deux approches inspirées de la nature.

Mots-clés : Problème de formation des cellules ; Métaheuristiques inspirées de la nature ; Recherche du coucou ; Sélection clonale ; Fiabilité des machines ; Routes alternatives ; Problème de formation de cellules généralisé.

Abstract

The research work presented in this thesis focuses on solving the cell formation problem, which is considered one of the first problems faced in designing a cellular manufacturing system. The lack of a deterministic algorithm able to solve the problem optimally and in a reasonable computational time often justifies the use of approximation algorithms. In this context, we propose approaches based on two metaheuristics inspired by nature : the clonal selection algorithm and the cuckoo search algorithm.

We start with the first variant of the problem by considering its incidence matrix (product / machine) and by setting the number of production cells. We aim to maximize the efficiency of the system by reducing the intercellular movements of products. Then, we study the case where the number of cells is variable as a second variant of the cell formation problem. Afterwards, we address the third variant of the problem that takes into account the possibility of having alternative process routings for each product and to integrate different production factors ; which makes the problem more realistic but also more complex.

We have developed several approaches based on the clonal selection algorithm and the cuckoo search algorithm to solve these three variants. Each approach is adapted differently to the problem to satisfy the constraints of the corresponding variant. In addition, a parameter setting is applied to obtain better solutions. The performances of the proposed approaches are evaluated on a set of benchmark instances and compared to other algorithms collected from the literature. The obtained results indicate the superiority of our approaches in terms of solution quality and computational time.

Keywords : Cell formation problem ; Nature-inspired metaheuristics ; Cuckoo search ; Clonal selection ; Machine reliability ; Alternative routes ; Generalized cell formation problem.

Table des matières

Liste des acronymes

ACO	: Ant Colony Optimization
AIRS	: Artificial Immune Recognition System
AIS	: Artificial Immune System
ART	: Adaptive Resonance Theory
BA	: Bees Algorithm
CFP	: Cell Formation Problem
ClonalSA-TOP	: Clonal Selection Algorithm-Team Orienteering Problem
CS	: Cuckoo Search
DGSA	: Discrete Gravitational Search Algorithm
DE	: Differential Evolution
DF	: Degree of freedom
EA	: Evolutionary Algorithm
FA	: Firefly Algorithm
GA	: Genetic Algorithm
GA_LNS	: Genetic Algorithm and Large Neighborhood Search
GCFP	: Generalized Cell Formation Problem
GDE	: Grouping Differential Evolution
GH	: Greedy Heuristic
GLCA	: Grouping League Championship Algorithm
GRAFICS	: GRouping using Assignment method For Initial Cluster Seeds
GRASP	: Greedy Randomized Adaptive Search Procedure
GSA	: Gravitational Search Algorithm
GT	: Group Technology
HClonalSA1	: the first Hybrid Clonal Selection Algorithm
HClonalSA2	: the second Hybrid Clonal Selection Algorithm
HClonalSA-VNC	: Hybrid Clonal Selection Algorithm-Variant Number of Cells
HClonalSA-GCFP	: Hybrid Clonal Selection Algorithm - Generalized Cell Formation
HCuckooSA1	: the first Hybrid Cuckoo Search Algorithm-Cell Formation Problem

HCuckooSA2	: the second Hybrid Cuckoo Search Algorithm-Cell Formation Problem
HCuckooSA-GCFP	: Hybrid Cuckoo Search Algorithm-Generalized Cell Formation Problem
HGBPSO	: Hybrid Grouping Particle Swarm Optimisation
HGDE	: Hybrid Grouping Differential Evolution
HGGA	: Hybrid Grouping Genetic Algorithm
LCA	: League Championship Algorithm
MA	: Memetic Algorithm
MS	: Mean Square
MTBF	: Mean Time Between Failure
NN	: Neural Network
OP	: Orienteering Problem
OSHN	: Ortho-Synapse Hopfield Network
PFC	: Problème de Formation de Cellules
PSO	: Particle Swarm Optimisation
SA	: Simulated Annealing
SOM	: Self-Organizing Map
SONN	: Self-Organizing Neural Network
SS	: Sum of Squares
STSP	: Selective Traveling Salesman Problem
SVM	: Support Vector Machine
TCNN	: Transiently Chaotic Neural Network
TOP	: Team Orienteering Problem
TS	: Tabu Search
VNS	: Variable Neighborhood Search
VRP	: Vehicle Routing Problem
WFA	: Water Flow-like Algorithm
ZODIAC	: Zero One Data : Ideal Seed Algorithm for Clustering

Liste des tableaux

Table des figures

Liste des algorithmes

Introduction générale

Contexte générale et problématiques

Les changements permanents des besoins du marché et le développement rapide des technologies de fabrication exercent une pression intensive sur les entreprises manufacturières. L'objectif principal de ces entreprises est de livrer rapidement les produits aux clients avec une diminution des coûts de production et une augmentation de la qualité des produits. Le système de production devrait donc être en mesure de répondre rapidement aux changements dans la conception des produits et/ou la demande des produits sans investissement majeur. Cela a incité plusieurs chercheurs à trouver de meilleures méthodes pour la conception et l'exploitation des systèmes de production. Le système de production cellulaire (CMS) est considéré comme l'un des systèmes adéquats qui répondent à ces exigences. Il est habituellement appliqué pour rendre les systèmes de fabrication plus efficaces et productifs.

Le système de production cellulaire est l'une des applications les plus importantes de la Technologie de Groupe qui tente de décomposer l'ensemble du système en sous-systèmes ou cellules de fabrication aussi indépendants que possible. Le but est de regrouper les produits ayant des besoins de traitement similaires et d'identifier l'ensemble des machines nécessaires au traitement de ces produits. Cela permet de simplifier les flux, assurer la flexibilité dans la fabrication d'une grande variété de produits, minimiser le temps de traitement, améliorer la qualité des produits et augmenter le taux d'utilisation des équipements, etc.

La production cellulaire a été adoptée dans différents types d'industries et d'applications telles que la fabrication des cadres en bois (Dinis-Carvalho *et al.*, 2014), le forgeage des métaux (Hung et Maleki, 2014), l'assemblage des composants de munitions pour les applications de défense (Pattanaik et Sharma, 2009), la fabrication de tracteurs (Shayan et Sobhanallahi, 2002). Ce type de production peut être efficace dans les entreprises qui produisent une variété de produits.

Le premier problème rencontré pour concevoir un système de production cellulaire est le problème de formation de cellules (Cell Formation Problem : CFP). Ce problème est classé parmi les problèmes d'optimisation NP-difficiles (Dimopoulos et Zalzala, 2000). Le traitement de ce type de problèmes se fait à l'aide d'un certain nombre de méthodes qui peuvent être classifiées en considérant deux caractéristiques : la qualité de la solution et le temps de calcul. Dans ce contexte, les

métaheuristiques ont montré leurs capacités à résoudre une grande variété de problèmes d'optimisation souvent issus des domaines de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle.

Les métaheuristiques sont des algorithmes stochastiques de recherche vers un optimum global. Elles se caractérisent par une stratégie de recherche qui combine l'exploration de l'espace de recherche avec une exploitation intensifiée des régions prometteuses. Dans la plupart des cas, les métaheuristiques sont inspirées de la nature, imitant des phénomènes physiques (cas du recuit simulé), biologiques (cas des algorithmes génétiques ou du système immunitaire) ou éthologiques (cas de l'optimisation par essaims particulaires)...

Dans cette thèse, nous développons plusieurs approches basées sur deux métaheuristiques inspirées de la nature pour répondre aux différentes variantes du problème : l'algorithme de sélection clonale et l'algorithme de recherche du coucou. À notre meilleure connaissance, nous sommes les premiers à proposer la résolution du problème et ses variantes avec ces deux algorithmes.

Contributions

L'idée de résoudre le problème de formation de cellules avec certaines métaheuristiques, inspirées de la nature et non encore appliquées à ce problème, est le motif de nos contributions. Durant nos recherches, nous avons traité trois variantes du problème et nous avons effectué, pour chaque approche proposée, une étude approfondie sur les paramètres afin d'atteindre les meilleures solutions.

La première variante présente la version simple du problème en considérant sa matrice d'incidence (produits/machines) seulement et en fixant à l'avance le nombre de cellules. Nous avons proposé deux approches basées sur l'algorithme de sélection clonale pour résoudre cette variante. La première approche (Karoum *et al.*, 2016a) combine l'algorithme avec la méthode de recherche Tabou pour intensifier la recherche autour des régions prometteuses. La deuxième approche (Karoum *et al.*, 2016b) présente une nouvelle adaptation de l'algorithme par l'utilisation d'autres types d'opérateurs génétiques et par l'intégration d'un mécanisme de recherche visant à renforcer la qualité des solutions obtenues. De plus, nous avons adapté l'algorithme de recherche du coucou par deux façons différentes (Karoum et Elbenani, 2017c) et (Karoum *et al.*, 2018) afin de répondre aux différentes contraintes de cette variante. Les résultats de tests montrent que nos méthodes sont comparables, sinon meilleures par rapport à certaines approches tirées de la littérature.

La deuxième variante du problème traite le cas où le nombre de cellules est variable. Un nombre limité de chercheurs ont étudié cette variante à cause de la difficulté rencontrée lors de la manipulation des cellules variables. De notre part, nous avons introduit une nouvelle méthode basée sur l'algorithme de sélection clonale (Karoum et Elbenani, 2017a). L'objectif est d'atteindre les meilleures solutions les plus connues dans la littérature avec le moins de cellules possible. Les solutions obtenues par notre méthode sont comparées avec celles des méthodes récemment développées. L'étude comparative a montré l'efficacité de notre méthode sur un ensemble d'instances du problème.

La troisième variante prend en compte la présence des routes alternatives pour le traitement des produits. De plus, elle considère plusieurs facteurs de production tels que la fiabilité des machines, la capacité des cellules, le temps et le volume de production. La présence de ces facteurs rend le

problème plus réaliste mais aussi plus complexe. Le but majeur est d'optimiser l'utilisation des cellules de fabrication et de minimiser le coût total dû aux traitements des produits et aux défaillances des machines. Dans ce contexte, notre première méthode est basée sur l'algorithme de sélection clonale et elle vise à minimiser le coût des mouvements intercellulaires et à maximiser la fiabilité des machines (Karoum et Elbenani, 2017b). La deuxième méthode proposée est une adaptation de l'algorithme de recherche du coucou (Karoum et Elbenani, 2018). Elle cherche à minimiser le coût des mouvements intercellulaires et intracellulaires, le coût de pannes des machines et le coût d'installation. Les solutions obtenues par ces deux méthodes sont comparées avec celles de la méthode Branch and Bound, en utilisant le logiciel LINGO 16.0. Les résultats obtenus ont montré l'efficacité de nos méthodes en termes de qualité des solutions et du temps de calcul requis.

Nous avons également appliqué notre deuxième approche basée sur l'algorithme de sélection clonale au problème d'orientation de l'équipe (Team Orienteering Problem), qui est considéré comme une généralisation du célèbre problème de tournées de véhicules (Karoum et Elbenani, 2016). Les résultats obtenus ont prouvé que notre méthode est capable de résoudre efficacement d'autres problèmes d'optimisation combinatoire.

Nos travaux ont fait l'objet de publications dans des revues de renommée internationale et spécialisées dans le domaine de l'ingénierie de production telles que *"Neural Computing and Applications"*, *"Production Engineering"* et *"Journal of Computational Methods in Sciences and Engineering"*, et aussi dans des livres de l'éditeur Springer : *"Recent Developments in Metaheuristics"* ; *"Lecture Notes in Electrical Engineering"*.

Organisation de la thèse

Les travaux présentés dans cette thèse sont structurés selon cinq chapitres :

Le premier chapitre introduit le problème de formation de cellules. Au début, nous présentons les différents types d'organisation de la production. Puis, nous soulignons les avantages et les inconvénients de la production cellulaire et ses différentes applications. Ensuite, nous définissons le problème de formation de cellules et ses mesures de performance. Les méthodes de résolution de ce problème avec une seule route de production sont aussi introduites dans ce chapitre. Enfin, nous présentons le modèle mathématique utilisé dans les trois chapitres qui suivent.

Le deuxième chapitre présente les algorithmes inspirés du système immunitaire naturel et leurs applications dans le monde réel. Puis, il s'intéresse aux changements appliqués à l'algorithme de sélection clonale pour l'adapter à notre problème. Les méthodes proposées sont présentées et décrites en détail. Les performances de ces méthodes sont validées sur plusieurs instances de tests et la comparaison est effectuée avec d'autres méthodes collectées de la littérature.

A travers le troisième chapitre, deux nouvelles approches basées sur l'algorithme de recherche du coucou ont été appliquées à notre problème de formation de cellules. Au début, nous présentons le mode de reproduction parasitaire de l'oiseau coucou. Ensuite, nous décrivons l'algorithme de recherche du coucou et ses différentes applications. Puis, nous détaillons nos approches proposées pour résoudre le problème et nous les testons sur un ensemble d'instances pour évaluer leurs qualités.

Le quatrième chapitre traite une deuxième variante du problème où le nombre de cellules est variable. Nous proposons l'algorithme de sélection clonale pour la résoudre. Tout d'abord, nous présentons les changements apportés à cet algorithme pour l'adapter à la nouvelle variante. Puis, nous étudions le paramétrage de l'approche proposée. Les résultats d'évaluation de cette approche sur plusieurs instances de tests sont rapportés dans ce chapitre.

Dans le cinquième chapitre, nous introduisons une autre variante du problème qui prend en considération plusieurs facteurs de production comme les routes alternatives de fabrication de chaque produit, la fiabilité des machines, le volume et le temps de production, etc. Ensuite, nous proposons deux méthodes pour résoudre le problème. La première méthode est une adaptation de l'algorithme de sélection clonale et la deuxième est une application de l'algorithme de recherche du coucou. Les résultats obtenus par ces deux méthodes sont comparés avec ceux de la méthode Branch and Bound, en utilisant le logiciel LINGO.

Pour montrer l'efficacité de nos méthodes, nous avons appliqué celle basée sur l'algorithme de sélection clonale à un autre problème d'optimisation combinatoire connu sous le nom Team orienteering problem ou le problème d'orientation d'équipe. La définition du problème, ses applications et l'état d'art des méthodes proposées pour le résoudre sont présentés dans l'annexe. De plus, une description de la méthode proposée est rapportée suivie des résultats obtenus.

Chapitre

1 Problème de formation de cellules CFP

Sommaire

1.1 Introduction

Le problème abordé dans cette thèse est le problème de formation de cellules (Cell Formation Problem : CFP) qui est considéré comme l'un des premiers problèmes rencontrés dans la conception du système de production cellulaire. Ce problème se pose dans le cas d'implantation ou de réimplantation d'un système de production. La réimplantation se produit lorsque les caractéristiques de fabrication des produits évoluent de manière significative par une augmentation importante du volume demandé ou par une modification de certaines caractéristiques des produits traités.

Dans ce chapitre, nous allons présenter dans un premier temps les différents types d'organisation de la production et nous soulignons l'intérêt que porte la production cellulaire par rapport aux autres types. Dans un second temps, nous définissons le CFP et ses mesures de performance. Puis, nous listons quelques méthodes proposées pour résoudre le problème. Enfin, nous présentons le modèle mathématique utilisé pour formuler notre problème.

1.2 La production cellulaire

1.2.1 Types d'organisation de la production

L'étude des systèmes de production montre l'existence de plusieurs types d'organisation de la production. Les plus connus sont l'organisation linéaire, l'organisation fonctionnelle, l'organisation cellulaire, et l'organisation fractale. Chaque entreprise doit adopter un type qui convient le mieux à son secteur d'activité, sa relation avec ses clients et son organisation interne.

Organisation linéaire : Ce mode d'organisation, dit aussi *flow shop*, est recommandé pour la fabrication du même type de produit en grande quantité (production de masse). Les machines sont placées en ligne dans l'ordre logique des activités de fabrication du produit (Figure 1.1).
Durant leurs passages devant les différents postes (machines), les produits subissent un ensemble d'opérations de transformation ou d'assemblage, conduisant à la création d'un produit ou d'une famille de produits manufacturés.

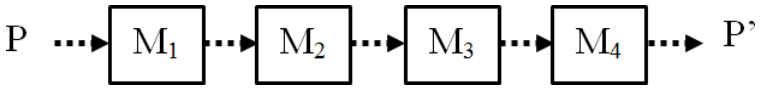


FIGURE 1.1 – Organisation linéaire

L'organisation linéaire présente une multitude d'avantages telle que la simplicité des flux, taux de productivité élevé, manutention faible, et temps de passage réduit. Mais elle présente aussi des inconvénients au niveau de la grande fiabilité des équipements exigée, l'investissement élevé dans les équipements, ainsi que la flexibilité relativement faible concernant la réingénierie de produits et le routage des produits.

Organisation fonctionnelle : ce type est adopté lorsqu'une entreprise fabrique un grand nombre de produits de différents types avec des lots de petites tailles. Il consiste à placer les ressources effectuant la même fonction dans le même département (Figure 1.2). Le transfert des produits entre les départements s'effectue selon leurs gammes de fabrication.

L'organisation fonctionnelle, aussi connu par *job shop*, est caractérisée par sa grande flexibilité, un taux élevé d'utilisation des processeurs, une forte spécialisation des équipements et du personnel au niveau des fonctions, ainsi qu'une polyvalence élevée des opérateurs au niveau des produits à l'intérieur de chaque département. Néanmoins, ses flux sont compliqués. Le temps de passage de production (lead time) et le coût de gestion de la matière première sont élevés. Elle demande beaucoup de manutentions et elle a un manque au niveau du travail d'équipe à cause de la spécialisation des opérateurs à l'intérieur des ateliers ou départements. De plus, il a été démontré que les pièces dépensent jusqu'à 95% de leur temps de fabrication dans des files d'attente pour accéder à une machine ou pour passer d'une machine à l'autre (Askin et Standridge, 1993).

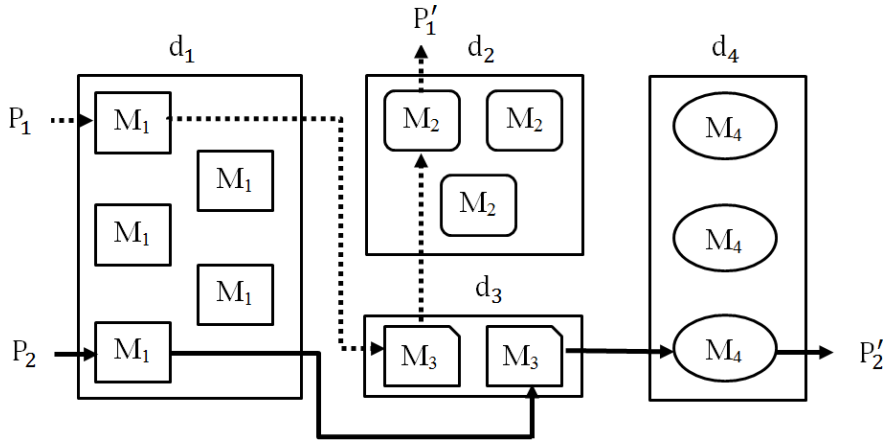


FIGURE 1.2 – Organisation fonctionnelle

Organisation fractale : consiste à décomposer le système de fabrication en plusieurs cellules dont chacune contient, dans la majorité des cas, une copie de tous les types de processeurs se trouvant dans le système ((Venkatadri *et al.*, 1997) et (Montreuil, 1999)). Cela implique que chaque cellule est capable de traiter n'importe quel produit, mais certaines sont plus adaptées pour un produit particulier que d'autres.
L'organisation fractale est caractérisée par la simplicité des flux et la grande flexibilité des centres ou cellules. Mais, la consommation considérable des espaces influence l'efficacité de ce type d'organisation qui reste, dans la plupart des cas, plus performante comparativement aux autres types présentés précédemment.

Organisation cellulaire : peut être considérée comme une application de la Technologie de Groupe (*Group Technology*). Elle consiste à regrouper les machines dédiées au traitement des produits similaires dans la même cellule (Figure 1.3). Les produits similaires, de point de vue forme ou processus de production, sont groupés pour former une famille de produits et les machines nécessaires à la fabrication de chaque famille sont assemblées en groupes.

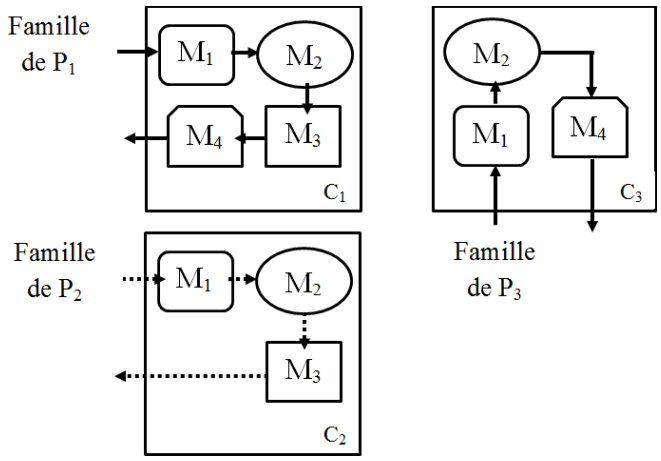


FIGURE 1.3 – Organisation cellulaire

Ce type d'organisation combine les avantages des deux organisations susmentionnées et tente de faciliter la gestion en décomposant le système de fabrication entier en plusieurs sous-systèmes

presque indépendants. Les machines sont assemblées de telle sorte que les distances entre les machines de la même cellule soient faibles et que chaque cellule contient presque toutes les machines nécessaires au traitement de la famille de produits. Cela permet de simplifier les flux, tout comme l'organisation linéaire, et assurer la flexibilité dans la fabrication d'une grande variété de produits, similaire à l'organisation fonctionnelle.

Les principaux avantages de la production cellulaire peuvent être listés comme suit ((Wemmerlöv et Hyer, 1986), (Goldengorin *et al.*, 2013)) :

— Minimisation du temps d'installation : chaque cellule de production vise à fabriquer des pièces similaires (forme, taille, etc.) dans la même entité. Ce qui garantit une réduction du temps de réglage des machines et une standardisation des outils.

— Réduction du coût et du temps de manutention : presque chaque produit est traité dans une seule cellule. Le temps de déplacement des produits et la distance entre les machines sont minimes. Ainsi, tous les flux sont concentrés dans les cellules, ce qui réduit le temps d'écoulement. Kusiak (2000) a montré qu'une réduction de 20% à 80% dans les coûts de manutention de matériaux peut être obtenue en utilisant des cellules de production.

— Minimisation du temps de traitement : le temps du déplacement et du transfert de chaque produit à la machine suivante est réduit. Chaque produit est transféré immédiatement à la machine suivante après son traitement, contrairement à l'organisation fonctionnelle où les produits sont transférés aux autres machines sous forme des lots.

— Réduction de la taille des lots : la diminution du temps de traitement permet de travailler avec des lots de tailles plus petites et plus économiques. En conséquence, le flux de production devient plus fluide.

— Réduction des besoins d'outillage : les produits fabriqués dans une cellule ont la même forme, taille et composition et nécessitent généralement les mêmes outils.

— Réduction du stock des produits en cours (WIP : work-in-process) et les produits finis : Askin et Standridge (1993) ont montré que WIP pourrait être réduit de 50% lorsque le temps de traitement est réduit en moitié. Cette réduction diminue également le délai de livraison des commandes.

— Minimisation du stock des produits en cours et les besoins d'outillage permet de gagner de l'espace. Ceci, à son tour, peut être utilisé pour rapprocher les machines les unes des autres et diminuer davantage les coûts de manutention.

— Facilité dans l'ordonnancement et la planification de la production : en général, le nombre élevé des machines et des produits complique la gestion de la production. Mais grâce à la production cellulaire, la décomposition du système de fabrication en plusieurs cellules, pratiquement indépendantes, facilite la planification.

— Amélioration de la qualité du produit : Les pièces sont produites dans des petites zones et la rétroaction est immédiate. Ainsi, le processus peut être arrêté rapidement en cas d'une panne.

Toutefois, la production cellulaire a un certain nombre d'inconvénients :

— Coût de mise en œuvre élevé : la réorganisation d'une configuration existante peut s'avérer coûteuse. Les coûts comprennent l'identification des cellules de fabrication optimales et les familles de pièces, la réorganisation physique (déplacement des machines), des formations supplémentaires du personnel, etc.

— Difficultés d'équilibrage de la surcharge du travail et manque de robustesse : chaque machine est essentielle pour le bon fonctionnement de la cellule. La panne d'une machine peut provoquer le non-fonctionnement de la cellule entière. Pour éviter d'arrêter la production, le processus peut être transféré à une machine alternative dans une autre cellule, mais cette solution entraîne des déplacements intercellulaires.

— Demande d'employés spécialisés : le groupement de différents types de machines, dans la même cellule, nécessite des employés très spécialisés dans chaque cellule et cela peut s'avérer coûteux.

— Synchronisation des pièces pour un montage ultérieur : des mesures et des ressources supplémentaires (par exemple l'espace de stockage) sont nécessaires pour manipuler des pièces qui sont traitées dans des cellules différentes mais qui doivent être assemblées.

— Utilisation faible des machines : en augmentant le nombre de machines pour générer des cellules indépendantes, l'utilisation moyenne de la machine devient plus faible. De plus, si deux ou plusieurs cellules contiennent des machines équivalentes, le problème d'équilibrage de charge peut se produire lorsqu'une de ces machines est sous-utilisée tandis que l'autre est surchargée.

Malgré ses inconvénients, la production cellulaire améliore la performance du système dans le cas de fabrication de produits de grandes variétés et de moyens volumes. L'objectif principal de la production cellulaire est de pouvoir fabriquer des pièces complètement finies dans des cellules sans le besoin de recourir aux mouvements intercellulaires.

Pour résumer, le choix d'un des types d'organisation est une tâche primordiale pour chaque entreprise. D'après la littérature, deux critères conditionnent principalement le type d'organisation retenue pour les systèmes de production : la variété et les volumes de production. Le terme « variété » indique la variété des processus, et non des produits. Par exemple, dans l'industrie automobile, la variété des produits finis est très grande ; en revanche, le processus de réalisation est pratiquement identique pour tous ces produits, ce qui permet de travailler sur une ligne d'assemblage unique.

Plusieurs études comparatives ont été effectuées sur les différents types d'organisation. La majorité de ces études ont abouti à la supériorité de l'organisation cellulaire par rapport aux autres types, comme illustré dans la Figure 1.4 (Goldengorin *et al.*, 2013). Dans cette thèse, nous allons utiliser l'appellation *production cellulaire* pour faire référence à la production de type organisation cellulaire.

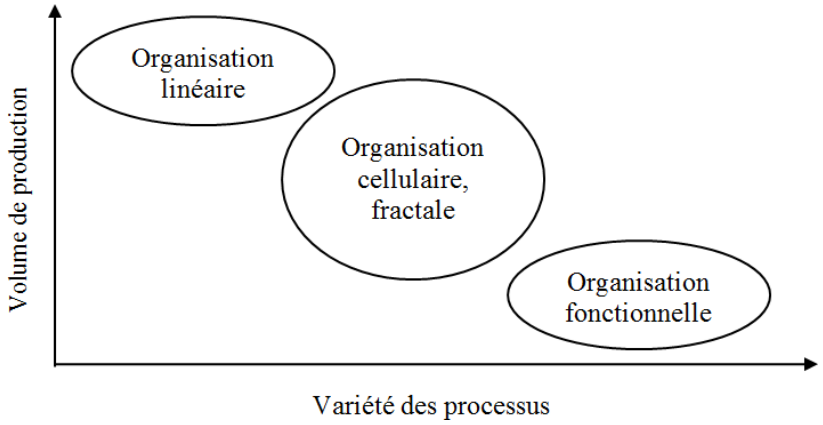


FIGURE 1.4 – Comparaison entre les types d'organisation de la production

Comme nous l'avons déjà mentionné, la production cellulaire (*Cellular Manufacturing*) peut être définie comme une application de la Technologie de Groupe. Cette technologie est une philosophie de fabrication introduite formellement par le soviétique Mitrofanov en 1959 (Mitrofanov, 1966). Elle consiste à identifier et regrouper des pièces similaires par familles afin de tirer profit de leurs analogies. Elle implique le regroupement de machines ou de processus sur la base des parties ou des familles de pièces qu'ils traitent (Wemmerlöv et Hyer, 1986).

Le processus de conception de ce type de production suit plusieurs étapes :

1. Regrouper les différentes machines en cellules de production ou de fabrication.
2. Ajouter de nouvelles machines si nécessaire.
3. Déterminer la famille de produits en fonction de leurs exigences de traitement.
4. Associer chaque famille de produits à une cellule spécifique.
5. Former les employés.

1.2.2 Applications de la production cellulaire

Pendant les vingt dernières années, la production cellulaire a été adoptée dans différents types d'industries et d'applications telles que la fabrication des cadres en bois (Dinis-Carvalho *et al.*, 2014), le forgeage des métaux (Hung et Maleki, 2014), l'assemblage des composants de munitions pour les applications de défense (Pattanaik et Sharma, 2009), la fabrication de tracteurs (Shayan et Sobhanallahi, 2002). De plus, la production cellulaire est appliquée dans les industries des métaux-mécaniques (Johnson et Wemmerlöv, 2004), les industries électroniques (Levasseur *et al.*, 1995), et le secteur de la santé (Landsbergis *et al.*, 1999). En fait, ce type de production est efficace dans les entreprises qui produisent une variété de produits.

Le premier problème rencontré pour concevoir un système de production cellulaire est le problème de formation de cellules (Cell Formation Problem : CFP). Par la suite, nous allons aborder ce problème et l'état de l'art des méthodes proposées pour le résoudre. L'objectif est d'analyser les travaux de la communauté scientifique pour identifier les approches, les méthodes et les techniques utilisées pour la formation des cellules de fabrication.

1.3 Problème de formation de cellules

1.3.1 Définition

Le problème de formation de cellules est un problème d'optimisation NP-difficile (Dimopoulos et Zalzala, 2000). Il est souvent présenté par une matrice binaire appelée la matrice d'incidence A où chaque ligne représente une machine et chaque colonne représente un produit. Chaque élément de la matrice montre l'interaction entre une machine et un produit.

Considérons les notations suivantes :
i : Indice de machines ;
j : Indice de produits ;
k : Indice de cellules ;
M : Nombre de machines ;
P : Nombre de produits ;
C : Nombre de cellules ;
I : Ensemble de M machines ($\{1, \ldots, M\}$) ;
J : Ensemble de P produits ($\{1, \ldots, P\}$) ;
K : Ensemble de C cellules ($\{1, \ldots, C\}$).
La matrice d'incidence $A = [a_{ij}]$ indique l'interaction entre les machines et les produits :

$$a_{ij} = \begin{cases} 1, & \text{si la machine } i \text{ traite le produit } j \\ 0, & \text{sinon} \end{cases}$$

Un produit j peut être traité par plusieurs machines. Une cellule de production $k(k = 1, \ldots, C)$ comprend un sous-ensemble de machines (groupe de machines) $I_k \subset I$ et un sous-ensemble de produits (famille de produits) $J_k \subset J$. Le problème consiste à déterminer une solution comprenant des cellules de production $C(I, J) = \{(I_1, J_1), \ldots, (I_C, J_C)\}$ aussi autonomes que possible avec :

$I_1 \cup \ldots \cup I_C = I$ et $J_1 \cup \ldots \cup J_C = J$ tel que :
$I_{k_1} \cap I_{k_2} = \emptyset$ et $J_{k_1} \cap J_{k_2} = \emptyset$ pour $\forall k_1$ et $k_2 \in \{1, \ldots, C\}, k_1 \neq k_2$

Pour mieux expliquer le concept des cellules de production, considérons la matrice d'incidence machine-produit illustrée dans la Figure 1.5. Cette matrice modélise l'intersection entre 7 machines et 11 produits. La machine 1 par exemple traite les produits : 1, 3, 7 et 11.

Produits		1	2	3	4	5	6	7	8	9	10	11
Machines	1	1	0	1	0	0	0	1	0	0	0	1
	2	1	1	0	0	0	1	0	0	0	0	0
	3	0	1	0	0	0	1	0	0	1	0	0
	4	0	0	0	1	1	0	0	0	0	1	0
	5	0	0	1	0	0	0	1	0	0	0	0
	6	0	0	1	1	0	0	0	0	0	0	1
	7	0	0	0	0	1	0	0	1	0	1	0

FIGURE 1.5 – Matrice d'incidence initiale : 7 x 11

Figure 1.6 présente la meilleure solution obtenue du problème par partition en quatre cellules différentes introduites par les blocs gris :
Cellule 1 : le groupe de machines $I_1 = \{2, 3\}$ et la famille de produits $J_1 = \{1, 2, 6, 9\}$.
Cellule 2 : le groupe de machines $I_2 = \{1, 5\}$ et la famille de produits $J_2 = \{3, 7\}$.
Cellule 3 : le groupe de machines $I_3 = \{4, 7\}$ et la famille de produits $J_3 = \{5, 8, 10\}$.
Cellule 4 : le groupe de machines $I_4 = \{6\}$ et la famille de produits $J_4 = \{4, 11\}$.
Dans la Figure 1.6, les entrées (1,1), (1,11), (4,4) et (6,3) ayant une valeur 1 en dehors des blocs diagonaux gris présentent les éléments exceptionnels de la solution. Tandis que les éléments vides correspondent aux entrées ayant une valeur 0 à l'intérieur des blocs diagonaux (2, 9), (3, 1) et (4, 8).

Produits		1	2	6	9	3	7	5	8	10	4	11
Machines	2	1	1	1	0	0	0	0	0	0	0	0
	3	0	1	1	1	0	0	0	0	0	0	0
	1	1	0	0	0	1	1	0	0	0	0	1
	5	0	0	0	0	1	1	0	0	0	0	0
	4	0	0	0	0	0	0	1	0	1	1	0
	7	0	0	0	0	0	0	1	1	1	0	0
	6	0	0	0	0	1	0	0	0	0	1	1

FIGURE 1.6 – Solution de la matrice d'incidence : 7 x 11

1.3.2 Complexité du problème

Dimopoulos et Zalzala (2000) ont montré que le problème de formation de cellules est considéré comme l'un des problèmes d'optimisation combinatoire NP-difficile. L'augmentation de la taille du problème entraîne une augmentation exponentielle du temps de calcul.
Le nombre de solutions possibles peut être déterminé à l'aide du nombre de Stirling de seconde espèce : $S(N,C) = \frac{C^N}{C!}$ où N est le nombre d'éléments (produits et machines) à regrouper en C cellules.
Si le nombre de cellules n'est pas fixé à l'avance, il peut être compris entre 1 et N. Par conséquent, le nombre de solutions réalisables devient plus grand ; ce nombre sera de l'ordre de : $\sum_{C=1}^{N} \frac{C^N}{C!}$.

1.3.3 Mesures de performance

Plusieurs mesures de performance sont proposées en littérature pour évaluer l'efficacité de la décomposition cellulaire obtenue pour une solution s. Les plus fréquemment utilisées sont (Keeling *et al.*, 2007) :

— Efficience de groupement (*Grouping efficiency*)

— Efficacité de groupement (*Grouping efficacy*)

Ces mesures sont utilisées dans le cas où la matrice d'incidence est binaire. En effet, elles sont indépendantes des paramètres de production comme la durée des opérations, la séquence des produits, la capacité des machines, etc.

Notations :
a_1 : Nombre total des 1 dans la matrice d'incidence ;
a_0 : Nombre total des 0 dans la matrice d'incidence ;
a_1^{in} : Nombre total des 1 dans les blocs diagonaux d'une solution s ;
a_1^{out} : Nombre total d'éléments exceptionnels en dehors des blocs diagonaux de s ;
a_0^{in} : Nombre total d'éléments vides dans les blocs diagonaux de s ;
a_0^{out} : Nombre total des 0 en dehors des blocs diagonaux de s ;

1.3.3.1 Efficience de groupement

Chandrasekharan et Rajagopalan (1989) ont introduit une première mesure quantitative, intitulé l'efficacité de regroupement η, pour évaluer la qualité d'une solution s. Cette mesure est définie comme suit :

$$\eta(s) = \alpha\eta_1(s) + (1-\alpha)\eta_2(s) \tag{1.1}$$

Avec :

— $\eta_1(s)$ représente le rapport entre le nombre des éléments unitaires et le nombre total des éléments dans les blocs diagonaux d'une solution s, c-à-d le pourcentage des éléments unitaires dans les blocs diagonaux.

$$\eta_1(s) = \frac{a_1^{in}}{a_1^{in} + a_0^{in}} \tag{1.2}$$

— $\eta_2(s)$ représente le rapport entre le nombre des éléments exceptionnels et le nombre total des éléments en dehors des blocs diagonaux d'une solution s, c-à-d le pourcentage des éléments exceptionnels en dehors des blocs diagonaux.

$$\eta_2(s) = \frac{a_1^{out}}{a_1^{out} + a_0^{out}} \tag{1.3}$$

— α est un coefficient de poids à choisir selon l'importance qu'on veut attribuer à l'un des deux termes $0 \leq \alpha \leq 1$.

1.3.3.2 Efficacité de groupement

Afin d'éviter les difficultés de fixation du paramètre α mentionné précédemment, Suresh Kumar et Chandrasekharan (1990) ont proposé une nouvelle mesure nommée *efficacité de groupement* d'une solution s :

$$Eff(s) = \frac{(a_1 - a_1^{out})}{(a_1 + a_0^{in})} \tag{1.4}$$

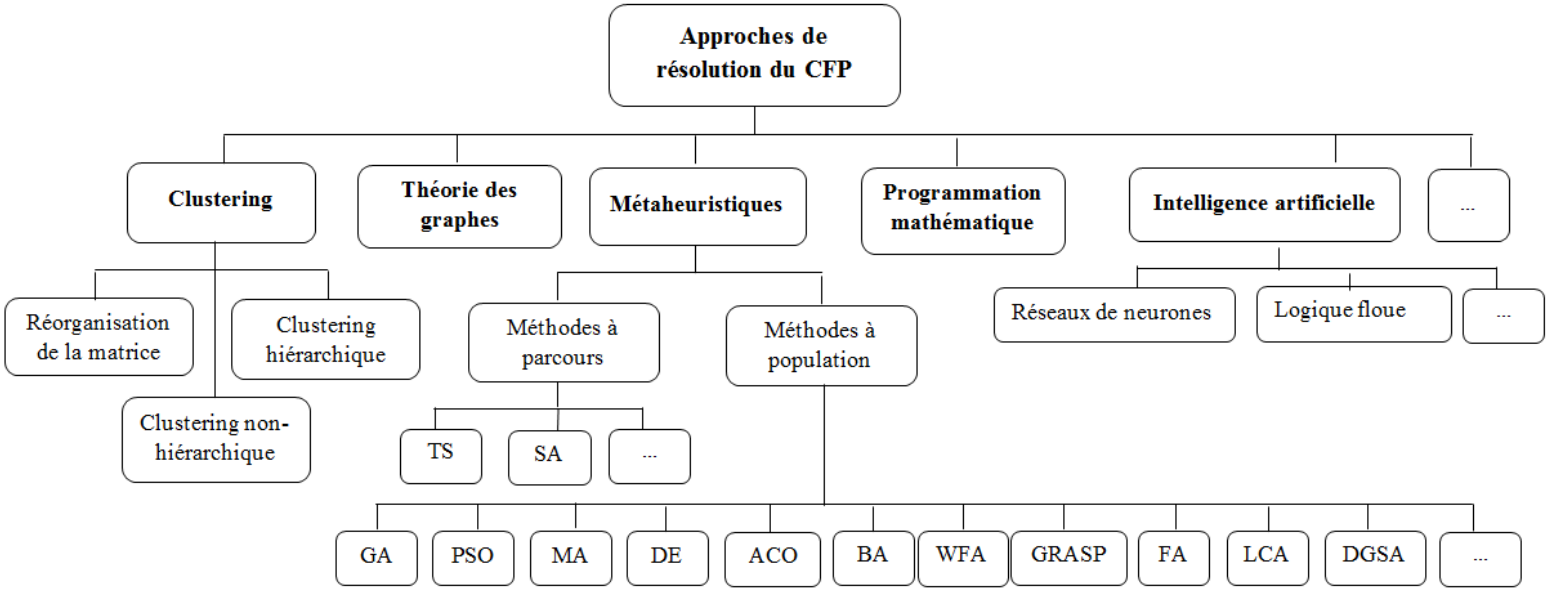
L'intégration de la taille de la matrice dans l'Equation (1.4) permet d'améliorer la solution finale par rapport à l'efficience du groupement. En plus, l'efficacité de groupement a une grande capacité à différencier entre les matrices bien structurées et mal structurées lorsque les blocs diagonaux contiennent un nombre élevé d'éléments vides. À ce jour, l'efficacité de groupement d'une solution est la mesure la plus utilisée dans la littérature pour la résolution du CFP.

1.4 Etat de l'art

Au cours des trois dernières décennies, un grand nombre de documents de recherche et de rapports pratiques ont été publiés dans le domaine de la production cellulaire. Dans cette section, nous allons présenter quelques méthodes proposées pour réaliser un système de production cellulaire. Ces méthodes peuvent être classifiées comme suit :

— Clustering ou analyse de regroupement ;
— Théorie des graphes ;
— Programmation mathématique ;
— Métaheuristiques ;
— Intelligence artificielle.

La Figure 1.7 détaille les différentes méthodes rapportées dans notre thèse.



TS: Tabu Search; SA: Simulated Annealing; GA: Genetic Algorithm; PSO: Particle Swarm Optimisation; MA: Memetic Algorithm; DE: Differential Evolution; ACO: Ant Colony Optimization; BA: Bees Algorithm; WFA: Water Flow-like Algorithm; GRASP: Greedy Randomized Adaptive Search Procedure; FA: Firefly Algorithm; LCA: League Championship Algorithm; DGSA: Discrete Gravitational Search Algorithm.

FIGURE 1.7 – Classement de quelques approches de résolution du CFP

1.4.1 Méthodes de clustering

1.4.1.1 La réorganisation de la matrice

Les méthodes de la réorganisation de la matrice (Array-Based Clustering) sont largement utilisées dans la littérature. Elles consistent à arranger l'ordre des lignes et des colonnes de la matrice afin d'obtenir des blocs diagonaux. Ces blocs permettent d'identifier les familles de produits et les groupes de machines nécessaires pour fabriquer ces produits. Plusieurs méthodes ont été déployées pour la réorganisation de la matrice d'incidence dont les plus populaires sont :

- L'algorithme d'énergie de liaisons (*Bond Energy Algorithm : BEA*) développé par (McCormick *et al.*, 1972). Il tente d'identifier et de créer de fortes liaisons entre les machines et les produits au sein de chaque cellule. D'après (McCormick *et al.*, 1972), ces liaisons sont considérées comme une énergie évaluée par une mesure d'efficacité (ME). Le but est de maximiser la valeur totale de ME en permutant les lignes et les colonnes des cellules. ME est applicable à des matrices de n'importe quelle taille et forme ; la seule exigence est la non-négativité des éléments. L'inconvénient de cette méthode est la possibilité d'avoir un nombre très élevé de permutations pour des matrices d'incidence de grande taille.
- La classification par ordre de rang (*Rank Order Clustering : ROC*) présentée par (King, 1980). C'est une méthode plus sophistiquée qui limite le nombre de permutations des lignes et des colonnes de la méthode BEA précédente en affectant une valeur (puissance de 2) à chaque élément (machine/produit) de la matrice. Malgré sa simplicité, la configuration initiale de la matrice d'incidence et les valeurs attribuées aux éléments influencent beaucoup la qualité des résultats. Une version améliorée, appelée *ROC2*, a été développée par (King et Nakornchai, 1982) pour surmonter cette limitation et augmenter l'efficacité du calcul.
- L'algorithme de classification directe (*Direct Clustering Algorithm : DCA*) est une méthode itérative proposée par (Chan et Milner, 1982). Elle consiste à compter le nombre des 1 pour chaque ligne et pour chaque colonne de la matrice initiale puis ordonner les lignes dans l'ordre croissant et les colonnes dans l'ordre décroissant. Le but est d'augmenter le nombre des 1 dans les blocs diagonaux. Cette méthode garantit une souplesse au niveau de la taille du problème. En outre, *DCA* initie la procédure en comptant le nombre des 1 au lieu de dépendre d'une forme d'intuition comme c'est le cas pour l'algorithme *ROC*.

Les trois méthodes sont simples mais elles présentent plusieurs inconvénients. Elles ne considèrent pas l'intervention d'autres paramètres de conception cellulaire comme les durées opératoires des produits sur les machines et la capacité maximale de chaque cellule. De plus, elles nécessitent une inspection manuelle pour valider la composition finale des cellules de production. Il semble que la recherche scientifique concernant l'axe du développement des méthodes de la réorganisation de la matrice d'incidence s'est abandonnée depuis les années 90 (Hachicha, 2009).

1.4.1.2 Clustering hiérarchique

Le but du clustering hiérarchique (*hierarchical clustering*) est de créer une hiérarchie de classes (clusters). Pour former une telle hiérarchie, deux principales techniques peuvent être utilisées :

- La méthode ascendante ou agglomérative démarre avec autant de classes qu'il y'a d'éléments puis procède à leur fusion en classes de plus en plus grandes jusqu'à l'obtention d'une classe contenant tous les éléments.
- La méthode descendante ou divisive où tous les éléments (machines ou produits) sont regroupés dans une seule classe ; ensuite une série de partitions est réalisée de manière à ce que les

classes résultantes soient différentes le plus possible.

Ces deux méthodes se basent sur la définition d'une mesure de similarité. L'objectif est de rassembler les éléments similaires ensemble selon leur ressemblance.

McAuley (1972) est parmi les premiers chercheurs considérant une approche de clustering hiérarchique itérative fondée sur des mesures de similarité. Il a proposé un algorithme de clustering à liaison unique (Single Linkage Clustering Algorithm : SLCA) dans lequel le coefficient de similarité entre deux classes est défini en se basant sur l'indice de Jaccard. Egalement, (Carrie, 1973) et (Mosier et Taube, 1985b) ont utilisé le même algorithme pour résoudre le problème avec des améliorations au niveau du coefficient de similarité.

Seifoddini et Wolfe (1986) ont développé un modèle basé sur l'algorithme de clustering à liaison moyenne (*Average Linkage Clustering algorithm : ALC*). Le coefficient de similarité entre deux cellules est défini comme la moyenne des coefficients de similarité entre tous les éléments des deux cellules.

Chow et Hawaleshka (1992) ont proposé une nouvelle procédure de regroupement spécifique pour la formation de cellules appelée "fusion d'ensemble" (*Set Merging : SM*). Cette procédure vise à calculer la similarité entre les cellules au lieu des éléments dans chaque cellule contrairement aux algorithmes *SLCA* et *ALC*.

Islam et Sarker (2000) ont présenté un nouveau coefficient de similarité entre les machines (ou les produits) et qui prend en considération un ensemble de propriétés de similarité : aucune incompatibilité, aucune correspondance, correspondance minimale, correspondance complète et correspondance maximale. De plus, ils ont proposé un modèle mathématique et une approche heuristique qui utilisent un tel coefficient pour résoudre le CFP.

Plusieurs chercheurs ont utilisé les algorithmes de clustering hiérarchique pour former des cellules de production. Cependant, de nombreuses études (Anderberg, 1973) et (Chandrasekharan et Rajagopalan, 1986b) ont souligné des inconvénients liés à cette approche telles que :

— La nécessité des informations supplémentaires : le nombre de cellules est souvent prédéfini en tant que paramètre d'entrée. Parfois le nombre maximum d'éléments dans chaque cellule est également prédéterminé.

— L'irréversibilité : une fois deux machines ou produits sont liés, il est impossible de les séparer même si la liaison conduit à une solution non optimale.

1.4.1.3 Clustering non-hiérarchique

Les méthodes de clustering non-hiérarchique (*non-hierarchical clustering*) sont des méthodes itératives faciles à implémenter. Elles utilisent des mesures de similarité ou de dissemblance pour regrouper les machines ou les produits. Elles commencent par une partition initiale de l'ensemble de données ou par un choix aléatoire de quelques points de départ (seed points). De plus, le nombre de cellules doit être défini à l'avance dans les deux cas.

Chandrasekharan et Rajagopalan (1986a) ont proposé une méthode de regroupement non hiérarchique de machines et de produits appelée *Ideal Seed Non-hierarchical Clustering Algorithm (ISNC)*. Le nombre de cellules est déterminé en utilisant la théorie des graphes. Tout d'abord, une méthode de clustering non-hiérarchique est utilisée pour le groupement des machines en cellules et des produits en familles. Ensuite, une méthode *ideal seed* est adoptée pour l'amélioration du regroupement initiale. Les mêmes auteurs, Chandrasekharan et Rajagopalan (1987), ont présenté une version améliorée de l'algorithme ISNC nommée *ZODIAC* acronyme de *Zero One Data : Ideal seed Algorithm for Clustering* et ils ont développé un nouveau concept 'efficacité relative' comme règle d'arrêt aux itérations.

Srinivasan et Narendran (1991) ont développé la méthode GRAFICS (GRouping using Assignment method For Initial Cluster Seeds) pour résoudre le problème de formation de cellules CFP. Les cellules initiales des machines sont obtenues grâce à une méthode d'affectation introduite par (Srinlvasan *et al.*, 1990). La comparaison avec ZODIAC montre une bonne performance de GRAFICS en matière d'efficacité de regroupement et du temps d'exécution.

Malgré que les méthodes de clustering sont simples à implémenter et capables d'obtenir des solutions dans des durées raisonnables, elles présentent un inconvénient majeur : un seul objectif est pris en compte tel que la minimisation des mouvements intercellulaires. D'autres informations telles que le temps de production, le volume de production et la séquence des opérations ne sont pas intégrées dans le processus de conception. Ainsi, les solutions obtenues peuvent être valables dans des situations limitées.

1.4.2 Méthodes de la théorie des graphes

Dans la littérature, un certain nombre de chercheurs a adopté des méthodes basées sur la théorie des graphes pour former des cellules de production. On peut citer quelques exemples :

Rajagopalan et Batra (1975) ont développé une approche de partitionnement graphique pour la formation des cellules de production. Les sommets du graphe correspondent aux machines et les arcs représentent les relations entre les machines selon les pièces qui les utilisent. Cette relation est évaluée par un coefficient de similarité. Une fois les cellules de production sont formées, un ensemble de règles est utilisé pour affecter les produits aux cellules.

Ng (1993) a employé l'algorithme d'arbre couvrant de poids minimal (*minimum spanning tree : MST*) et il a proposé une nouvelle mesure de performance appelée la mesure d'efficacité de regroupement pondéré (*weighted grouping efficacy*). De plus, il a réalisé une analyse du pire des cas de l'algorithme MST en utilisant les deux mesures de performance : l'efficacité de regroupement et l'efficacité de regroupement pondéré.

L'une des approches les plus utilisées pour la résolution du CFP repose sur le problème p-médian (*P-Median Problem : PMP*). Le but est d'obtenir une décomposition optimale du graphe en arbres constitués d'une racine et de quelques feuilles sans nœuds internes. Parmi les auteurs proposant des améliorations du modèle p-médian pour le CFP, nous citons (Wang et Roze, 1997), (Deutsch *et al.*, 1998), (Won et Lee, 2004) et (Ashayeri *et al.*, 2005)

1.4.3 Méthodes de la programmation mathématique

Boctor (1991) a formulé le problème de formation de groupes de machines comme étant un modèle linéaire en nombres entiers mixtes. L'objectif du modèle est de minimiser le nombre d'éléments exceptionnels. Il permet au concepteur du système cellulaire de choisir le nombre de cellules et de contrôler leur taille. Pour traiter les instances de grande taille, une approche basée sur la méthode de recuit simulé a été introduite.

Étant donné que la duplication des machines peut être plus coûteuse, Foulds *et al.* (2006) ont développé un modèle de programmation mathématique en nombres entiers mixtes qui prend en considération l'introduction de nouvelles types de machines pour ne pas les dupliquer. L'objectif est de minimiser le coût de changement des machines et le coût des déplacements intercellulaires.

Paydar *et al.* (2011) ont développé un nouveau modèle mathématique pour identifier simultanément les groupes de machines et les familles de produits, de telle sorte que le nombre d'éléments vides et exceptionnels soit réduit au minimum. Leur approche a la souplesse nécessaire pour permettre au concepteur de cellules d'identifier le nombre de cellules requis. Le modèle proposé est résolu par

le logiciel LINGO qui utilise la méthode Branch & Bound. Les auteurs ont suggéré d'utiliser des métaheuristiques pour résoudre ce modèle dans un temps raisonnable.

Elbenani et Ferland (2012) ont introduit un modèle de programmation linéaire en nombres entiers mixtes fractionnaires pour le problème de formation de cellules. Ils ont également proposé une approche basée sur l'algorithme de Dinkelbach pour résoudre plusieurs instances de tests à l'aide du logiciel CPLEX. Les résultats obtenus ont permet de prouver pour la première fois l'optimalité de nombreuses solutions de la littérature. Ces résultats sont utilisés pour tester l'efficacité des métaheuristiques proposées pour résoudre le problème CFP.

Bychkov *et al.* (2014) ont présenté une formulation linéaire en nombres entiers mixtes pour le CFP avec un nombre variable de cellules de production. Ils ont suggéré une approche exacte qui permet de remplacer le problème de programmation fractionnaire par plusieurs problèmes de programmation en nombres entiers. L'approche proposée est efficace pour les instances de petite taille, mais elle demande plus de temps d'exécution pour les instances de moyenne et de grande taille.

Bychkov et Batsyn (2017) ont développé un nouveau modèle de programmation linéaire en nombres entiers mixtes où le nombre de cellules est variable. Ensuite, ils ont appliqué l'algorithme de Dinkelbach pour résoudre le modèle. Ce dernier est basé sur les relations machine-machine et produit-machine au lieu de la relation la plus utilisée : machine-produit-cellule.

Trois limitations peuvent être soulignées pour ce type de méthodes de programmation mathématique. Tout d'abord, à cause de la forme non linéaire résultante de la fonction objectif, la plupart des méthodes ne déterminent pas simultanément les groupes de machines et les familles de produits. Deuxièmement, le nombre de cellules doit être précisé à l'avance. En outre, ces méthodes nécessitent un temps de calcul très élevé, en particulier pour les instances de grande taille.

1.4.4 Métaheuristiques

Grâce au rôle primordial de la métaheuristique dans la résolution des problèmes d'optimisation combinatoire, plusieurs chercheurs l'ont appliqué pour résoudre le problème de formation de cellules de production.

Les métaheuristiques peuvent être classées de nombreuses façons. Dans cette thèse, nous avons adopté le classement en métaheuristiques à parcours (*Trajectory based methods*) et métaheuristiques à population (*Population based methods*). La méthode du recuit simulé, par exemple, appartient au premier type de métaheuristiques puisqu'elle traite une seule solution dans chaque itération. D'autre part, les algorithmes génétiques et l'optimisation par essaims particulaires (PSO) sont considérés des métaheuristiques à population car ils manipulent un ensemble de solutions (chromosomes ou particules) à chaque itération.

1.4.4.1 Métaheuristiques à parcours

Recherche Tabou : La recherche Tabou (Tabu search : TS) est considérée l'une des métaheuristiques les plus réussies pour les problèmes NP-complets. Une introduction complète de cette méthode peut être trouvée dans le livre de (Clover et Laguna, 1997).

Sun *et al.* (1995) ont modélisé le CFP comme un problème de partition graphique dans le but de minimiser les flux de matériaux entre les cellules et ils ont développé un algorithme d'amélioration itératif basé sur la méthode Tabou pour résoudre le problème. Cet algorithme a amélioré la configuration existante des cellules par l'intégration du schéma *Look-ahead* (*backtracking*).

Onwubolu et Songore (2000) ont abordé le CFP avec trois fonctions objectives : minimiser les mouvements intercellulaires, minimiser la variation de la charge cellulaire et combiner les deux objectifs précédents. Ainsi, ils ont conçu une méthode Tabou en spécifiant le nombre de cellules requis à priori et en imposant des limites inférieures et supérieures sur la taille des cellules.

Spiliopoulos et Sofianopoulou (2003) ont proposé une approche en trois étapes pour le problème de conception des cellules de fabrication où l'étape principale est basée sur l'algorithme Tabou. Ils ont intégré les structures de mémoire à court et à long terme et une stratégie de recherche globale pour les utiliser.

Schaller (2005) a développé deux nouvelles procédures heuristiques basées sur Tabou : TSH et CBTSH pour la formation des cellules de production. Ils ont comparé les résultats obtenus avec des méthodes collectées de la littérature. Les études ont montré que la deuxième procédure est capable de traiter des instances de grande taille.

Adenso-Díaz *et al.* (2005) ont introduit des coefficients de similarité pour évaluer la qualité de la solution obtenue avec des limites sur le nombre de machines et de pièces dans chaque cellule. Ensuite, ils ont appliqué la méthode Tabou pour améliorer les résultats obtenus par d'autres heuristiques.

Solimanpur et Elmi (2013) ont proposé un modèle de programmation linéaire à nombre entier mixte pour le problème de formation de cellules. Puis, ils ont introduit une méthode Tabou imbriquée pour résoudre le modèle de façon heuristique.

Recuit simulé : Le recuit simulé (Simulated annealing : SA) est une méthode de recherche aléatoire générale introduite par (Kirkpatrick *et al.*, 1983). Son but est de trouver des solutions optimales pour les problèmes d'optimisation combinatoire. Pour plus de détails, les auteurs peuvent consulter le livre de (Aarts et Korst, 1990).

Venugopal et Narendran (1992) ont suggéré une méthode basée sur le recuit simulé et l'ont appliquée sur le problème de la fabrication cellulaire en utilisant la variation totale de la charge cellulaire entre les machines comme fonction objectif. Les résultats obtenus sont comparés avec ceux de l'algorithme K-means.

Chen et Srivastava (1994) ont proposé un modèle de programmation quadratique pour maximiser la somme de similarité des machines à l'intérieur des cellules, en limitant la taille de ces dernières. Deux heuristiques basées sur SA ont été développées pour résoudre le modèle proposé : Algorithme à une seule étape, SA1, donne de bons résultats mais avec un temps de calcul élevé ; Algorithme à deux étapes, SA2, demande moins de temps avec une qualité de solution équivalente à celle obtenue par SA1.

Vakharia et Chang (1997) ont proposé deux approches de recherche combinatoire pour le CFP basées sur le recuit simulé (SAHCF) et la méthode Tabou (TSHCF). Le but est de minimiser les dépenses totales des machines et le coût de manutention requise pour transférer les pièces entre les cellules. L'étude a indiqué que SAHCF a dépassé TSHCF en matière de qualité de la solution et du temps de calcul.

Ying *et al.* (2011) ont développé une métaheuristique basée sur le recuit simulé avec un voisinage variable pour résoudre le CFP. Trois types de voisinage sont utilisés selon des probabilités : la division d'une cellule, la fusion des cellules existantes, et l'échange d'un produit ou d'une machine entre deux cellules. Les auteurs ont fixé à deux, le nombre de cellules pour la génération d'une solution initiale.

1.4.4.2 Métaheuristiques à population

Les métaheuristiques à population ont soulevé un grand intérêt de la part de nombreuses communautés scientifiques pour la résolution des problèmes d'optimisation. Nous allons passer en revue certaines méthodes utilisées pour la conception des systèmes de production cellulaire.

Joines *et al.* (1996) ont développé un modèle de programmation mathématique en nombres entiers pour le CFP et ils ont appliqué l'algorithme génétique pour résoudre ce modèle. De plus, ils ont introduit une nouvelle représentation de la solution qui inclut les deux partitions (machines et produits) et qui réduit la taille du problème.

Gonçalves et Resende (2004) ont combiné une heuristique de recherche locale avec l'algorithme génétique. Cette heuristique vise à se réitérer l'affectation des machines selon les positions des produits dans les cellules, puis à réassigner les produits selon les nouvelles positions des machines jusqu'à l'obtention d'une solution stable. La méthode suggérée a pu améliorer la qualité des solutions de plusieurs instances de référence.

Muruganandam *et al.* (2005) ont proposé une nouvelle approche basée sur l'algorithme mémétique (MA) pour résoudre le problème de formation de cellules. Cet algorithme est fondé sur l'hybridation entre les algorithmes génétiques et les algorithmes de recherche locale. Le but des auteurs est de minimiser la variation de la charge cellulaire et le nombre total de mouvements. Les résultats ont montré que l'algorithme mémétique surpasse GA et Tabou pour les grandes instances.

Attila Islier (2005) a proposé un algorithme de fourmis artificielles pour optimiser le CFP. Il a utilisé une conception du problème qui considère le passage des fourmis de places mineurs (machine / produit) aux endroits principaux (cellules de fabrication). La méthodologie a été comparée avec des techniques antérieures telles que SA, TS, GA. Les résultats ont montré que ACO ne peut être comparé qu'avec l'algorithme GA.

Andres et Lozano (2006) étaient les premiers à appliquer l'algorithme d'optimisation par essaims particulaires (PSO) à ce type de problèmes, ce qui a nécessité une adaptation appropriée à un ensemble de ses éléments. L'algorithme proposé tente de réduire les mouvements intercellulaires en respectant une taille maximale de chaque cellule.

Ateme-Nguema et Dao (2007) ont introduit un modèle de programmation linéaire en nombres entiers binaires pour le problème de formation de cellules. Ils ont également présenté une hybridation entre l'algorithme de colonies de fourmis (ACO) et la méthode Tabou pour résoudre le modèle proposé.

L'algorithme des abeilles (BA) est proposé par (Pham *et al.*, 2007) afin de réduire les mouvements intracellulaires et intercellulaires en considérant l'efficacité du groupement comme mesure de performance. Les solutions initiales ont été générées aléatoirement avec un certain nombre d'abeilles scoutes, puis une phase de recherche est appliquée pour les améliorer. Les tests ont indiqué que cet algorithme peut être efficace pour la phase de conception du système de production cellulaire.

Ming et Ponnambalam (2008) ont proposé une approche hybride combinant le PSO avec GA pour former les cellules de production. De plus, ils ont utilisé trois mesures pour évaluer la performance de la méthode proposée : la minimisation des mouvements intercellulaires et intracellulaires, la réduction de la variation totale de la charge cellulaire, et la minimisation des mouvements intercellulaires des familles de produits.

Spiliopoulos et Sofianopoulou (2008) ont appliqué une méthode basée sur ACO pour optimiser le système de production cellulaire en tenant compte de la taille maximale des cellules qui est initialement égale au nombre de machines dans chaque cellule. Les auteurs ont associé une valeur propre jouant le rôle de la limite pour évaluer correctement les solutions et accélérer la recherche.

Mahdavi *et al.* (2009) ont introduit un modèle de programmation mathématique pour la formation des cellules. L'objectif du modèle est de minimiser le nombre des éléments exceptionnels et des éléments vides afin d'obtenir une meilleure utilisation des cellules. Les auteurs ont développé une méthode heuristique basée sur l'algorithme génétique pour résoudre le modèle proposé.

Anvari *et al.* (2010) ont proposé une version modifiée du PSO pour résoudre le CFP. Ils ont appliqué une méthode de diversification en tant que générateur d'une population initiale. Également, les auteurs ont utilisé l'opérateur de mutation pour diversifier et accélérer le processus de recherche

de solutions et s'échapper de l'optimum local. Comme un mécanisme d'intensification, les auteurs ont appliqué une méthode de recherche locale.

Pailla *et al.* (2010) ont développé deux approches pour former les cellules de production. La première approche est un algorithme évolutif qui améliore l'efficacité de l'algorithme génétique par l'application d'une recherche locale sur certaines solutions. Tandis que la deuxième approche est basée sur le recuit simulé et utilise une même représentation de la solution. Les résultats obtenus par les deux approches étaient prometteurs.

Également, l'algorithme ACO est proposé par (Li *et al.*, 2010) en implémentant le système max-min dans le framework Hyper-cube. Les auteurs ont intégré un mécanisme de recherche locale comme une stratégie d'intensification. En fonction de l'affectation des machines (ou des produits), ce mécanisme peut réattribuer de manière efficace les produits (ou des machines) aux cellules.

Wu *et al.* (2010) ont introduit l'algorithme d'écoulement d'eau (WFA) qui traite la taille dynamique des solutions et surmonte les inconvénients des techniques basées sur une seule ou plusieurs solutions. Une méthode d'affectation des produits et des machines est utilisée pour générer les solutions initiales, puis des opérations de division et de déplacement sont appliquées pour obtenir des meilleures solutions.

Arkat *et al.* (2011) ont présenté un modèle mathématique bi-objectif pour minimiser simultanément le nombre des éléments exceptionnels et vides. Ils ont introduit deux méthodes pour résoudre le modèle : la méthode exacte e-contrainte (*e-constraint method*) et l'algorithme génétique multi-objectifs. Les résultats obtenus ont montré la capacité de GA à résoudre des instances de grande taille dans un temps raisonnable.

Elbenani *et al.* (2012) ont combiné l'algorithme génétique avec une méthode de recherche locale (GA_LNS). Cette méthode applique séquentiellement une stratégie d'intensification pour améliorer localement une solution courante et une stratégie de diversification qui détruit cette solution courante pour en récupérer une nouvelle solution. La méthode proposée a pu améliorer les solutions les plus connues de certaines instances de référence.

Díaz *et al.* (2012) ont utilisé la procédure de recherche gloutonne aléatoire adaptative (GRASP : Greedy Randomized Adaptive Search Procedure) pour former les cellules de production. La méthode suggérée se compose de deux phases : premièrement l'affectation initiale des machines aux cellules (ou les produits aux familles de produits). Ensuite, l'attribution des produits (ou les machines) en intégrant une heuristique de recherche locale pour améliorer l'efficacité de la solution.

Sayadi *et al.* (2013) ont présenté une version discrète de l'algorithme des lucioles (FA) pour résoudre des problèmes d'optimisation discrète et ils ont évalué l'efficacité de cette version en l'appliquant sur six instances du problème de formation de cellules. De plus, les auteurs ont utilisé trois critères pour évaluer l'approche proposée : le pourcentage des éléments exceptionnels, l'utilisation des machines et l'efficience de groupement.

Zeb *et al.* (2016) ont présenté une méthode hybride basée sur la mesure d'efficacité de groupement. Cette méthode combine la technique d'exploration de l'algorithme génétique avec la puissance d'intensification du recuit simulé. Si la même solution est répétée pendant un certain nombre d'itérations, une certaine perturbation (shake-off) est utilisé dans le recuit simulé pour s'échapper du maximum local. Les tests ont montré que l'utilisation du shake-off améliore beaucoup les résultats.

Zettam et Elbenani (2016) étaient les premiers à appliquer l'algorithme de recherche gravitationnelle discrète (DGSA) au problème avec une adaptation appropriée de ses éléments. Le but est de minimiser le nombre des éléments exceptionnels et des éléments vides. Les auteurs ont introduit des opérateurs de mutation et de croisement pour intensifier la recherche de la meilleure solution.

Kamalakannan *et al.* (2017) ont proposé une méthode fondée sur ACO pour former des cellules et ils ont utilisé l'efficacité de groupement comme mesure de performance. De plus, une procédure de recherche locale basée sur l'index de la machine est intégrée afin d'améliorer la solution.

1.4.5 Méthodes de l'intelligence artificielle

La réalisation d'un système de production comporte souvent des incertitudes et des ambiguïtés. Un certain nombre de chercheurs ont appliqué des méthodes basées sur le clustering ou le regroupement flou, la programmation mathématique floue et les réseaux de neurones flous pour résoudre le CFP.

1.4.5.1 Réseaux de neurones

Les réseaux de neurones (Neural Networks : NNs) ont été largement utilisés dans la résolution du CFP grâce à leur nature robuste et adaptative. Différents types de NNs ont été employés avec succès dont les plus connus sont : le réseau de Hopfield, la carte auto adaptative (SOM), et la théorie de résonance adaptative (ART).

Zolfaghari et Liang (1997) ont proposé une nouvelle structure du réseau de neurones de Hopfield, (OSHN). Cette structure a été conçue en conjonction avec un schéma de recherche guidée par des objectifs. L'approche proposée n'exige pas de processus de formation et n'est pas affectée par une disposition initiale des machines.

Soleymanpour *et al.* (2002) ont abordé un certain nombre d'inconvénients des approches des réseaux neuronaux développées pour le CFP et ils ont proposé un algorithme de réseau neuronal transitoirement chaotique (TCNN) avec des procédures supplémentaires pour surmonter quelques lacunes. L'algorithme proposé a été testé sur plusieurs instances, et comparé avec diverses approches de la littérature. Les résultats obtenus ont montré la supériorité de TCNN dans le regroupement des produits et des machines.

Guerrero *et al.* (2002) ont proposé une stratégie en deux phases pour regrouper les produits et les machines. La première phase consiste à modéliser le problème de groupement des produits comme un problème de programmation quadratique où les coefficients de similarité pondérés ont été calculés et les produits ont été regroupés en utilisant un nouveau réseau neuronal auto-adaptatif (SONN). Dans la deuxième phase, un modèle linéaire de flux de réseau a été adapté pour assigner les machines aux cellules.

Solimanpur *et al.* (2004b) ont étudié la sensibilité, la faisabilité et la robustesse du réseau neuronal transitoirement chaotique (TCNN) pour former les cellules de production. TCNN présente les avantages du réseau neural chaotique et du réseau neuronal de Hopfield. Les auteurs ont démontré l'effet de la dynamique du réseau sur la taille des problèmes de formation des cellules et la qualité des solutions obtenues.

Venkumar et Haq (2005) ont proposé une version modifiée de ART connue sous le nom ART1 où le paramètre d'entrée est la matrice d'incidence binaire. La méthode suggérée génère des solutions sous forme de listes contenant des familles de produits, des cellules de machines et un nombre d'éléments exceptionnels. Le but est de réduire le nombre des mouvements intercellulaires. En 2006, les mêmes auteurs (Venkumar et Haq, 2006) ont présenté un autre algorithme basé sur le réseau de neurones de cartes auto-adaptatives de Kohonen. La performance de la méthode a été mesurée par le nombre d'éléments exceptionnels et l'efficacité de groupement.

Ateme-Nguema et Dao (2009) ont présenté une version modifiée du réseau de neurones de Hopfield pour former des cellules de production. Afin d'améliorer la qualité des solutions et résoudre des instances de grande taille, les auteurs ont reformulé le problème comme un modèle de programmation linéaire et ils ont combiné l'approche proposée avec la méthode Tabou.

1.4.5.2 Logique floue

Masnata et Settineri (1997) ont adapté un algorithme de regroupement flou (*fuzzy c-means clustering*) pour développer une approche non-binaire pour la Technologie de groupe. Cette approche est basée sur les fondements de la logique floue. Également, les auteurs ont intégré l'algorithme de regroupement flou dans une stratégie d'ordonnancement minimum et ils l'ont évalué par deux études de cas.

Susanto *et al.* (1999) ont proposé une nouvelle approche de regroupement flou basée sur la formation séparée des clusters de produits et des clusters de machines. En outre, ils ont adopté une technique d'affectation dans la formation des cellules de fabrication.

Ravichandran et Rao (2001) ont présenté un nouveau coefficient de similarité et ils ont développé un modèle mathématique qui utilise ce coefficient pour résoudre de façon presque optimale le CFP. De plus, les auteurs ont introduit un nouvel algorithme de regroupement flou et ils ont comparé les résultats avec ceux d'autres algorithmes y compris fuzzy c-means clustering.

Lozano *et al.* (2002) ont proposé une version plus performante du regroupement flou (fuzzy c-means clustering algorithm) qui consiste à grouper à la fois les produits et les machines de façon simultané et ils ont rendu la méthode capable d'estimer le nombre de cellules à former. Les auteurs ont pris en considération l'influence des liaisons existantes entre les différents produits et machines inclus dans des solutions.

Park et Suresh (2003) ont appliqué une approche floue améliorée du réseau neural ART et ils l'ont comparé aux procédures classiques de regroupement hiérarchique. Ils ont aussi développé de nouveaux schémas de représentation du problème et des mesures de performance de regroupement afin d'améliorer la qualité des solutions.

Torkul *et al.* (2006) ont utilisé la logique floue pour former simultanément les familles de produits et les cellules de machines. Leur objectif principal était de comparer la conception des cellules de production à l'aide de l'algorithme de regroupement flou (*fuzzy c-means*) avec les méthodes crisp (*crisp methods*). Les résultats obtenus ont montré la supériorité des solutions de regroupement flou sur un ensemble d'instances de référence.

1.5 Formulation mathématique

Pour formuler ce problème mathématiquement, nous avons adopté le modèle introduit par (Elbenani et Ferland, 2012). Le modèle mathématique adapté est basé sur la mesure de performance : efficacité de groupement d'une solution s (*Eff(s)*). Dans la littérature, cette mesure est la plus utilisée par les auteurs pour comparer l'efficacité de leurs méthodes.

$$Eff(s) = \frac{a_1 - a_1^{out}}{a_1 + a_0^{in}} = 1 - \left(\frac{a_0^{in} + a_1^{out}}{a_1 + a_0^{in}} \right) \tag{1.5}$$

Avec :

M	: Nombre de machines ;
P	: Nombre de produits ;
C	: Nombre de cellules ;
$a_1 = \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij}$	: Nombre total de 1 dans la matrice d'incidence ;
a_1^{out}	: Nombre total des 1 en dehors des blocs diagonaux d'une solution s, appelés éléments exceptionnels ;
a_0^{in}	: Nombre total des 0 dans des blocs diagonaux d'une solution s, appelés éléments vides ;

Maximiser $\mathit{Eff}(s)$ est donc équivalent à minimiser $\mathit{ComEff}(s)$:

$$\mathit{ComEff}(s) = \left(\frac{a_0^{in} + a_1^{out}}{a_1 + a_0^{in}} \right) \tag{1.6}$$

Pour spécifier une formulation mathématique du CFP, deux variables binaires sont introduites :

Pour chaque paire $i = 1, \ldots, M \, ; \, k = 1, \ldots, C$

$$x_{ik} = \begin{cases} 1, & \text{si machine } i \text{ appartient à la cellule } k \\ 0, & \text{sinon} \end{cases}$$

Pour chaque paire $j = 1, \ldots, P \, ; \, k = 1, \ldots, C$

$$y_{jk} = \begin{cases} 1, & \text{si produit } j \text{ appartient à la cellule } k \\ 0, & \text{sinon} \end{cases}$$

La détermination de X et Y permet de déterminer une solution s. Dans ce cas, les éléments exceptionnels et les éléments vides peuvent être présentés selon ces deux variables :

$$a_1^{out} = a_1 - \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} x_{ik} y_{jk}$$
$$a_0^{in} = \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} (1 - a_{ij}) x_{ik} y_{jk}$$

La fonction objectif peut être écrite comme :

$$\begin{aligned} \mathit{ComEff}(s) = \mathit{ComEff}(X, Y) &= \left(\frac{a_0^{in} + a_1^{out}}{a_1 + a_0^{in}} \right) = \\ &= \frac{\sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} (1 - a_{ij}) x_{ik} y_{jk} + a_1 - \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} x_{ik} y_{jk}}{a_1 + \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} (1 - a_{ij}) x_{ik} y_{jk}} \\ &= \frac{a_1 + \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} (1 - 2a_{ij}) x_{ik} y_{jk}}{a_1 + \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} (1 - a_{ij}) x_{ik} y_{jk}} \end{aligned} \tag{1.7}$$

Le problème de formation de cellules peut être formulé comme dans le modèle $M_1(X, Y)$ suivant :

$$M_1(X, Y) : \quad \min \mathit{ComEff}(X, Y) = \frac{a_1 + \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} (1 - 2a_{ij}) x_{ik} y_{jk}}{a_1 + \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} (1 - a_{ij}) x_{ik} y_{jk}} \tag{1.8}$$

Sujet à :

$$\sum_{k=1}^{C} x_{ik} = 1, \quad \forall i = 1, ..., M \tag{1.9}$$

$$\sum_{k=1}^{C} y_{jk} = 1, \quad \forall j = 1, ..., P \tag{1.10}$$

$$\sum_{i=1}^{M} x_{ik} \geq 1, \quad \forall k = 1, ..., C \tag{1.11}$$

$$\sum_{j=1}^{P} y_{jk} \geq 1, \quad \forall k = 1, ..., C \tag{1.12}$$

$$x_{ik} \in \{0, 1\}, \quad \forall i = 1, ..., M; \forall k = 1, ..., C \tag{1.13}$$

$$y_{jk} \in \{0, 1\}, \quad \forall j = 1, ..., P; \forall k = 1, ..., C \tag{1.14}$$

La contrainte (1.9) assure que chaque machine i est affectée à une et une seule cellule. La contrainte (1.10) garantit que chaque produit j doit être affecté à une et une seule cellule. Les inégalités (1.11) et (1.12) garantissent que chaque cellule k comprend au moins une machine et un produit. Enfin, les contraintes (1.13) et (1.14) indiquent que les variables de décision sont binaires.

1.6 Conclusion

La formation de cellules de fabrication est une étape clé dans la mise en œuvre de la Technologie de groupe. Dans ce chapitre, nous avons parcouru la littérature pour présenter les différents types d'organisation de production. Puis, nous avons présenté le problème de formation de cellules et les différentes méthodes proposées pour le résoudre. Dans ce contexte, nous constatons que la plupart des auteurs ont utilisé les métaheuristiques, vu le rôle important qu'elles jouent dans la résolution des problèmes d'optimisation combinatoire. Le but de notre thèse est de proposer de nouveaux algorithmes inspirés de la nature pour résoudre le problème et améliorer la qualité des solutions obtenues dans un temps de calcul raisonnable.
Dans le chapitre suivant, nous allons présenter les méthodes basées sur l'algorithme de sélection clonale que nous avons développé pour former les cellules de production où le nombre de ces cellules est fixé à l'avance.

Chapitre

2 CFP avec cellules fixes : Algorithme de sélection clonale

Sommaire

2.1 Introduction

L'objectif de ce second chapitre est de présenter les méthodes inspirées de l'algorithme de sélection clonale que nous avons proposé pour résoudre le problème de formation de cellules. Nous allons commencer par définir le système immunitaire naturel et ses principales caractéristiques. Puis, nous présentons les algorithmes inspirés du système immunitaire naturel et leurs applications. Ensuite, nous détaillons nos méthodes proposées. Nous présentons l'adaptation de chaque méthode au problème ainsi que son paramétrage. Les résultats d'évaluation de ces méthodes sur plusieurs instances de tests sont rapportés dans ce chapitre. De plus, les méthodes sont comparées à d'autres méthodes collectées de la littérature pour valider leurs performances.

2.2 Système immunitaire

2.2.1 Système immunitaire naturel

Le système immunitaire naturel est un système biologique constitué d'un ensemble de mécanismes de défenses de l'organisme capable de distinguer le soi du non-soi. Il existe deux types de mécanismes : inné et acquis.

— Les mécanismes de défense naturelle, non-spécifique ou innée constituent la première ligne de défense face à une contamination. Ils reposent sur une distinction globale non adaptative du soi et du non-soi. La protection de la peau, l'acidité gastrique, les cellules phagocytaires, et les larmes sont parmi les mécanismes de défense naturelle.

— Les mécanismes de défense adaptative, spécifique ou acquise utilisent deux types de cellules spécialisées appelées lymphocytes B et lymphocytes T. Les lymphocytes B produisent des anticorps spécifiques dirigés contre un agent infectieux. Les lymphocytes T sont produits dans le thymus et ils visent à détruire directement les particules étrangères.

Les principales caractéristiques du système immunitaire sont :

— La spécificité ou l'identification : Le système immunitaire a la capacité de reconnaître et d'éliminer certains agents étrangers (antigènes). Chaque antigène contient une structure moléculaire particulière qui déclenche la production des anticorps spécifiques orientés contre lui.

— La diversité : le système est capable de produire des cellules immunisées contre des millions de types d'antigènes envahissants le corps. Durant cette production, les cellules sont soumises à un processus de mutation somatique qui permet la création des nouveaux modèles de récepteurs, et par conséquent augmenter la diversité des récepteurs immunisés.

— L'étude : le processus d'hypermutation somatique et le mécanisme de la sélection permettent au système de raffiner sa réponse immunitaire à un antigène. Ce processus est connu par la maturation d'affinité. Elle assure l'augmentation de l'efficacité de la réponse immunitaire par identification des modèles d'antigènes de façon plus rapide.

— La mémoire : le système immunitaire se caractérise par sa capacité de mémoriser les antigènes qu'il a rencontrés. Par conséquent, il fournit des immuno-réactions plus puissantes et plus rapides lors d'expositions ultérieures par le même antigène ou par des antigènes semblables.

2.2.2 Système immunitaire artificiel

Un système immunitaire artificiel (AIS) est un système adaptatif inspiré du fonctionnement du système immunitaire naturel et qui vise à exploiter ses caractéristiques importantes telles que la mémorisation, l'apprentissage, la reconnaissance de formes, et les capacités d'adaptation.

Ce système est appliqué dans plusieurs domaines comme l'apprentissage par machines, la reconnaissance et la classification des modèles, la sécurité informatique, la détection d'anomalie, l'optimisation, et diverses autres applications dans le domaine de la science et de l'ingénierie.

Les caractéristiques du système immunitaire ont poussé plusieurs chercheurs à développer des algorithmes d'où les plus connus sont : l'algorithme de la sélection négative, l'algorithme du réseau immunitaire et l'algorithme de la sélection clonale.

2.2.3 Les algorithmes d'AIS

2.2.3.1 Algorithme de la sélection négative

L'algorithme de la sélection négative est inspiré par le mécanisme principal du thymus qui produit un ensemble de cellules T matures capables de lier uniquement des antigènes du non soi. Il a été proposé, pour la première fois, par (Forrest *et al.*, 1994) pour détecter la manipulation de données causées par un virus dans un système informatique. L'algorithme commence par la production d'un ensemble S qui définit l'état normal du système à protéger. Ensuite, un ensemble de détecteurs D est généré et qui sera responsable d'identifier tous les éléments qui n'appartiennent pas à S, c'est-à-dire les éléments non conformes. Les étapes de l'algorithme de sélection négative sont décrites par l'algorithme 2.1.

Algorithme 2.1 Algorithme de la sélection négative

Tant que critère d'arrêt est non satisfait **Faire** :

1. Générer un ensemble S qui définit l'état normal du système à protéger ;
2. Générer des détecteurs potentiels de manière aléatoire et les placer dans un ensemble P ;
3. Déterminer l'affinité de chaque membre de P avec chaque membre de S ;
4. **Si** un élément de S reconnaît un détecteur de P selon un seuil de reconnaissance **alors**
   le détecteur sera rejeté
5. **Sinon**
   le détecteur sera ajouté à l'ensemble des détecteurs disponibles D.
6. **Finsi**

Fin Tant que

La sélection négative propose une approche alternative pour effectuer la reconnaissance de formes. Le but de cette reconnaissance est de trouver les relations et les modèles utiles à partir de l'information.

L'algorithme cible les problèmes de détection d'anomalie tels que la détection d'intrusion informatique et du réseau. De plus, il est utilisé dans la prévision des séries chronologiques, l'inspection et la segmentation des images et la tolérance aux pannes matérielles (Hofmeyr et Forrest, 2000).

2.2.3.2 Algorithme du réseau immunitaire

En 1974, Jerne (1974) a proposé une théorie des réseaux immunitaires pour aider à expliquer certaines propriétés émergentes observées du système immunitaire, telles que l'apprentissage et la mémoire. Le principe de cette théorie est que tout récepteur de lymphocytes au sein d'un organisme peut être reconnu par un sous-ensemble de récepteurs ayant leur propre ensemble de reconnaissance.

La théorie des réseaux immunitaires paraît attrayante pour n'importe quel chercheur d'intelligence artificielle. Elle constitue un système dynamique capable d'effectuer des interactions entre ses propres composants ou entre ses composants et les antigènes étrangers. De plus, elle permet de régler la structure du système et ajuster ses paramètres par rapport à l'environnement. Son algorithme général est présenté dans l'algorithme 2.2.

Algorithme 2.2 Algorithme du réseau immunitaire

Etape 1 : Créer un premier ensemble aléatoire d'anticorps de réseau ;

Etape 2 : **Tant que** critère d'arrêt est non satisfait **Faire** :

1. Evaluer l'affinité entre les anticorps et l'antigène ;
2. Calculer l'affinité entre chaque paire d'anticorps ;
3. Ajouter de nouveaux anticorps au réseau et supprimer les anticorps inutiles selon un certain critère ;
4. Déterminer le degré de stimulation de chaque cellule du réseau en tenant compte des résultats obtenus par les étapes précédentes ;
5. Mettre à jour la structure du réseau selon le degré de stimulation des cellules ;

Fin Tant que

2.2.3.3 Algorithme de la sélection clonale

Quand une liaison entre les anticorps d'une cellule B et un antigène se produit, la cellule B devient active et commence à proliférer en copies exactes. Ces copies sont soumises à un mécanisme d'hypermutation pour créer des anticorps spécifiques à l'antigène pénétrant. Les propriétés de base de ce type de réponse immunitaire sont définies par le terme de sélection clonale. Son principe réside dans la détermination des cellules capables d'identifier le stimulus antigénique et qui ont les plus grands degrés d'affinité.

Il existe différentes critiques à propos de la sélection clonale dans la littérature. Certains chercheurs (Hofmeyr et Forrest, 2000) estiment que la sélection clonale est un algorithme génétique avec un opérateur de croisement ; Tandis que d'autres chercheurs (De Castro et Von Zuben, 2000) montrent que l'algorithme génétique ne peut pas couvrir tous les aspects de la sélection clonale, comme la reproduction proportionnelle à l'affinité et l'hypermutation et ils ont proposé un algorithme de sélection clonale, nommé CLONALG, qui utilise les processus de base impliqués dans la sélection clonale (De Castro et Von Zuben, 2002). Les étapes de base de l'algorithme de sélection clonale sont rapportées dans l'algorithme 2.3.

Algorithme 2.3 Algorithme de la sélection clonale

Etape 1 : Initialiser une population de n anticorps définis selon le problème étudié ;

Etape 2 : **Tant que** critère d'arrêt est non satisfait **Faire** :

1. Choisir n_1 anticorps qui ont la plus grande affinité à l'antigène ;
2. Produire des copies exactes (clones) aux anticorps choisis : le nombre de copies est proportionnel aux affinités (plus l'affinité est élevée, plus le nombre de clones est grand) ;
3. Changer la structure des anticorps choisis (hypermutation) : le taux de changement dépend de l'affinité ;
4. Ajouter les anticorps mutés à la population et remplacer n_2 anticorps de faible affinité avec de nouveaux anticorps (receptor editing) ;

Fin Tant que

Ce type d'algorithmes peut être appliqué pour la résolution des problèmes de clustering, d'optimisation, ou de reconnaissance des formes.

2.2.4 Applications d'AIS

Les algorithmes du système immunitaire artificiel ont été largement testés sur diverses applications grâce à la polyvalence et la puissance du système immunitaire biologique duquel ils s'inspirent. Par la suite, nous citons quelques exemples d'applications de ces algorithmes dans différents domaines comme la reconnaissance et la classification des modèles, la sécurité informatique, la détection d'anomalie et l'optimisation, etc.

Watkins et Timmis (2004) ont présenté les premières étapes de la réalisation d'un système immunitaire artificiel parallèle pour la classification en bénéficiant des caractéristiques de reconnaissance et d'apprentissage des AIS. Ils ont développé une version simple et parallèle de l'algorithme de classification intitulée *Artificial Immune Recognition System : AIRS*.

Engin et Döyen (2004) ont proposé une nouvelle approche pour résoudre le problème d'ordonnancement du flow-shop en utilisant l'algorithme de sélection clonale. Ce problème est un problème NP difficile. L'objectif des auteurs est de minimiser le temps de production en respectant les contraintes du problème. Plusieurs autres chercheurs ont introduit des approches basées sur AIS pour résoudre le même problème tels que (Zandieh *et al.*, 2006) et (Chung et Liao, 2013).

En profitant de la relation entre la concentration d'anticorps et l'intensité de l'intrusion pathogène, Li (2005) a présenté un modèle capable d'évaluer les types et l'intensité des attaques réseaux, et ainsi calculer le niveau de risque de la sécurité du réseau en temps réel.

Zhong *et al.* (2006) ont développé un nouvel algorithme d'apprentissage automatique non supervisé pour effectuer la classification des images de télédétection. Les auteurs ont combiné les propriétés de classification non linéaires avec les propriétés biologiques de AIS telles que la sélection clonale, le réseau immunitaire et la mémoire immunitaire.

Gao *et al.* (2009) ont développé une approche combinant l'algorithme de sélection clonale et une méthode d'évolution différentielle. Cette approche est évaluée par l'optimisation des fonctions non linéaires et la formation du réseau neuronal en cascade-corrélation.

Liu *et al.* (2012) ont proposé un nouveau algorithme de clustering basé sur l'algorithme de sélection clonale, dans lequel un nouvel opérateur nommé opérateur d'immunodominance d'anticorps est introduit. Cet opérateur permet d'utiliser les connaissances antérieures accumulées dans le processus de recherche des classes et d'améliorer les performances de l'algorithme tout en évitant de se piéger dans des optima locaux.

Diabat *et al.* (2013) ont développé une méthode fondée sur le système immunitaire artificiel pour optimiser le processus de réutilisation des produits déjà utilisés. Leur but est d'augmenter les performances environnementales et les bénéfices qui en résultent ; tout en minimisant le coût total de la logistique, qui consiste en la location, le transport d'inventaire, la manutention, l'installation et les coûts d'expédition.

Kuo *et al.* (2013) ont combiné l'algorithme d'optimisation par essaims particulaires (PSO) et l'algorithme de sélection clonale pour optimiser les paramètres des machines à vecteurs de support (SVM). La méthode résultante est appliquée dans l'identification de la fréquence du radio.

Lima *et al.* (2015) ont introduit une nouvelle méthode pour détecter et classer les perturbations de tension dans les systèmes de distribution électrique. Cette méthode combine l'algorithme de sélection négative avec le concept discret de transformation d'ondelettes. L'algorithme de sélection négative est responsable de la réalisation des diagnostiques, puis l'identification et la classification des anomalies. Le but des auteurs est l'amélioration du fonctionnement du système durant l'apparition des erreurs.

Schmidt *et al.* (2017) ont présenté une nouvelle approche basée sur l'algorithme de sélection positive, qui est similaire à l'algorithme de sélection négative, pour résoudre le problème de classification des flux de trafic dans les réseaux. Ils ont également utilisé plusieurs techniques d'optimisation dans les phases de formation et de classification pour améliorer la qualité des solutions obtenues.

Le système immunitaire artificiel est un domaine de recherche très intéressant qui se distingue des autres modèles inspirés de la biologie. Il regroupe quelques caractéristiques des algorithmes génétiques et des réseaux de neurones ce qui le rend plus flexible et applicable à plusieurs domaines technologiques.

La difficulté d'utilisation des AIS réside dans le choix de l'algorithme à partir duquel on peut nous inspirer pour résoudre le problème et la spécification de sa représentation biologique, c'est-à-dire trouver les données présentant les anticorps et celles jouant le rôle des antigènes.

Durant notre travail, nous avons proposé deux approches basées sur l'algorithme de sélection clonale pour résoudre le CFP. Cet algorithme est plus adéquat à notre problème et il répond à nos attentes. Par la suite, nous présentons une description détaillée des méthodes proposées.

2.3 Première méthode de résolution : HClonalSA1

La première méthode consiste à combiner l'algorithme de sélection clonale avec la méthode de recherche Tabou dans le but de minimiser les éléments exceptionnels et les éléments vides (Karoum *et al.*, 2016a). Dans cette thèse, cette méthode sera notée par **HClonalSA1** : *the first Hybrid Clonal Selection Algorithm*. Par la suite, nous allons montrer les changements apportés par HClonalSA1 pour l'adapter au problème. Les tests d'évaluation de la méthode proposée sont réalisés sur 30 instances de référence. Les résultats obtenus sont comparés avec les résultats de plusieurs algorithmes collectés de la littérature.

Le problème de formation de cellules est considéré dans ce cas comme étant l'antigène. Les solutions sont codées comme des anticorps et elles sont évaluées par une fonction d'affinité.

2.3.1 Adaptation de HClonalSA1 au CFP

Pour adapter HClonalSA1 à notre problème, un nombre d'éléments doivent être spécifiés : la représentation d'une solution, la génération d'une solution et d'une population initiale, les équations de la sélection clonale, l'opérateur du croisement appliqué et la méthode de recherche locale utilisée.

2.3.1.1 Représentation d'une solution

Chaque solution s est codée sous forme d'un vecteur de longueur P+M : $s = (P_1, \ldots, P_P | M_1, \ldots, M_M)$ Où P_j est l'indice de la famille contenant le produit j, $j = 1, \ldots, P$ et M_i est l'indice du groupe comprenant la machine i, $i = 1, \ldots, M$. Chaque cellule de production contient une famille de produits et un groupe de machines.

Par exemple, la matrice illustrée dans la Figure 2.1 sera présentée par une solution s :
$s =$ (1 1 2 4 3 1 2 3 1 3 4 | 2 1 1 3 2 4 3)
Les produits (P_1, P_2, P_6, P_9) et les machines (M_2, M_3) sont affectés à la cellule 1 :
$s =$ (**1 1** 2 4 3 **1** 2 3 **1** 3 4 | 2 **1 1** 3 2 4 3)
Les produits (P_3, P_7) et les machines (M_1, M_5) sont assignés à la cellule 2 :
$s =$ (1 1 **2** 4 3 1 **2** 3 1 3 4 | **2** 1 1 3 **2** 4 3)
Les produits (P_5, P_8, P_{10}) et les machines (M_4, M_7) sont dans la cellule 3 :

$s = (1\ 1\ 2\ 4\ \mathbf{3}\ 1\ 2\ \mathbf{3}\ 1\ \mathbf{3}\ 4 \mid 2\ 1\ 1\ \mathbf{3}\ 2\ 4\ \mathbf{3}\ )$
Les produits (P_4, P_{11}) et la machine (M_6) sont affectés à la cellule 4 :
$s = (1\ 1\ 2\ \mathbf{4}\ 3\ 1\ 2\ 3\ 1\ 3\ \mathbf{4} \mid 2\ 1\ 1\ 3\ 2\ \mathbf{4}\ 3\ )$
Cette solution est dite réalisable car chaque cellule contient au moins un produit et une machine.

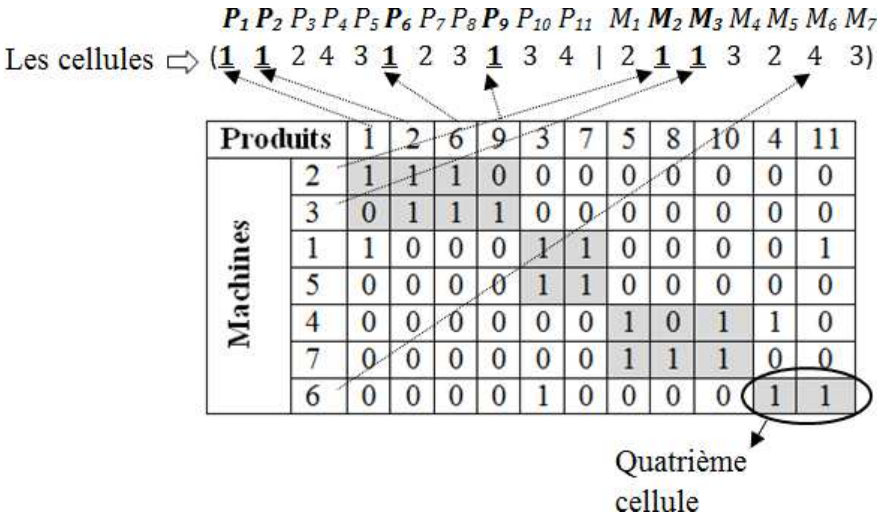


FIGURE 2.1 – HClonalSA1 : Exemple de représentation de la solution

2.3.1.2 Génération d'une solution

Les solutions de la population initiale sont générées aléatoirement. Chaque machine i et produit j est attribué de manière aléatoire à une cellule k. Soit $s0$ une solution générée aléatoirement pour une instance du problème de 11 produits, 7 machines et 4 cellules :
$s0 = (1\ 1\ 2\ 4\ 3\ 1\ 2\ 3\ 1\ 3\ 4 \mid 2\ 1\ 1\ 3\ 2\ 3\ 3\ )$
Cette solution est non réalisable du fait que la quatrième cellule ne contient aucune machine. Dans ce cas, le processus de réparation déplace une machine de la cellule contenant le plus grand nombre de machines (la troisième cellule) vers la quatrième cellule et qui entraine une diminution minimale de l'efficacité de la solution. La solution ainsi obtenue après le processus de réparation est :
$s1 = (1\ 1\ 2\ 4\ 3\ 1\ 2\ 3\ 1\ 3\ 4 \mid 2\ 1\ 1\ 3\ 2\ \mathbf{4}\ 3\ )$

Les étapes nécessaires pour générer une solution sont présentées dans l'algorithme 2.4. Pour corriger les solutions non réalisables qui peuvent résulter à cause d'une cellule vide (cellule sans machines ou produits), un processus de réparation est appliqué pour introduire l'élément manquant dans cette cellule (voir l'algorithme 2.5).

2.3.1.3 Population initiale

La génération de la population initiale est un processus répétitif permettant de créer des solutions en suivant les étapes indiquées dans l'algorithme 2.4. La population est représentée sous forme de liste de solutions. Cette liste est de taille fixe (*popsize*) déterminée après une étude approfondie sur les paramètres. Chaque solution est générée de manière aléatoire et ajoutée à la population si elle n'y figure pas. Il est à souligner que la génération aléatoire des solutions initiales permet de doter la méthode d'un aspect diversifiant dans l'espace de recherche sans besoin de recourir à un mécanisme de diversification.

Algorithme 2.4 HClonalSA1 : Génération d'une solution initiale

Etape 1 : Pour chaque produit j ($j = 1, \ldots, P$) **Faire** :

1. Générer aléatoirement un nombre k entre 1 et C ;
2. Affecter le produit j à la cellule k ;
3. Augmenter le compteur ($Cmpt\_produits_k$) du nombre de produits affectés à la cellule k.

Fin pour

Etape 2 : Pour chaque machine i ($i = 1, \ldots, M$) **Faire** :

1. Générer aléatoirement un nombre k entre 1 et C ;
2. Assigner la machine i à la cellule k ;
3. Augmenter le compteur ($Cmpt\_machines_k$) du nombre de machines affectées à la cellule k.

Fin pour

Algorithme 2.5 HClonalSA1 : Processus de réparation de la solution

Pour chaque cellule k ($k = 1, \ldots, C$) **Faire** :

1. Si $Cmpt\_produits_k == 0$ (c-à-d aucun produit n'est affecté à la cellule k) alors
   a) Déterminer une cellule l avec un plus grand nombre de produits ;
   b) Déplacer le produit, qui améliore le plus l'efficacité de la solution, de la cellule l vers la cellule k.
2. Si $Cmpt\_machines_k == 0$ (c-à-d aucune machine n'est affectée à la cellule k) alors
   a) Déterminer une cellule l avec un plus grand nombre de machines ;
   b) Déplacer la machine, qui améliore le mieux l'efficacité de la solution, de la cellule l vers la cellule k.

Fin pour

2.3.1.4 Sélection clonale

La valeur d'affinité est utilisée comme mécanisme de sélection des solutions à cloner. Dans notre méthode, l'affinité d'une solution s est basée sur son efficacité de groupement telle que représentée dans l'Equation (2.1).

$$Affinite(s) = \frac{1}{ComEff\,(s)} \tag{2.1}$$

Puisque le but de la fonction objectif est de minimiser *ComEff* (Chapitre 1, Equation (1.8)) donc les solutions ayant la plus grande affinité dans la population seront sélectionnées pour proliférer en plusieurs clones ou copies.

Le clonage est la procédure de création de copies exactes de la solution. Le nombre de clones d'une solution s est calculé en fonction de sa valeur d'affinité, la taille de la population (*popsize*) et la somme des affinités de la population entière (Equation (2.2)).

$$Nombre\ de\ clones = \frac{Affinite(s)}{\sum_{t=1}^{popsize} Affinite(t)} \times popsize \tag{2.2}$$

2.3.1.5 Croisement uniforme

L'opérateur de croisement combine l'information de deux parents de sorte que les deux enfants peuvent avoir une "ressemblance" avec chaque parent. Après des tests préliminaires sur les différents types de croisement, on a choisit d'appliquer le croisement uniforme sur l'ensemble des clones générés. Le choix des deux parents est effectué au hasard.

Considérons deux solutions parents :

$SP1 = (SP_1^1, ..., SP_P^1, SP_1^1, ..., SP_M^1)$
$SP2 = (SP_1^2, ..., SP_P^2, SP_1^2, ..., SP_M^2)$

Un vecteur de bits, appelé mask, ayant $(P + M)$ éléments est généré : $(S_1, ..., S_P, S_1, ..., S_M)$.
Les solutions enfants, $SE1 = (SE_1^1, ..., SE_P^1, SE_1^1, ..., SE_M^1)$ et $SE2 = (SE_1^2, ..., SE_P^2, SE_1^2, ..., SE_M^2)$, sont spécifiées comme suit :
Pour $j = 1, ..., P$:

$$\begin{cases} SE_j^1 = SP_j^1 \ et \ SE_j^2 = SP_j^2 & si \ S_j = 1 \\ SE_j^1 = SP_j^2 \ et \ SE_j^2 = SP_j^1 & si \ S_j = 0 \end{cases}$$

Pour $i = 1, ..., M$:

$$\begin{cases} SE_i^1 = SP_i^1 \ et \ SE_i^2 = SP_i^2 & si \ S_i = 1 \\ SE_i^1 = SP_i^2 \ et \ SE_i^2 = SP_i^1 & si \ S_i = 0 \end{cases}$$

La Figure 2.2 présente un exemple d'application du croisement uniforme sur deux solutions parents afin de produire deux nouvelles solutions enfants.

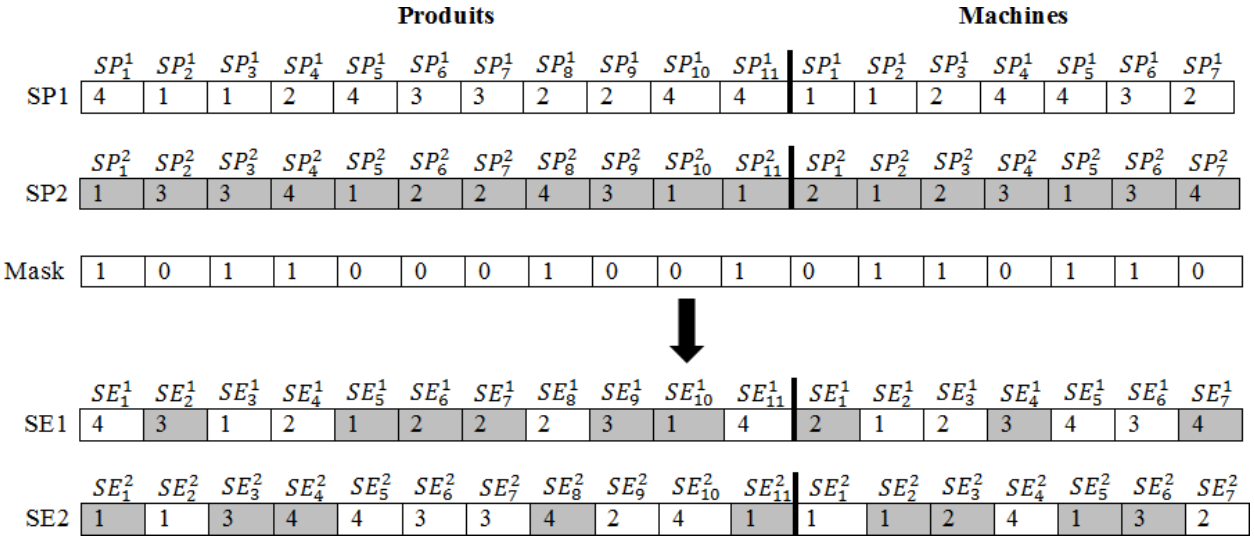


FIGURE 2.2 – HClonalSA1 : Exemple d'application du croisement uniforme

2.3.1.6 Recherche Tabou

Dans la méthode HClonalSA1, nous avons combiné l'algorithme de sélection clonale avec une version améliorée de la méthode de recherche Tabou. Cette dernière est appliquée sur une seule

solution parmi celles résultantes de l'opérateur du croisement. Le but est d'intensifier la recherche, améliorer la qualité des solutions et éviter les optima locaux.

a) Voisinage :

Nous avons proposé deux types de mouvements pour générer les solutions voisines : un mouvement d'insertion et un mouvement d'échange.

- Mouvement d'insertion : déplacer un élément (machine ou produit) de sa cellule actuelle vers une autre cellule choisie au hasard. Par exemple, le cinquième produit dans la Figure 2.3 est déplacé de la cellule 4 vers la cellule 1.

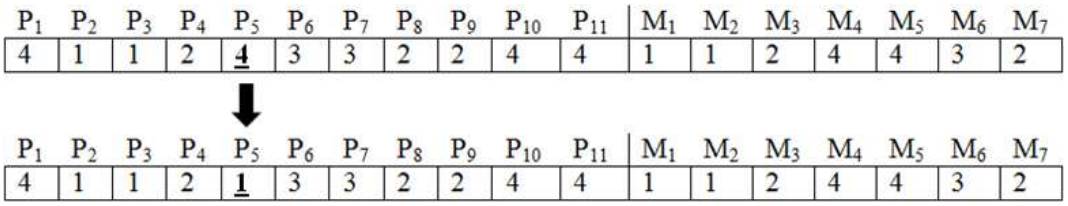


FIGURE 2.3 – HClonalSA1 : Recherche Tabou - mouvement d'insertion

- Mouvement d'échange ou swap : échanger les cellules de deux éléments qui sont localisés respectivement dans deux positions u et v. L'échange peut s'effectuer entre deux machines, deux produits ou entre un produit et une machine. Dans la Figure 2.4, le voisin de la solution initiale est obtenu en échangeant la cellule du produit 5 avec celle de la machine 2. Par conséquent, le produit 5 est déplacé dans la cellule 1 et la machine 2 est affectée à la cellule 4.

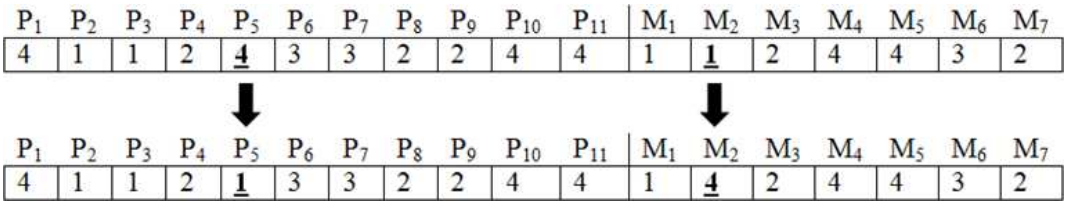


FIGURE 2.4 – HClonalSA1 : Recherche Tabou - mouvement d'échange

Un échantillon des solutions voisines est obtenu en appliquant ces deux mouvements. La taille de l'échantillon est fixe et elle est déterminée dans l'étude du paramétrage de la méthode. À chaque itération, la meilleure solution dans cet échantillon, atteinte par l'un des deux mouvements utilisés, est sélectionnée. Le choix de cette solution dépend de son efficacité.

b) Liste taboue :

La recherche Tabou est caractérisée par une liste appelée la liste taboue qui représente un mécanisme permettant d'éviter la répétition des solutions déjà trouvées et de guider le processus de recherche vers des régions non explorées. La taille de la liste utilisée est fixe. Lorsque la liste est pleine, le nouveau mouvement remplace le mouvement le plus ancien de la liste. Le déplacement du produit j vers la cellule k est ajouté à la liste taboue sous la forme $(1, j, k)$. De même, le déplacement d'une machine i vers une cellule k est représenté par $(2, i, k)$. Ces déplacements sont interdits tant qu'ils appartiennent à la liste taboue.

2.3.1.7 Algorithme du HClonalSA1

L'algorithme démarre avec une population initiale de solutions réalisables où chaque cellule de la solution contient au moins une machine et un produit. Puis, un pourcentage (noté b) de solutions, ayant les meilleures affinités dans la population, est sélectionné dans chaque itération. Les solutions choisies se multiplient en plusieurs clones (ou copies exactes) selon leurs affinités, c'est-à-dire, si l'affinité est élevée, le nombre de clones sera élevée (Equation (2.2)). Les clones sont ensuite combinés par un opérateur de croisement pour créer de nouvelles solutions et pour mieux explorer l'espace de recherche. Afin d'intensifier la recherche, l'algorithme de recherche Tabou est appliqué à l'une des nouvelles solutions choisie de manière aléatoire. Enfin, un pourcentage w des mauvaises solutions dans la population sont remplacés par les meilleures solutions obtenues (*receptor editing*). La méthode proposée s'arrête après un certain nombre d'itérations. Les phases de cette méthode sont illustrées dans l'algorithme 2.6.

Algorithme 2.6 Algorithme de HClonalSA1

Etape 1 : Génération de la population initiale ;

Etape 2 : **Tant que** critère d'arrêt est non satisfait **Faire** :

1. Calcul de l'affinité de chaque solution de la population en utilisant l'Equation (2.1) ;
2. Sélection d'un pourcentage b de solutions de la population ayant les meilleures affinités ;
3. Sélection clonale : le processus de génération des copies de ces meilleures solutions est effectué en fonction de la valeur d'affinité, la taille de la population (*popsize*) et la somme des affinités de la population (Equation (2.2)).
4. Application du croisement uniforme sur les copies des solutions générées ;
5. Application de la recherche Tabou sur une solution choisie aléatoirement ;
6. Remplacement d'un pourcentage w des solutions, ayant des mauvaises affinités, de la population par les meilleures solutions obtenues.

Fin Tant que

2.3.2 Résultats expérimentaux

La méthode HClonalSA1 est testée sur 30 instances de référence de différentes tailles collectées de la littérature. Ces instances sont classées en trois catégories : petites, moyennes et grandes instances en fonction du nombre d'entrées dans la matrice d'incidence. Pour plus de 30 machines, une instance est considérée de grande taille. Pour moins de 12 machines, une instance est de petite taille.

Nous avons implémenté HClonalSA1 en utilisant le langage Java. Les tests sont réalisés sur un ordinateur Pentium ® Dual Core CPU T4500 cadencé à 2.30 GHz et 2 GB de RAM.

2.3.2.1 Réglage de paramètres

Le choix de bons paramètres est une tâche importante qui influence la qualité de la solution finale et le temps de calcul nécessaire pour l'atteindre. Nos tests préliminaires ont indiqué que les valeurs suivantes semblent appropriées pour trouver les meilleures solutions :

— Taille de la population : $popsize = 30$;

— Nombre d'itérations : $5P + 1$ où P est le nombre de produits ;

— Nombre d'itérations pour la recherche Tabou : 20 ;
— Nombre de voisins pour la recherche Tabou : 20 (10 pour le mouvement d'insertion et 10 pour le mouvement d'échange) ;
— Longueur de la liste taboue : 5.

En se basant sur ce paramétrage, nous avons étudié le pourcentage b nécessaire pour la sélection clonale (Figure 2.5) et le pourcentage w requis pour le receptor editing (Figure 2.6). D'après les résultats rapportés dans les deux figures, nous avons fixé la valeur de b à 30% et la valeur de w à 20% car la plupart des instances ont atteint des meilleures solutions en utilisant ces valeurs.

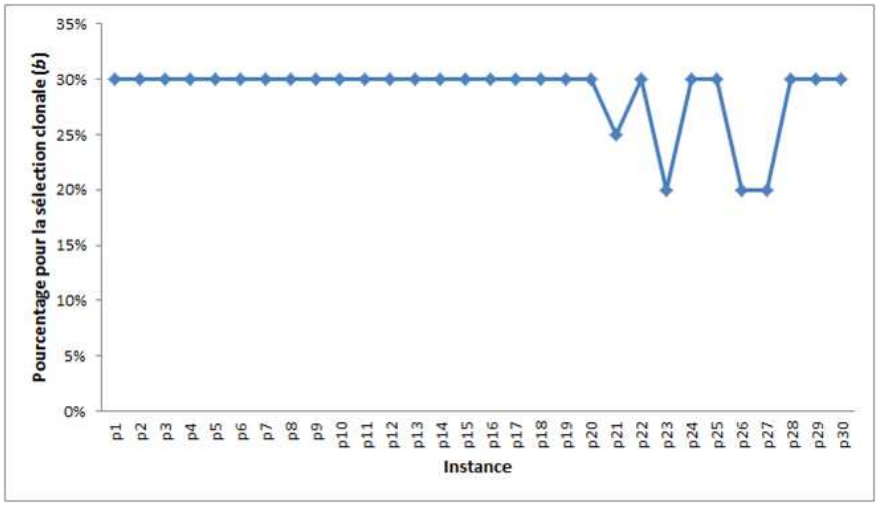


FIGURE 2.5 – HClonalSA1 : Paramétrage du pourcentage b pour la sélection clonale

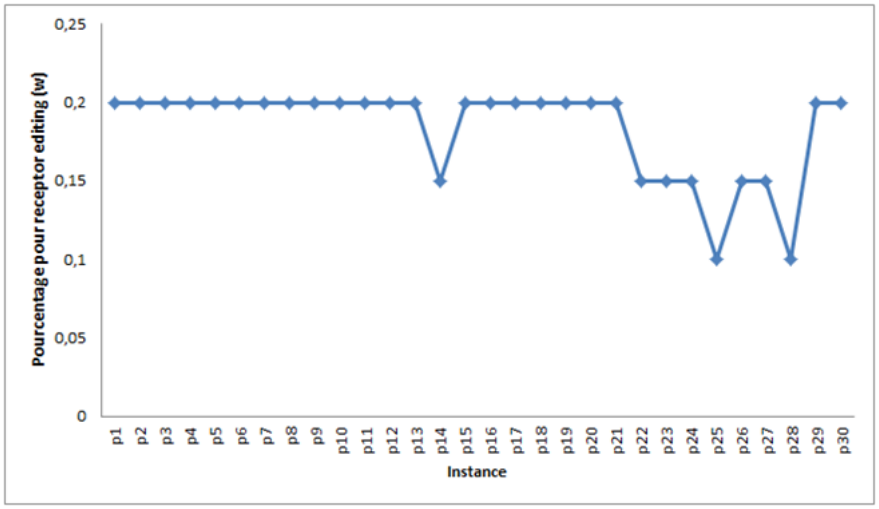


FIGURE 2.6 – HClonalSA1 : Paramétrage du pourcentage w pour receptor editing

2.3.2.2 Résultats

Pour chaque instance, 10 exécutions distinctes de HClonalSA1 avec les paramètres choisis sont effectuées et en utilisant des germes (*seed*) différents. Le Tableau 2.1 résume les résultats obtenus,

où :

No. : le numéro de l'instance ;

Source : la référence origine de l'instance ;

M : le nombre de machines ;

P : le nombre de produits ;

C : le nombre de cellules ;

Eff_Best : l'efficacité de la meilleure solution connue dans la littérature ;

Eff_Min : l'efficacité de la solution minimale obtenue durant 10 exécutions distinctes de HClonalSA1 ;

Eff_Avg : l'efficacité de la solution moyenne de 10 exécutions distinctes de HClonalSA1 ;

Eff_Max : l'efficacité de la meilleure solution trouvée par notre méthode HClonalSA1 durant 10 exécutions ;

Time(s) : temps de calcul moyen en secondes de 10 exécutions.

TABLE 2.1 – HClonalSA1 : Résultats expérimentaux pour 30 instances de référence du CFP

No.	Source	M	P	C	Eff_Best	Eff_Min	Eff_Avg	Eff_Max	Time(s)
1	(King et Nakornchai, 1982)	5	7	2	**82.35**	82.35	82.35	**82.35**	0.16
2	(Waghodekar et Sahu, 1984)	5	7	2	**69.57**	69.57	69.57	**69.57**	0.18
3	(Seifoddini, 1989)	5	18	2	**79.59**	79.59	79.59	**79.59**	0.78
4	(Kusiak et Cho, 1992)	6	8	2	**76.92**	76.92	76.92	**76.92**	0.26
5	(Kusiak et Chow, 1987)	7	11	5	**60.87**	60.87	60.87	**60.87**	1.42
6	(Boctor, 1991)	7	11	4	**70.83**	70.83	70.83	**70.83**	1.06
7	(Seifoddini et Wolfe, 1986)	8	12	4	**69.44**	69.44	69.44	**69.44**	1.37
8	(Chandrasekharan et Rajagopalan, 1986b)	8	20	3	**85.25**	85.25	85.25	**85.25**	2.39
9	(Chandrasekharan et Rajagopalan, 1986a)	8	20	2	**58.72**	58.72	58.72	**58.72**	1.36
10	(Mosier et Taube, 1985a)	10	10	5	**75**	75	75	**75**	1.66
11	(Chan et Milner, 1982)	10	15	3	**92**	92	92	**92**	1.66
12	(Askin et Subramantan, 1987)	14	24	7	**72.06**	72.06	72.06	**72.06**	19.63
13	(Stanfel, 1985)	14	24	7	**71.83**	71.83	71.83	**71.83**	19.99
14	(McCormick *et al.*, 1972)	16	24	8	**53.26**	52.75	53.20	**53.26**	28.47
15	(Srinlvasan *et al.*, 1990)	16	30	6	**69.53**	69.53	69.53	**69.53**	27.95
16	(King, 1980)	16	43	8	**57.53**	56.58	57,20	**57.53**	90.05
17	(Carrie, 1973)	18	24	9	**57.73**	57.29	57.64	**57.73**	38.84
18	(Kumar *et al.*, 1986)	20	23	7	**50.81**	50.81	50.81	**50.81**	27.12
19	(Carrie, 1973)	20	35	5	**77.91**	77.91	77.91	**77.91**	34.27
20	(Chandrasekharan et Rajagopalan, 1989)	24	40	7	**100**	100	100	**100**	94.89
21	(Chandrasekharan et Rajagopalan, 1989)	24	40	7	**85.11**	85.11	85.11	**85.11**	95.45
22	(Chandrasekharan et Rajagopalan, 1989)	24	40	7	**73.51**	73.51	73.51	**73.51**	90.60
23	(Chandrasekharan et Rajagopalan, 1989)	24	40	11	**53.29**	53.06	53.22	**53.29**	204.36
24	(Chandrasekharan et Rajagopalan, 1989)	24	40	12	**48.95**	48.28	48.53	**48.95**	228.71
25	(McCormick *et al.*, 1972)	27	27	5	**54.82**	52.85	54.42	**54.82**	27.02
26	(Carrie, 1973)	28	46	10	**47.06**	46.47	46.93	**47.06**	256.35
27	(Kumar et Vannelli, 1986)	30	41	14	**63.31**	62.14	62.71	**63.31**	412.58
28	(Stanfel, 1985)	30	50	14	**50.83**	50.27	50.52	**50.83**	599.45
29	(McCormick *et al.*, 1972)	37	53	3	**60.64**	60.08	60.52	**60.64**	61.04
30	(Chandrasekharan et Rajagopalan, 1987)	40	100	10	**84.03**	84.03	84.03	**84.03**	1635.63

D'après les résultats rapportés dans le Tableau 2.1, HClonalSA1 atteint l'efficacité des meilleures solutions connues (Eff_Best) pour les 30 instances. Parmi ces instances, les résultats sont stables

pour 20 instances (1 à 13, 15, 18 à 22 et 30) au cours des 10 exécutions de l'algorithme (Eff_Best == Eff_Min == Eff_Avg == Eff_Max), ce qui confirme la performance de HClonalSA1. Concernant les instances restantes, les efficacités des solutions moyennes sont très proches de celles des meilleures solutions connues. En outre, la méthode proposée peut atteindre la meilleure solution connue en consommant un temps de calcul raisonnable, moins de 30 secondes, pour les instances de petite et de moyenne tailles.

Pour confirmer la performance de la méthode proposée, les meilleures efficacités obtenues par HClonalSA1 sont comparées avec celles enregistrées par d'autres méthodes développées pour résoudre le CFP. Les méthodes utilisées dans l'étude comparative sont déjà présentées dans l'état d'art du problème de formation de cellules (voir Chapitre 1, Section 1.4) :

— ZODIAC : Méthode de clustering non-hiérarchique (Chandrasekharan et Rajagopalan, 1987) ;

— GRAFICS : Méthode de clustering non-hiérarchique (Srinivasan et Narendran, 1991) ;

— EA : Algorithme évolutionnaire (Gonçalves et Resende, 2004) ;

— ACO : Optimisation par colonies de fourmis (Li *et al.*, 2010) ;

— SA : Recuit simulé (Ying *et al.*, 2011) ;

— GRASP : Procédure de recherche gloutonne aléatoire adaptative (Díaz *et al.*, 2012).

Les résultats de la comparaison sont rapportés dans le Tableau 2.2. Les approches ZODIAC, GRAFICS, EA et GRASP n'autorisent pas les singletons (cellules ayant moins de deux machines ou deux produits) ce qui peut influencer la qualité des solutions par rapport à d'autres approches. Concernant les deux premières approches, les résultats des instances 3, 4 et 6 n'étaient pas disponibles.

Les résultats de la comparaison montrent que HClonalSA1 surpasse ZODIAC, GRAFICS, EA, ACO et GRASP en termes de qualité des solutions. HClonalSA1 a obtenu une meilleure solution dans l'instance 15, tandis que SA a pu atteindre un bon résultat lors de la résolution de l'instance 26. D'après cette comparaison, nous pouvons déduire le bon fonctionnement et l'efficacité de notre première méthode HClonalSA1.

Les résultats expérimentaux valident la performance de HClonalSA1 puisqu'il a pu atteindre les meilleures solutions connues de 30 instances de référence en un temps de calcul raisonnable. De plus, la comparaison avec différentes méthodes collectées de la littérature a montré la supériorité de notre méthode HClonalSA1.

TABLE 2.2 – HClonalSA1 : Performance de la méthode HClonalSA1 par rapport à d'autres méthodes

No.	ZODIAC	GRAFICS	EA	ACO	SA	GRASP	HClonalSA1
1	73.68	73.68	73.68	**82.35**	**82.35**	73.68	**82.35**
2	56.22	60.87	62.50	**69.57**	**69.57**	62.50	**69.57**
3	-	-	**79.59**	**79.59**	**79.59**	**79.59**	**79.59**
4	-	-	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**
5	39.13	53.12	53.13	**60.87**	**60.87**	53.13	**60.87**
6	-	-	70.37	**70.83**	**70.83**	70.37	**70.83**
7	68.30	68.30	68.29	**69.44**	**69.44**	**69.44**	**69.44**
8	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**
9	58.33	58.33	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**
10	70.59	70.59	70.59	**75.00**	**75.00**	70.59	**75.00**
11	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**
12	64.36	64.36	69.86	**72.06**	**72.06**	69.86	**72.06**
13	65.55	65.55	69.33	**71.83**	**71.83**	69.33	**71.83**
14	32.09	45.52	52.58	52.75	**53.26**	51.96	**53.26**
15	67.83	67.83	67.83	68.99	68.99	67.83	**69.53**
16	53.76	54.39	54.86	**57.53**	**57.53**	56.52	**57.53**
17	41.84	48.91	54.46	**57.73**	**57.73**	54.46	**57.73**
18	38.96	49.36	49.65	**50.81**	**50.81**	49.65	**50.81**
19	75.14	75.14	76.22	**77.91**	**77.91**	76.54	**77.91**
20	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
21	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**
22	37.85	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**
23	20.42	43.27	51.88	**53.29**	**53.29**	51.97	**53.29**
24	18.23	44.51	46.69	**48.95**	**48.95**	47.37	**48.95**
25	52.14	47.37	54.27	**54.82**	**54.82**	54.27	**54.82**
26	33.01	32.86	44.37	47.08	**47.23**	46.06	47.06
27	33.46	55.43	58.11	**63.31**	**63.31**	59.52	**63.31**
28	21.11	47.96	50.48	**50.83**	**50.83**	50.51	**50.83**
29	52.21	52.21	56.42	**60.64**	**60.64**	59.85	**60.64**
30	83.92	83.92	**84.03**	**84.03**	**84.03**	**84.03**	**84.03**

2.4 Deuxième méthode de résolution : HClonalSA2

La deuxième méthode **HClonalSA2** (the second Hybrid Clonal Selection Algorithm) présente une nouvelle adaptation de l'algorithme de sélection clonale pour résoudre le problème de formation de cellules (Karoum *et al.*, 2016b). Le croisement en un point est utilisé comme opérateur de génération de nouvelles solutions. Une nouvelle méthode de recherche locale, inspirée de (Gonçalves et Resende, 2004), est intégrée pour renforcer la recherche autour des solutions générées par l'opérateur de croisement. La performance de la méthode proposée est testée sur les 35 instances les plus étudiées dans la littérature. Les résultats obtenus sont comparés avec ceux d'autres méthodes développées pour optimiser la production cellulaire.

2.4.1 Adaptation de HClonalSA2 au CFP

Dans cette méthode, nous avons étudié d'autres instances plus difficiles à résoudre. Cela a nécessité l'intégration du croisement en un point au lieu du croisement uniforme. Ainsi, l'utilisation d'un autre mécanisme de recherche locale appliqué à toutes les solutions obtenues par le croisement contrairement à la première méthode.

2.4.1.1 Croisement en un point

Nous avons appliqué le croisement en un point deux fois sur une solution. Une première fois sur la partie des produits et une deuxième fois sur la partie des machines. Le choix des deux solutions parents est effectué aléatoirement. Un exemple d'application de l'opérateur de croisement sur une instance contenant 11 produits et 7 machines est illustré dans la Figure 2.7. Pour corriger les solutions non réalisables qui peuvent apparaitre durant cette opération, nous avons intégré le processus de réparation (voir l'algorithme 2.5) pour déplacer l'élément manquant (machine ou produit) de la cellule ayant plus d'éléments vers la cellule enregistrant un manque.

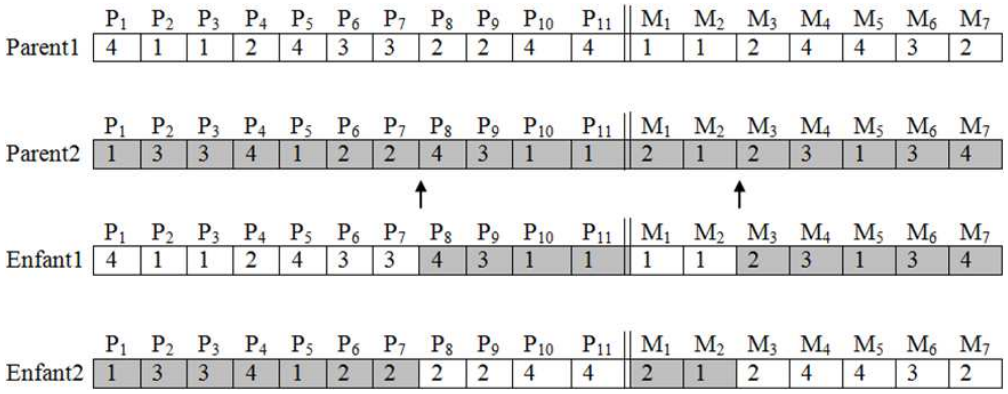


FIGURE 2.7 – HClonalSA2 : Exemple d'application du croisement en un point

2.4.1.2 Recherche locale

Pour intensifier la recherche et améliorer les solutions, nous avons inclus une méthode de recherche locale proposée par (Gonçalves et Resende, 2004). Cette méthode est simple à implémenter, utilise la même mesure d'efficacité et presque la même représentation de solutions. Son principe réside dans l'affectation des produits selon l'emplacement des machines dans les cellules puis la

réaffectation des machines en fonction des nouvelles positions de produits. Cette opération se répète jusqu'à ce qu'aucune amélioration ne soit possible et la solution soit stable.

Considérons les notations suivantes :

argmax	: Argument qui maximise l'expression ;
a_1	: Nombre total des 1 dans la matrice d'incidence ;
$a_{1,k}^{out}$	: Nombre total des 1 en dehors des blocs diagonaux d'une solution si l'élément à déplacer (produit ou machine) est affecté à la cellule k ;
$a_{0,k}^{in}$	: Nombre total des 0 dans des blocs diagonaux d'une solution $s0$ si si l'élément à déplacer est affecté à la cellule k.

L'algorithme 2.7 présente les étapes de la méthode de recherche locale. Dans cette thèse, cette méthode sera notée par **la recherche locale Goncalves**.

Algorithme 2.7 Algorithme de recherche locale Goncalves

Tant que le critère d'arrêt n'est pas atteint **Faire** :

Etape 1 : Réassigner les produits selon les emplacements des machines pour transférer chaque produit à la cellule contenant le plus de machines dédiées à son traitement.

1. Affecter chaque produit j, $j = 1, ..., P$, à une cellule C_j^* tel que :

$$C_j^* = \underset{k}{\text{argmax}} \left\{ \frac{a_1 - a_{1,k}^{out}}{a_1 + a_{0,k}^{in}} \right\} \tag{2.3}$$

2. Calculer l'efficacité de la nouvelle solution $s1$ obtenue après la réaffectation des produits.

Etape 2 : Déplacer les machines selon les nouvelles positions des produits. En effet, chaque machine est transférée à la cellule qui l'utilise le mieux.

1. Assigner chaque machine i, $i = 1, ..., M$, à une cellule C_i^* tel que :

$$C_i^* = \underset{k}{\text{argmax}} \left\{ \frac{a_1 - a_{1,k}^{out}}{a_1 + a_{0,k}^{in}} \right\} \tag{2.4}$$

2. Calculer l'efficacité de la nouvelle solution $s2$ obtenue après le déplacement des machines.

Etape 3 : Garder la meilleure solution obtenue à partir de $s1$ et $s2$ comme solution résultante (sr) de l'itération courante.

Fin Tant que

Critère d'arrêt : Si la section des machines des solutions reste la même entre l'étape 1 et l'étape 2 ou si l'efficacité de la solution résultante diminue, alors le processus itératif s'arrête et la solution obtenue à l'itération précédente sera la solution finale. Sinon, on passe à l'itération suivante.

La Figure 2.8 présente un exemple explicatif de la recherche locale Goncalves.

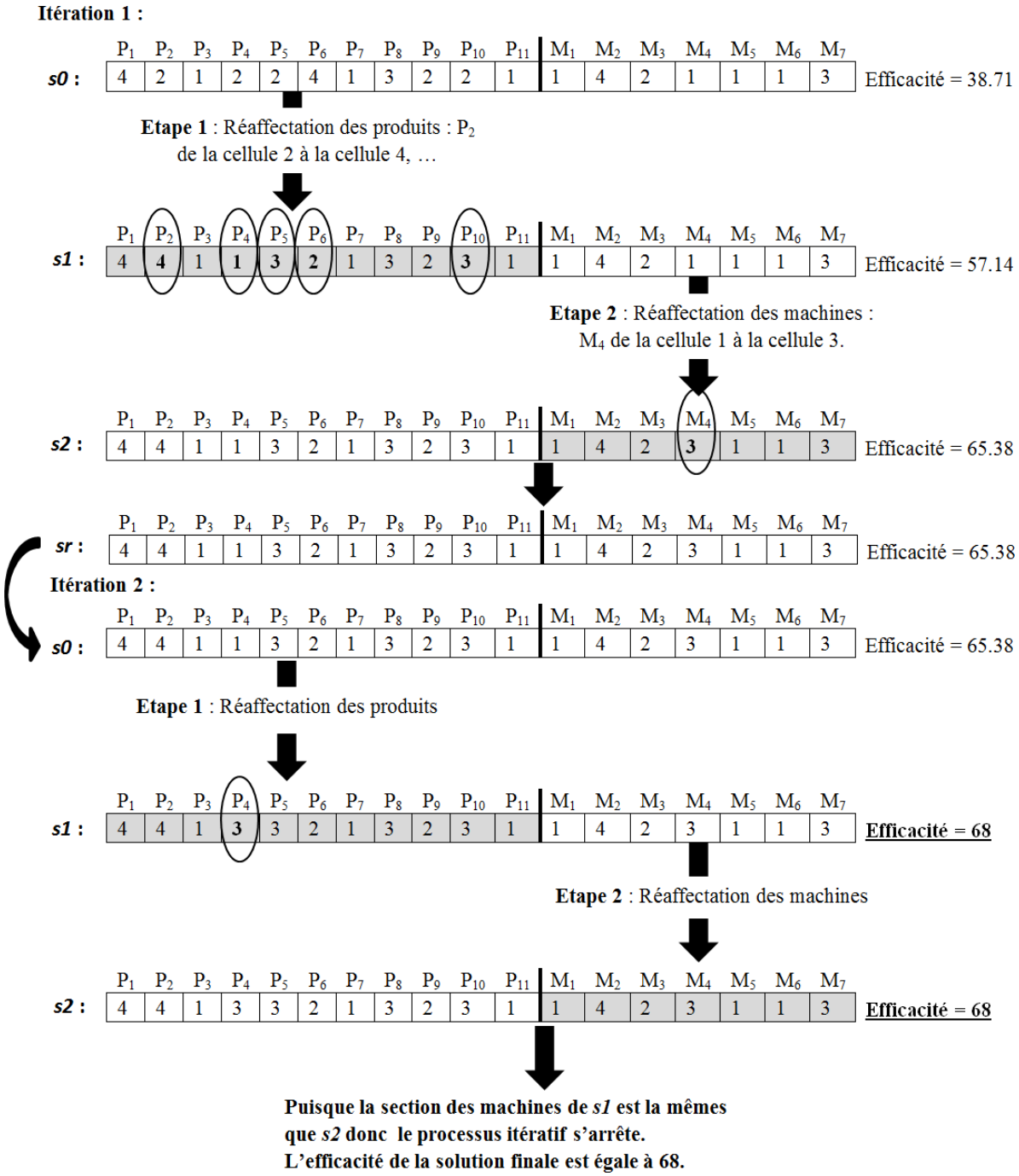


FIGURE 2.8 – HClonalSA2 : Exemple de la méthode de recherche locale Goncalves

2.4.1.3 Algorithme du HClonalSA2

Au début, HClonalSA2 génère une population de solutions initiales aléatoirement. Ensuite, un pourcentage b de solutions, ayant les meilleures affinités dans la population, est choisi pour la

sélection clonale en appliquant les Equations (2.1) et (2.2). Un croisement en un point est utilisé sur les clones obtenus. Chacune des solutions produites par cet opérateur subit une recherche locale. Un pourcentage w des solutions de mauvaises affinités seront remplacées par les meilleures solutions trouvées grâce à l'opérateur de croisement et la méthode de recherche locale. Les étapes de notre méthode HClonalSA2 sont introduites par l'algorithme 2.8.

Algorithme 2.8 Algorithme de HClonalSA2

Etape 1 : Génération de la population initiale ;

Etape 2 : **Tant que** critère d'arrêt est non satisfait **Faire** :

1. Calcul de l'affinité de chaque solution en utilisant l'Equation (2.1) ;
2. Sélection d'un pourcentage b de solutions de la population ayant les meilleures affinités ;
3. Sélection clonale : le processus de génération des copies de ces meilleures solutions est effectué en fonction de la valeur d'affinité, la taille de la population (*popsize*) et la somme des affinités de la population entière (Equation (2.2)).
4. **Application du croisement en un point sur les copies des solutions générées ;**
5. **Application de la recherche locale sur les solutions obtenues par le croisement ;**
6. Remplacement d'un pourcentage w des solutions, ayant des mauvaises affinités, de la population.

Fin Tant que

2.4.2 Résultats expérimentaux

La performance de la méthode HClonalSA2 est évaluée sur 35 instances de référence de différentes tailles. Le langage Java est utilisé pour le codage de HClonalSA2. Les tests sont effectués sur un ordinateur Pentium ® Dual Core CPU T4500 clock at 2.30 GHz et 2 GB de RAM.

2.4.2.1 Réglage de paramètres

Après avoir effectué des tests préliminaires, nous avons considéré quatre valeurs possibles pour le nombre d'itérations (50, 100, 200, 500), cinq valeurs pour *popsize* (15, 20, 30, 40, 50) et trois valeurs pour b et w (20%, 25%, 30%). Les résultats ont montré que les valeurs suivantes semblent appropriées pour trouver les meilleures solutions en consommant un temps de calcul raisonnable :

— Taille de la population : $popsize = 20$;

— Nombre d'itérations : 200 ;

— Pourcentage b nécessaire pour la sélection clonale : 25% ;

— Pourcentage w requis pour le receptor editing : 20%.

2.4.2.2 Résultats

10 exécutions distinctes de HClonalSA2 sont réalisées pour chaque instance. Le Tableau 2.3 introduit les résultats trouvés. D'après le Tableau 2.3, la méthode HClonalSA2 génère de très bonnes

solutions en consommant un temps de calcul raisonnable. Elle atteint les efficacités des meilleures solutions connues dans la littérature pour 29 instances de référence. Les résultats sont stables pour 17 instances (1 à 12, 19, 22 à 24 et 35) au cours des 10 exécutions de l'algorithme (Eff_Best == Eff_Min == Eff_Avg == Eff_Max). Concernant les instances restantes, les résultats obtenus sont très proches des meilleures solutions connues.

TABLE 2.3 – HClonalSA2 : Résultats expérimentaux pour 35 instances de référence du CFP

No.	Source	M	P	C	Eff_Best	Eff_Min	Eff_Avg	Eff_Max	Time(s)
1	(King et Nakornchai, 1982)	5	7	2	**82.35**	82.35	82.35	**82.35**	0.06
2	(Waghodekar et Sahu, 1984)	5	7	2	**69.57**	69.57	69.57	**69.57**	0.06
3	(Seifoddini, 1989)	5	18	2	**79.59**	79.59	79.59	**79.59**	0.10
4	(Kusiak et Cho, 1992)	6	8	2	**76.92**	76.92	76.92	**76.92**	0.07
5	(Kusiak et Chow, 1987)	7	11	5	**60.87**	60.87	60.87	**60.87**	0.18
6	(Boctor, 1991)	7	11	4	**70.83**	70.83	70.83	**70.83**	0.14
7	(Seifoddini et Wolfe, 1986)	8	12	4	**69.44**	69.44	69.44	**69.44**	0.18
8	(Chandrasekharan et Rajagopalan, 1986b)	8	20	3	**85.25**	85.25	85.25	**85.25**	0.29
9	(Chandrasekharan et Rajagopalan, 1986a)	8	20	2	**58.72**	58.72	58.72	**58.72**	0.15
10	(Mosier et Taube, 1985a)	10	10	5	**75**	75	75	**75**	0.24
11	(Chan et Milner, 1982)	10	15	3	**92**	92	92	**92**	0.21
12	(Askin et Subramantan, 1987)	14	24	7	**72.06**	72.06	72.06	**72.06**	5.13
13	(Stanfel, 1985)	14	24	7	**71.83**	70.59	71.17	**71.83**	4.62
14	(McCormick *et al.*, 1972)	16	24	8	**53.26**	51.09	52.17	**53.26**	8.62
15	(Srinlvasan *et al.*, 1990)	16	30	6	**69.53**	63.43	67.09	**69.53**	8.15
16	(King, 1980)	16	43	8	**57.53**	57.32	57.43	**57.53**	32.82
17	(Carrie, 1973)	18	24	9	**57.73**	56.84	57.46	**57.73**	11.10
18	(Mosier et Taube, 1985b)	20	20	5	**43.45**	42.45	42.94	**43.45**	2.77
19	(Kumar *et al.*, 1986)	20	23	7	**50.81**	50.81	50.81	**50.81**	7.01
20	(Carrie, 1973)	20	35	5	**77.91**	71.93	77.31	**77.91**	10.49
21	(Boe et Cheng, 1991)	20	35	5	**57.98**	57.22	57.90	**57.98**	10.96
22	(Chandrasekharan et Rajagopalan, 1989)	24	40	7	**100**	100	100	**100**	33.04
23	(Chandrasekharan et Rajagopalan, 1989)	24	40	7	**85.11**	85.11	85.11	**85.11**	41.07
24	(Chandrasekharan et Rajagopalan, 1989)	24	40	7	**73.51**	73.51	73.51	**73.51**	34.60
25	(Chandrasekharan et Rajagopalan, 1989)	24	40	11	**53.29**	52.94	53.20	**53.29**	94.99
26	(Chandrasekharan et Rajagopalan, 1989)	24	40	12	**48.95**	47.89	48.29	48.61	97.39
27	(Chandrasekharan et Rajagopalan, 1989)	24	40	12	**47.26**	44.90	45.79	46.21	87.63
28	(McCormick *et al.*, 1972)	27	27	5	**54.82**	52.85	54.42	**54.82**	7.50
29	(Carrie, 1973)	28	46	10	**47.23**	44.94	46.93	47.08	163.68
30	(Kumar et Vannelli, 1986)	30	41	14	**63.31**	62.24	62.80	**63.31**	276.54
31	(Stanfel, 1985)	30	50	13	**60.12**	58.86	59.57	59.77	295.12
32	(Stanfel, 1985)	30	50	14	**50.83**	50.00	50.44	**50.83**	355.11
33	(King et Nakornchai, 1982)	36	90	17	**47.75**	45.66	46.49	47.71	2489.81
34	(McCormick *et al.*, 1972)	37	53	3	**60.64**	59.41	59.41	59.41	18.28
35	(Chandrasekharan et Rajagopalan, 1987)	40	100	10	**84.03**	84.03	84.03	**84.03**	1527.35

La dernière colonne du Tableau 2.3 montre que les 11 premières instances sont résolues en moins d'une seconde et les 11 autres instances sont résolues en moins de 20 secondes. En ce qui concerne les restes, malheureusement le temps de calcul augmente avec la taille du l'instance et le nombre de cellules fixé auparavant. Selon ces tests, on peut conclure que la performance de HClonalSA2 est plutôt constante sur la majorité des instances de différentes tailles.

Le Tableau 2.4 rapporte les résultats de comparaisons des meilleures efficacités obtenues par la méthode HClonalSA2 et par six méthodes développées dans la littérature :

— GRAFICS : Méthode de clustering non-hiérarchique (Srinivasan et Narendran, 1991) ;

— EA : Algorithme évolutionnaire (Gonçalves et Resende, 2004) ;

— SA : Recuit simulé (Ying *et al.*, 2011) ;

— GRASP : Procédure de recherche gloutonne aléatoire adaptative (Díaz *et al.*, 2012) ;

— GA_LNS : Algorithme génétique (Elbenani *et al.*, 2012) ;

— HGBPSO : Optimisation par essaims particulaires (Kashan *et al.*, 2014).

TABLE 2.4 – HClonalSA2 : Performance de la méthode HClonalSA2 par rapport à d'autres méthodes

No.	GRAFICS	EA	SA	GRASP	GA_LNS	HGBPSO	HCLONALSA2
1	73.68	73.68	**82.35**	73.68	**82.35**	**82.35**	**82.35**
2	60.87	62.50	**69.57**	62.50	**69.57**	**69.57**	**69.57**
3	-	**79.59**	**79.59**	**79.59**	**79.59**	**79.59**	**79.59**
4	-	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**
5	53.12	53.13	**60.87**	53.13	**60.87**	**60.87**	**60.87**
6	-	70.37	**70.83**	70.37	**70.83**	**70.83**	**70.83**
7	68.30	68.29	**69.44**	**69.44**	**69.44**	**69.44**	**69.44**
8	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**
9	58.33	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**
10	70.59	70.59	**75.00**	70.59	**75.00**	**75.00**	**75.00**
11	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**
12	64.36	69.86	**72.06**	69.86	**72.06**	**72.06**	**72.06**
13	65.55	69.33	**71.83**	69.33	**71.83**	**71.83**	**71.83**
14	45.52	52.58	**53.26**	51.96	**53.26**	53.41	**53.26**
15	67.83	67.83	68.99	67.83	**69.53**	68.99	**69.53**
16	54.39	54.86	**57.53**	56.52	**57.53**	**57.53**	**57.53**
17	48.91	54.46	**57.73**	54.46	**57.73**	**57.73**	**57.73**
18	38.26	42.94	**43.45**	42.96	**43.45**	43.26	**43.45**
19	49.36	49.65	**50.81**	49.65	**50.81**	**50.81**	**50.81**
20	75.14	76.22	**77.91**	76.54	**77.91**	**77.91**	**77.91**
21	-	58.07	**57.98**	58.15	**57.98**	**57.98**	**57.98**
22	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
23	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**
24	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**
25	43.27	51.88	**53.29**	51.97	**53.29**	**53.29**	**53.29**
26	44.51	46.69	**48.95**	47.37	**48.95**	**48.95**	48.61
27	41.67	44.75	**47.26**	44.87	47.06	**47.26**	46.21
28	47.37	54.27	**54.82**	54.27	**54.82**	**54.82**	**54.82**
29	32.86	44.37	**47.23**	46.06	47.06	-	47.08
30	55.43	58.11	**63.31**	59.52	63.12	**63.31**	**63.31**
31	56.32	59.21	59.77	60.00	**60.12**	59.77	59.77
32	47.96	50.48	**50.83**	50.51	**50.83**	-	**50.83**
33	39.41	42.12	47.14	45.93	**47.75**	-	47.71
34	52.21	56.42	**60.64**	59.85	60.63	**60.64**	59.41
35	83.92	**84.03**	**84.03**	**84.03**	**84.03**	-	**84.03**

Le Tableau 2.4 montre que, pour la plupart des instances, les résultats de HClonalSA2 sont égaux parfois meilleurs aux résultats des algorithmes de comparaison. Sur six instances, l'efficacité obtenue est très proche de celle des meilleures solutions connues.

En ce qui concerne l'instance 14, la solution optimale est déjà atteinte en utilisant une méthode exacte (Elbenani et Ferland, 2012) ; son efficacité est égale à 53.26 (la même valeur atteinte par notre méthode), ce qui contredit celle obtenue par la méthode HGBPSO. En outre, les données rapportées pour l'instance 21 dans les méthodes EA et GRASP sont incompatibles avec les données de l'instance originale dans la littérature.

Suite aux tests réalisés, nous avons prouvé l'efficacité de HClonalSA2. Cette méthode a pu résoudre 82.86% des instances avec un temps de calcul raisonnable. En outre, les résultats de comparaison avec six différents algorithmes ont confirmé la bonne performance de notre méthode en terme de la qualité des solutions.

2.5 Conclusion

Tout au long de ce chapitre, nous avons proposé deux méthodes basées sur l'algorithme de sélection clonale. Chaque méthode est adaptée différemment au problème de formation de cellules. Pour intensifier la recherche, nous avons intégré des mécanismes de recherche locale.
Les méthodes proposées présentent plusieurs points forts tels que la simplicité et la rapidité de résolution. Les tests réalisés ont montré que ces méthodes jouissent d'une performance élevée pour résoudre le CFP. En effet, elles ont réussi à résoudre plus de 80% des instances tout en consommant un temps raisonnable.
L'idée motif du choix de l'algorithme de sélection clonale demeure dans ces caractéristiques : la sélection clonale, la reproduction proportionnelle à l'affinité et l'hyper-mutation. En outre, d'après la littérature, nous sommes les premiers à adapter cet algorithme au problème de formation de cellules.
Dans le chapitre qui suit, nous allons introduire un autre algorithme inspiré de la nature pour résoudre le problème.

Chapitre

3 CFP avec cellules fixes : Algorithme de recherche du coucou

Sommaire

3.1 Introduction

Le coucou est un oiseau discret, longiligne, de taille moyenne d'environ 30 cm, avec un chant fascinant, caractérisé par sa stratégie de reproduction. Il est considéré comme un bon exemple d'oiseaux parasites de couvée. Il est connu par son habitude de parasiter les nids d'autres espèces, précédemment observés, en y pondant ses œufs. La femelle coucou retire un des œufs du nid hôte (parasité) avant d'y pondre le sien pour éviter d'être repéré. Après presque 11 à 13 jours, le petit coucou éclot et éjecte immédiatement les autres œufs hors du nid. Il arrive même à imiter le cri des poussins hôtes pour être nourri par l'oiseau hôte. Si l'oiseau hôte découvert la présence de l'œuf coucou dans son nid, il va soit s'en débarrasser de l'œuf étranger ou bien abandonner son propre nid et construire un nouveau ailleurs.

FIGURE 3.1 – Rousserolle effarvatte nourrissant un poussin coucou gris (Crédits : Per Harald Olsen - Wikipédia)

Dans ce chapitre, nous allons décrire l'algorithme de la recherche du coucou, sa formulation et les motivations derrière notre choix. Nous présentons également un aperçu sur les différentes applications de cet algorithme. Ensuite, nous détaillons nos méthodes proposées et nous les testons sur un ensemble d'instances pour évaluer leurs qualités.

3.2 Algorithme de la recherche du coucou

La recherche du coucou (CS) est une métaheuristique récente proposée par (Yang et Deb, 2009). Elle s'inspire du mode de reproduction parasitaire de quelques espèces de coucous. Elle combine le comportement de parasitisme de couvées chez les coucous avec un mécanisme du vol de Lévy pour chercher efficacement un nouveau nid.

3.2.1 Vol de Lévy

Le pas de déplacement des coucous est déterminé par les vols de Lévy. Le vol de Lévy ou Lévy flight est un modèle de marches aléatoires proposé par le mathématicien français Paul Lévy, un des fondateurs de la théorie moderne de probabilités. Ces marches sont caractérisées par les longueurs de pas qui suivent une distribution de loi de puissance (Figure 3.2). La marche aléatoire à travers les vols de Lévy est plus efficace pour explorer l'espace de recherche que les autres marches aléatoires, car la taille de son pas est beaucoup plus grande à long terme.

La distribution de Lévy a une panoplie de variantes avec une infinité de sens. Elle peut être approchée à $y = s^{-\beta}$ avec β un entier compris entre 1 et 3 :

$$lévy(\beta) \sim y = s^{-\beta},\ 1 < \beta \leq 3 \tag{3.1}$$

Des études expérimentales ont montré que le vol de Lévy a été interprété à un ensemble de phénomènes physiques, chimiques, et naturels dans différents domaines : les mouvements des organismes biologiques (Viswanathan *et al.*, 2002), le déplacement des mouches de fruits (Reynolds et Frye,

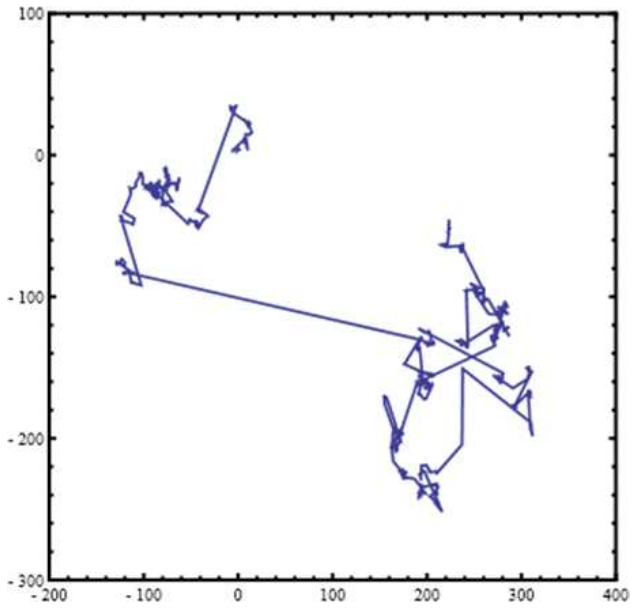


FIGURE 3.2 – Exemple de 1000 pas par les vols de Lévy en 2 dimensions

2007), la diffusion de la lumière (Barthelemy *et al.*, 2008), la recherche aléatoire des objets (Raposo *et al.*, 2009). De plus, il a été utilisé dans la résolution des problèmes d'optimisation (Pavlyukevich, 2007), (Balamurugan *et al.*, 2016), etc.

De leur part, Yang et Deb (2009) ont intégré le vol de Lévy dans le développement de l'algorithme de la recherche du coucou. Le but est la génération de nouvelles solutions autour de la meilleure solution obtenue et l'accélération de la recherche globale. Ils ont introduit un schéma simple pour décrire la notion de vol de Lévy :

$$lévy(\beta) = 0.01\frac{\mu}{|\nu|^{1/\beta}}(s_b^{(t)} - s_l^{(t)}),\ 1 < \beta \leq 3 \tag{3.2}$$

Où μ et ν sont tirés de la distribution normale :

$$\mu \sim N(0, \sigma_\mu^2),\ \nu \sim N(0, \sigma_\nu^2) \tag{3.3}$$

Avec :

$$\sigma_\mu = \left(\frac{\Gamma(1+\beta)\sin(\frac{\Pi\beta}{2})}{\Gamma(\frac{1+\beta}{2})\beta 2^{\frac{(\beta-1)}{2}}} \right)^{\frac{1}{\beta}}, \sigma_\nu = 1 \tag{3.4}$$

Où Γ est la fonction Gamma standard.

3.2.2 Description de l'algorithme

En s'inspirant du mode de reproduction des coucous, Yang et Deb (2009) se sont basés sur les trois règles suivantes pour développer l'algorithme de la recherche du coucou (CS) :

— Chaque coucou pond un seul œuf à la fois et il le dépose dans un nid choisi aléatoirement.

— Les meilleurs nids avec des œufs de haute qualité (solutions) seront transmis aux prochaines itérations.

— Le nombre de nids hôtes est fixe. Un oiseau hôte peut détecter un œuf pondu par un coucou étranger avec une probabilité $p_a \in [0, 1]$. Dans ce cas, l'oiseau hôte choisit entre se débarrasser de l'œuf en l'éjectant hors nid ou bien abandonner son nid et reconstruire un autre quelque part. Pour simplifier, la probabilité p_a représente la fraction des *popsize* nids qui seront remplacés par de nouveaux nids (avec de nouvelles solutions aléatoires dans des positions différentes).

La génération d'une nouvelle solution $s_l^{(t+1)}$ à l'itération $t + 1$ à partir d'une solution $s_l^{(t)}$ de l'itération t, pour un coucou l, est réalisée par la formule ci-dessous :

$$s_l^{(t+1)} = s_l^{(t)} + \alpha \oplus l\acute{e}vy(\beta) \tag{3.5}$$

Dans l'Equation (3.5), le terme $(s_l^{(t)})$ indique la position courante du coucou, suivi par la probabilité de transition qui est le terme $(\alpha \oplus l\acute{e}vy(\beta))$. Le produit $\oplus$ représente le produit matriciel. α est la taille maximale du pas et elle est liée au problème étudié. Dans la plupart des problèmes, α est égale à 1. L'algorithme 3.1 présente les étapes générales de l'algorithme de recherche du coucou CS.

Algorithme 3.1 Algorithme de la recherche du coucou

Fonction objectif $F(s)$
Générer une population initiale de *popsize* solutions (nids)
Tant que critère d'arrêt est non satisfait **Faire**
  Générer une solution s_1 du coucou en utilisant le vol de Lévy
  Calculer sa qualité/fitness $F(s_1)$
  Choisir au hasard une solution s_2 de la population
  Si $F(s_1) > F(s_2)$ **Alors**
    la nouvelle solution s_1 remplace la solution s_2
  Fin Si
  Abandonner une fraction p_a des mauvaises solutions dans la population et générer des nouvelles solutions
  Classer les solutions de la population
Fin Tant que

L'un des avantages de l'algorithme de recherche du coucou est le nombre réduit de paramètres à adapter par rapport aux autres méthodes ; ce qui le rend plus générique et plus efficace pour une large gamme de problèmes d'optimisation. En outre, cet algorithme se caractérise par la stabilité entre l'exploitation et l'exploration de l'espace de recherche. L'exploitation de la solution est renforcée par la recherche locale aléatoire, tandis que l'exploration est réalisée par la génération aléatoire de nouvelles solutions en utilisant le vol de Lévy.

Yang et Deb (2010) ont effectué une comparaison entre CS, GA et PSO. Les résultats ont montré que CS surpasse les performances de GA et PSO dans un certain nombre de problèmes d'optimisation tels que le problème d'optimisation de la conception des ressorts (*Spring Design Optimisation*) et le problème de conception d'un faisceau soudé (*Welded Beam Design*). Le fait de combiner la recherche du coucou avec le vol de Lévy rend la recherche des solutions plus efficace et permet la génération des solutions suffisamment diversifiées. De plus, contrairement aux autres métaheuristiques, CS utilise deux paramètres de contrôle seulement : la taille de la population n et la fraction p_a des nids à abandonner. Le nombre réduit des paramètres rend l'algorithme CS moins complexe et plus générique.

3.2.3 Applications de CS

Depuis son apparition, CS a été cité plus que 2000 fois dans différents domaines. Les caractéristiques de cet algorithme ont facilité son adaptation aux différents types de problèmes. Par la suite, nous allons citer quelques exemples d'applications du CS.

Valian *et al.* (2011) ont proposé une nouvelle version de l'algorithme CS qui améliore la précision et le taux de convergence de l'algorithme standard et ils l'ont appliqué pour former les réseaux de neurones de type «feed forward».

Durgun et Yildiz (2012) ont appliqué CS pour résoudre les problèmes d'optimisation de la conception structurale des véhicules. Leur but est d'améliorer le coût et l'efficacité du carburant et de répondre aux exigences actuelles de conception des véhicules.

Gherboudj *et al.* (2012) ont introduit une version discrète et binaire de l'algorithme de CS pour résoudre le problème du sac à dos. Pour obtenir des solutions binaires, les auteurs ont utilisé une fonction sigmoïde similaire à celle utilisée dans l'algorithme PSO binaire.

Dans le domaine de l'énergie, Abdelaziz et Ali (2015) ont utilisé CS pour contrôler la fréquence de charge d'un système d'alimentation interconnecté non linéaire dans le but d'améliorer la réponse du système en termes de temps de règlements et de dépassements. Les résultats obtenus par leur méthode surpassent ceux de GA et PSO. Chaine et Tripathy (2015) ont traité la conception d'un système optimal de stockage d'énergie magnétique afin de contrôler la génération automatique d'un système d'énergie thermique à deux zones.

En outre, CS contribue à l'économie d'énergie et la génération distribuée. Kumari et Shukla (2015) ont étudié l'attribution optimale de l'énergie distribuée dans un système de distribution. Ils ont utilisé l'algorithme de CS pour améliorer la qualité de la tension et minimiser la perte de puissance.

Dans le domaine de réseau et de télécommunications, Ramakrishnan *et al.* (2015) ont proposé une conception pour le protocole de routage adaptatif en utilisant CS. Leur objectif est d'assurer la transmission sécurisée des données avec moins de retard et avec un taux élevé de livraison des paquets.

Nguyen et Truong (2015) ont utilisé l'algorithme de recherche du coucou pour reconfigurer le réseau de distribution dans le but de minimiser le pourcentage de perte de la puissance et maximiser la tension.

Puntura *et al.* (2016) ont également développé une méthode pour optimiser les paramètres des machines à vecteurs de support (SVM) ou séparateurs à vaste marge. Une étude comparative a montré que cette méthode donne de meilleurs résultats en comparaison avec l'algorithme PSO.

Yasar (2016) ont développé une solution basée sur l'algorithme CS pour optimiser l'exploitation du réservoir en tenant en compte la capacité de stockage, l'afflux, l'élévation de l'eau et l'évaporation. Les tests sont effectués sur le barrage Adiguzel en Turquie. L'approche proposée a montré une convergence constante dans la recherche des solutions.

Yuan *et al.* (2016) ont appliqué une version améliorée du CS pour promouvoir la capacité de détection du spectre dans les systèmes cognitifs satellitaires. La méthode proposée consiste à ajuster dynamiquement la taille des pas et la probabilité de découverte afin d'éviter la recherche redondante, et augmenter ainsi le taux de convergence et améliorer les capacités de recherche des solutions optimales.

Kashyap *et al.* (2017) ont introduit une nouvelle approche pour détecter copy-move forgery, l'une des opérations simples et efficaces pour créer des images forgées ou fabriquées. Cette approche intègre l'algorithme CS avec la technique de décomposition en valeurs singulières pour générer des valeurs personnalisées de paramètres des images.

L'algorithme de la recherche de coucou a été également appliqué pour la résolution du problème d'ordonnancement de type Job Shop (Ouaarab *et al.*, 2014), l'ordonnancement des cellules robotiques (Majumder et Laha, 2016), le dispatching économique (Tran *et al.*, 2015) et (Sekhar et Mohanty, 2016), le traitement d'images (Suresh et Lal, 2016) (Pare *et al.*, 2016), le clustering (Jia *et al.*, 2016) et les applications médicales (Daniel et Anitha, 2016), etc.

La robustesse de l'algorithme de CS réside dans la manière d'explorer et d'exploiter l'espace des solutions à l'aide des coucous. Le comportement du parasitisme de couvées montre l'intelligence des coucous dans la détection des meilleures solutions (nids). De plus, l'intégration du vol de Lévy et le nombre réduit de paramètres rendent CS une méthode très efficace. D'où vient l'idée de l'adapter à notre problème de formation de cellules de production. Il est à noter que nous sommes les premiers à appliquer CS pour résoudre ce problème. Nous avons adapté cet algorithme sous deux formes basées sur l'utilisation du vol de Lévy. Par la suite, nous présentons une description détaillée de ces deux formes.

3.3 Première forme : HCuckooSA1

Notre première forme est notée HCuckooSA1 (the first Hybrid Cuckoo Search Algorithm-Cell Formation Problem) et elle combine les caractéristiques de l'algorithme de CS avec une méthode de recherche locale pour renforcer la recherche intensive autour des solutions (Karoum *et al.*, 2018). La méthode proposée est testée sur 35 instances de référence. Les résultats obtenus sont comparés avec d'autres résultats disponibles dans la littérature. Les tests ont montré que HCuckooSA1 peut atteindre l'efficacité des meilleures solutions connues dans la littérature pour 31 instances (88.57% des instances). Par la suite, nous allons montrer comment adapter l'algorithme de recherche du coucou au CFP et comment se déplacer dans l'espace de recherche en utilisant les vols de Lévy.

3.3.1 Adaptation de HCuckooSA1 au CFP

Le mécanisme d'adaptation de l'algorithme de recherche du coucou au CFP repose principalement sur une ré-interprétation de ses éléments principaux : nids et vols de Lévy. Ces éléments clés sont très importants pour la transition vers des problèmes d'optimisation combinatoire.

3.3.1.1 Nid

Dans l'algorithme de recherche du coucou :

— Un nid est une solution de la population ;

— Un nid abandonné implique le remplacement d'une solution par une autre ;

— le nombre de nids est fixe et égale à la taille de la population ;

Dans notre méthode, nous supposons qu'un coucou pose un seul œuf dans un nid ; Cet œuf représente une nouvelle solution.

3.3.1.2 Génération d'une solution par le Vol de Lévy

Le vol de Lévy représente un modèle de marches aléatoires caractérisé par la longueur des pas. Pour appliquer la méthode HCuckooSA1 au CFP, nous avons représenté la différence entre les deux positions $(s_b^{(t)} - s_l^{(t)})$ de l'Equation (3.2) par les mouvements nécessaires pour se déplacer de la solution courante $s_l^{(t)}$ vers la meilleure solution obtenue $s_b^{(t)}$.

Pour chacun des mouvements nécessaires, un nombre aléatoire entre 0 et 1 est généré ; si sa valeur est inférieure au coefficient $Coef = \frac{\mu}{|\nu|^{1/\beta}}$, le mouvement sera appliqué à la solution courante.
Un mouvement est modélisé par (i, e, k) où i présente le type de l'élément à déplacer ($i = 1$ pour un produit et $i = 2$ pour une machine), e est le numéro (l'indice) de l'élément et k est la nouvelle cellule attribuée.
La structure de la solution est celle déjà exposée dans le chapitre 2 et qui est codée sous forme d'un vecteur de longueur $P + M$: $s = (P_1, \ldots, P_P | M_1, \ldots, M_M)$.

Exemple illustratif : Prenons une instance ayant 11 produits et 7 machines (Boctor, 1991).
La meilleure solution obtenue est : $s_b = (2, 4, 2, 3, 1, 4, 2, 1, 4, 1, 3 \mid 2, 4, 4, 1, 2, 3, 1)$, $Eff(s_b) = 70.83$
Si notre solution courante est : $s_l = (4, 4, 3, 2, 1, 4, 2, 1, 4, 3, 3 \mid 3, 4, 4, 1, 2, 4, 3)$, $Eff(s_l) = 34.38$

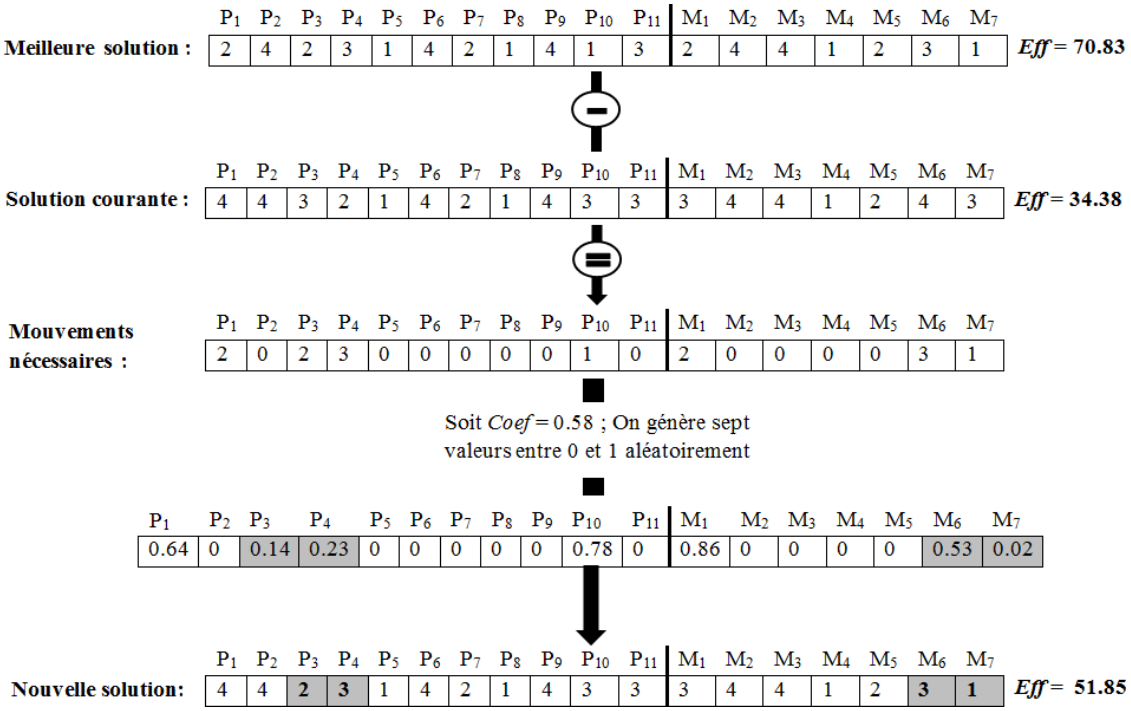


FIGURE 3.3 – HCuckooSA1 : Application du vol de Lévy

Comme décrit dans la Figure 3.3, la soustraction entre la meilleure solution et la solution courante a montré l'existence de sept mouvements nécessaires pour atteindre la meilleure solution :

- Mettre les deux produits 1 et 3 dans la cellule 2 : $(1, 1, 2), (1, 3, 2)$;
- Déplacer le produit 4 dans la cellule 3 : $(1, 4, 3)$;
- Déplacer le produit 10 dans la cellule 1 : $(1, 10, 1)$;
- Mettre la machine 1 dans la cellule 2 : $(2, 1, 2)$;
- Déplacer la machine 6 dans la cellule 3 : $(2, 6, 3)$.
- Mettre la machine 7 dans la cellule 1 : $(2, 7, 1)$;

Nous avons donc généré aléatoirement sept valeurs comprises entre 0 et 1 (0.64, 0.14, 0.23, 0.78, 0.86, 0.53, 0.02) et nous les avons comparées avec la valeur du $Coef (= \frac{\mu}{|\nu|^{1/\beta}} = \frac{0.26}{|0.30|^{1/1.5}} = 0.58)$.
Les valeurs de μ et ν sont définies en utilisant les Equations (3.3) et (3.4), et en fixant la valeur de β à 1.5.

Le mouvement ayant une valeur inférieure à celle du $Coef$ sera appliqué à la solution courante. Dans ce cas, quatre mouvements sont appliqués :
- Deux mouvements concernant les produits P_3 et P_4 : (1, 3, 2) et (1, 4, 3).
- Deux déplacements pour les machines M_6 et M_7 : (2, 6, 3) et (2, 7, 1).

La nouvelle solution s'est rapprochée de la meilleure solution en enregistrant une valeur égale à 51.85 :

La solution résultante est : $s_l = (4, 4, \mathbf{2}, \mathbf{3}, 1, 4, 2, 1, 4, 3, 3 \mid 3, 4, 4, 1, 2, \mathbf{3}, \mathbf{1}), \quad Eff(s_l) = 51.85$

3.3.1.3 Algorithme du HCuckooSA1

L'algorithme commence par une phase d'initialisation des valeurs des paramètres : le nombre de machines M, le nombre de produits P, le nombre de cellules C, la taille de la population $popsize$ et la fraction p_a des nids abandonnés. Ensuite, une population initiale de solutions réalisables est générée où chaque machine i et produit j sont attribués de manière aléatoire à une cellule k comme décrit dans la Section 2.3.1.2 du Chapitre 2. Après l'initialisation, la méthode proposée s'exécute jusqu'à atteindre un nombre spécifique d'itérations.
À chaque itération, un coucou se déplace par le vol de Lévy de la solution courante vers une nouvelle solution. Ensuite, la méthode de recherche locale proposée par (Gonçalves et Resende, 2004) est appliquée à la solution courante pour intensifier la recherche vers des régions prometteuses (voir Chapitre 2, Section 2.4.1.2). Si la qualité de la nouvelle solution trouvée est meilleure que celle d'un nid choisi au hasard de la population, la nouvelle solution remplace l'œuf (solution) choisi. Enfin, une fraction p_a des mauvais nids est remplacée par des nouveaux. L'algorithme 3.2 présente les étapes de notre méthode HCuckooSA1.

Algorithme 3.2 Algorithme du HCuckooSA1

Générer une population initiale de *popsize* solutions (nids)
Tant que critère d'arrêt est non satisfait **Faire**
  Générer une solution s_1 du coucou en utilisant le vol de Lévy
  Calculer sa qualité/fitness $F(s_1)$
  Appliquer la recherche locale Goncalves à la solution s_1
  Choisir au hasard une solution s_2 de la population
  Si $F(s_1) > F(s_2)$ **Alors**
    la nouvelle solution s_1 remplace la solution s_2
  Fin Si
  Abandonner une fraction p_a des mauvaises solutions dans la population et générer des nouvelles solutions
  Classer les solutions de la population
Fin Tant que

3.3.2 Résultats expérimentaux

La performance de la méthode HCuckooSA1 est testée sur 35 instances de référence. La taille de ces instances varie de 7 produits et 5 machines à 100 produits et 40 machines afin d'évaluer l'efficacité de la méthode. Le langage Java est utilisé pour le codage de HCuckooSA1. Les tests sont effectués sur un ordinateur Pentium ® Dual Core CPU T4500 cadencé à 2.30 GHz et 2 GB de RAM.

3.3.2.1 Réglage de paramètres

Nous avons considéré quatre valeurs possibles pour le nombre d'itérations (50, 100, 200, 500), cinq valeurs pour *popsize* (15, 20, 30, 40, 50) et trois valeurs pour p_a (20%, 25%, 30%). Nous avons essayé toutes les combinaisons possibles. Les résultats ont montré que les valeurs suivantes semblent appropriées pour trouver les meilleures solutions tout en consommant un temps de calcul raisonnable :

— Taille de la population ou le nombre de nids : $popsize = 50$;

— Nombre d'itérations : 200 ;

— Fraction des nids à abandonner : 25% ;

3.3.2.2 Résultats

La performance de notre méthode HCuckooSA1 est évaluée sur 35 instances de référence. Dix exécutions distinctes de l'algorithme, avec des germes (*seed*) différents, sont effectuées pour chaque instance avec les paramètres mentionnés précédemment.

Le Tableau 3.1 présente le nombre de machines, de produits et de cellules de chaque instance, ainsi que l'efficacité des meilleures solutions connues dans la littérature et l'efficacité des solutions minimales, moyennes et maximales atteintes par notre méthode durant dix exécutions. De plus, la valeur de l'écart (%gap) et le temps de calcul requis en secondes sont rapportés dans ce tableau, respectivement.

La valeur de l'écart (%gap) dépend de l'efficacité de la meilleure solution trouvée dans la littérature et l'efficacité de la solution maximale obtenue par notre méthode tel que présenté par l'Equation (3.6).

$$\%gap = \frac{Eff\_Best - Eff\_Max}{Eff\_Best} \tag{3.6}$$

D'après les tests, HCuckooSA1 a pu atteindre l'efficacité des meilleures solutions connues pour 31 instances dont les résultats sont stables pour 17 instances au cours des 10 exécutions de l'algorithme. Concernant les instances restantes, les efficacités des solutions moyennes sont très proches de celles des meilleures solutions connues.

À propos du temps de calcul requis pour trouver les solutions, les 11 premières instances sont résolues en moins d'une seconde et 11 autres instances sont résolues en moins de dix secondes. En ce qui concerne les autres instances, le temps de calcul accroît en fonction de la taille de l'instance. Les résultats obtenus montrent que la méthode proposée jouit d'une performance élevée pour trouver de très bonnes solutions en consommant un temps de calcul raisonnable.

Pour confirmer l'efficacité de notre méthode, le Tableau 3.2 rapporte les résultats de comparaison des meilleures efficacités obtenues par la méthode HCuckooSA1 et par six autres méthodes développées dans la littérature :

— ZODIAC : Méthode de clustering non-hiérarchique (Chandrasekharan et Rajagopalan, 1987)

— GRAFICS : Méthode de clustering non-hiérarchique (Srinivasan et Narendran, 1991)

— EA : Algorithme évolutionnaire (Gonçalves et Resende, 2004)

— SA : Recuit simulé (Ying *et al.*, 2011)

— GRASP : Procédure de recherche gloutonne aléatoire adaptative (Díaz *et al.*, 2012)

— GA_LNS : Algorithme génétique (Elbenani *et al.*, 2012)

TABLE 3.1 – HCuckooSA1 : Résultats expérimentaux pour 35 instances de référence du CFP

No.	M	P	C	Eff_Best	Eff_Min	Eff_Avg	Eff_Max	Gap(%)	Time(s)
1	5	7	2	**82.35**	82.35	82.35	**82.35**	0	0.32
2	5	7	2	**69.57**	69.57	69.57	**69.57**	0	0.33
3	5	18	2	**79.59**	79.59	79.59	**79.59**	0	0.35
4	6	8	2	**76.92**	76.92	76.92	**76.92**	0	0.32
5	7	11	5	**60.87**	60.87	60.87	**60.87**	0	0.57
6	7	11	4	**70.83**	70.83	70.83	**70.83**	0	0.46
7	8	12	4	**69.44**	69.44	69.44	**69.44**	0	0.48
8	8	20	3	**85.25**	85.25	85.25	**85.25**	0	0.52
9	8	20	2	**58.72**	58.72	58.72	**58.72**	0	0.40
10	10	10	5	**75.00**	75.00	75.00	**75.00**	0	0.65
11	10	15	3	**92.00**	92.00	92.00	**92.00**	0	0.51
12	14	24	7	**72.06**	71.64	72.02	**72.06**	0	3.70
13	14	24	7	**71.83**	69.44	71.51	**71.83**	0	4.30
14	16	24	8	**53.26**	53.26	53.26	**53.26**	0	5.79
15	16	30	6	**69.53**	67.42	68.92	**69.53**	0	5.20
16	16	43	8	**57.53**	56.25	56.80	**57.53**	0	14.26
17	18	24	9	**57.73**	55.67	57.21	**57.73**	0	8.45
18	20	20	5	**43.45**	41.61	42.59	**43.45**	0	2.89
19	20	23	7	**50.81**	48.09	50.25	**50.81**	0	6.38
20	20	35	5	**77.91**	77.91	77.91	**77.91**	0	5.04
21	20	35	5	**57.98**	57.14	57.89	**57.98**	0	5.12
22	24	40	7	**100.0**	100.0	100.0	**100.0**	0	14.35
23	24	40	7	**85.11**	85.11	85.11	**85.11**	0	14.19
24	24	40	7	**73.51**	73.51	73.51	**73.51**	0	14.65
25	24	40	11	**53.29**	52.74	53.13	**53.29**	0	38.39
26	24	40	12	**48.95**	47.95	48.19	48.61	0.69	44.32
27	24	40	12	**47.26**	43.84	45.49	46.21	2.22	43.86
28	27	27	5	**54.82**	54.82	54.82	**54.82**	0	5.23
29	28	46	10	**47.23**	44.87	46.45	47.06	0.36	50.68
30	30	41	14	**63.31**	62.42	62.92	**63.31**	0	83.68
31	30	50	13	**60.12**	58.29	59.31	59.77	0.58	93.43
32	30	50	14	**50.83**	49.72	50.46	**50.83**	0	85.97
33	36	90	17	**47.75**	45.66	47.08	**47.75**	0	532.59
34	37	53	3	**60.64**	59.26	59.97	**60.64**	0	9.58
35	40	100	10	**84.03**	84.03	84.03	**84.03**	0	248.31

Le Tableau 3.3 résume les résultats de comparaison entre ces méthodes. La performance de notre méthode HCuckooSA1 surpasse celles de ZODIAC, GRAFICS, EA et GRASP par la résolution de 88.57% des instances. De plus, HCuckooSA1 et GA-LNS ont réussi à résoudre le même nombre des instances ; tandis que la méthode SA a pu atteindre la meilleure solution connue pour 32 instances.

Pour évaluer la performance de notre méthode, nous avons calculé et comparé la valeur du gap des meilleures solutions obtenues par les trois méthodes qui ont résolu le plus grand nombre des solutions : SA, GA-LNS et HCuckooSA1, comme le montre la Figure 3.4.

TABLE 3.2 – HCuckooSA1 : Performance de la méthode HCuckooSA1 par rapport à d'autres méthodes

No.	ZODIAC	GRAFICS	EA	SA	GRASP	GA-LNS	HCuckooSA1	Best Sol.
1	73.68	73.68	73.68	**82.35**	73.68	**82.35**	**82.35**	**82.35**
2	56.22	60.87	62.50	**69.57**	62.50	**69.57**	**69.57**	**69.57**
3	-	-	**79.59**	**79.59**	**79.59**	**79.59**	**79.59**	**79.59**
4	-	-	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**
5	39.13	53.12	53.13	**60.87**	53.13	**60.87**	**60.87**	**60.87**
6	-	-	70.37	**70.83**	70.37	**70.83**	**70.83**	**70.83**
7	68.30	68.30	68.29	**69.44**	**69.44**	**69.44**	**69.44**	**69.44**
8	85.24	85.24	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**
9	58.33	58.33	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**
10	70.59	70.59	70.59	**75.00**	70.59	**75.00**	**75.00**	**75.00**
11	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**
12	64.36	64.36	69.86	**72.06**	69.86	**72.06**	**72.06**	**72.06**
13	65.55	65.55	69.33	**71.83**	69.33	**71.83**	**71.83**	**71.83**
14	32.09	45.52	52.58	**53.26**	51.96	**53.26**	**53.26**	**53.26**
15	67.83	67.83	67.83	68.99	67.83	**69.53**	**69.53**	**69.53**
16	53.76	54.39	54.86	**57.53**	56.52	**57.53**	**57.53**	**57.53**
17	41.84	48.91	54.46	**57.73**	54.46	**57.73**	**57.73**	**57.73**
18	21.63	38.26	42.94	**43.45**	42.96	**43.45**	**43.45**	**43.45**
19	38.96	49.36	49.65	**50.81**	49.65	**50.81**	**50.81**	**50.81**
20	75.14	75.14	76.22	**77.91**	76.54	**77.91**	**77.91**	**77.91**
21	-	-	58.07	**57.98**	58.15	**57.98**	**57.98**	**57.98**
22	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
23	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**
24	37.85	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**
25	20.42	43.27	51.88	**53.29**	51.97	**53.29**	**53.29**	**53.29**
26	18.23	44.51	46.69	**48.95**	47.37	**48.95**	48.61	**48.95**
27	17.61	41.67	44.75	**47.26**	44.87	46.58	46.21	**47.26**
28	52.14	47.37	54.27	**54.82**	54.27	**54.82**	**54.82**	**54.82**
29	33.01	32.86	44.37	**47.23**	46.06	47.06	47.06	**47.23**
30	33.46	55.43	58.11	**63.31**	59.52	63.12	**63.31**	**63.31**
31	46.06	56.32	59.21	59.77	60.00	**60.12**	59.77	**60.12**
32	21.11	47.96	50.48	**50.83**	50.51	**50.83**	**50.83**	**50.83**
33	32.73	39.41	42.12	45.93	47.14	**47.75**	**47.75**	**47.75**
34	52.21	52.21	56.42	**60.64**	59.85	60.63	**60.64**	**60.64**
35	83.92	83.92	**84.03**	**84.03**	**84.03**	**84.03**	**84.03**	**84.03**

Nous avons trouvé que SA a le plus grand écart (gap) au niveau de l'instance 33 suivie par notre méthode HCuckooSA1 et GA-LNS dans l'instance 27. Cependant, SA a enregistré des écarts dans trois instances seulement tandis que notre HCuckooSA1 et GA-LNS ont des écarts dans quatre instances. Ces résultats indiquent que notre méthode est prometteuse puisqu'elle est appliquée pour la première fois à ce type de problèmes.

Les tests réalisés ont montré que HCuckooSA1 est plus approprié pour résoudre le CFP et permet de fournir de bons résultats. La capacité d'exploiter et d'explorer l'espace de recherche grâce aux vols de Lévy et à la méthode de recherche utilisée rend notre méthode plus performante que la majorité des méthodes proposées pour résoudre le CFP.

TABLE 3.3 – HCuckooSA1 : résumé de comparaison entre plusieurs méthodes

	ZODIAC	GRAFICS	EA	SA	GRASP	GA-LNS	HCuckooSA1
Nombre de meilleures solutions	3/31	4/31	9/35	32/35	10/35	31/35	31/35
Nombre de meilleures solutions en termes de pourcentage (%)	9.68	12.90	25.71	91.43	28.57	88.57	88.57

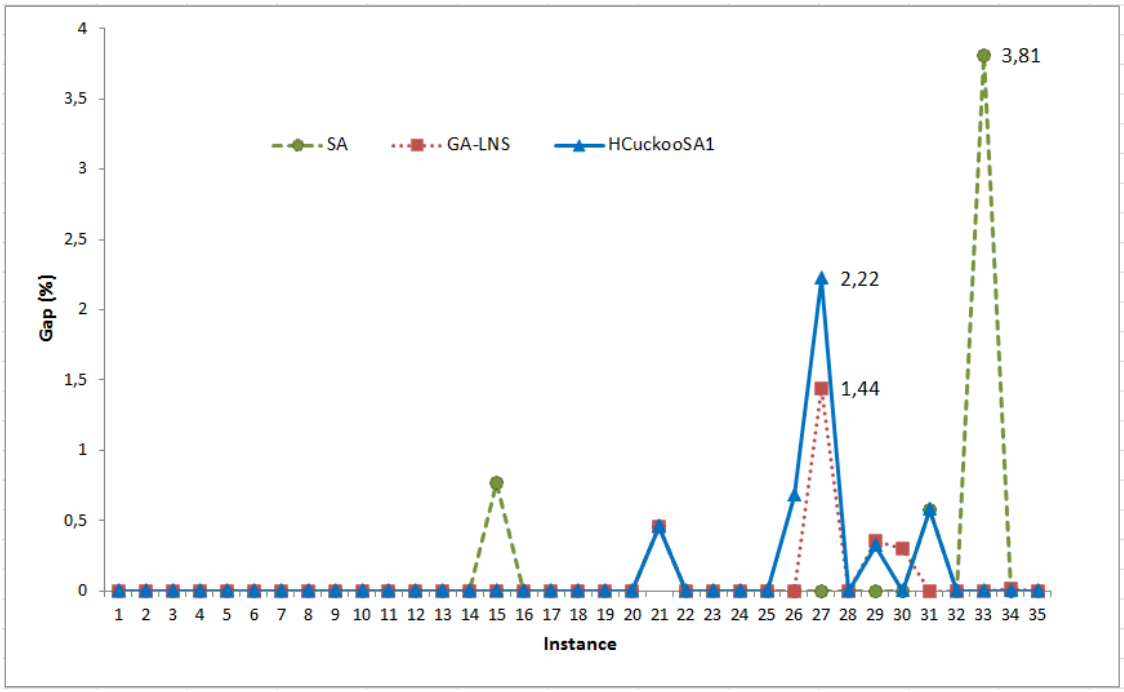


FIGURE 3.4 – HCuckooSA1 : Comparaion du Gap entre SA et HCuckooSA1 pour 35 instances

3.4 Deuxième forme : HCuckooSA2

3.4.1 Adaptation de HCuckooSA2 au CFP

Notre deuxième forme est notée HCuckooSA2 (the second Hybrid Cuckoo Search Algorithm-Cell Formation Problem). Dans cette forme, nous avons gardé la même structure des solutions que la première forme et nous avons introduit une nouvelle façon pour générer les nouvelles solutions en utilisant le vol de Lévy(Karoum et Elbenani, 2017c).

Étant donné que le vol de Lévy représente un modèle de marches aléatoires caractérisé par la longueur des pas ; ce pas représente la distance entre deux solutions. Il est basé généralement sur la topologie spatiale et le concept de voisinage dans l'algorithme de la recherche du coucou.
Pour adapter cet algorithme à notre problème de formation de cellules, nous avons remplacé la différence entre la position de la solution courante et la meilleure solution $(s_b^t - s_l^t)$ par leurs efficacités (*Eff*) comme montré dans l'Equation (3.7). Le but est d'être en mesure de s'approcher à la meilleure solution.

$$lévy(\beta) = 0.01\frac{\mu}{|\nu|^{1/\beta}}(Eff(s_b^{(t)}) - Eff(s_l^{(t)})),\ 1 < \beta \leq 3 \tag{3.7}$$

Où $Eff(s_b^{(t)})$ est l'efficacité du groupement de la meilleure solution trouvée au cours de l'itération t et $Eff(s_l^{(t)})$ est l'efficacité du groupement de la solution courante.

Le vol de Lévy va générer donc une valeur incluse dans l'intervalle $[0, 1]$. Le changement apporté à la solution courante dépend de la valeur du vol de Lévy. Une valeur faible de $lévy(\beta)$ indique que la solution est proche de la meilleure solution trouvée ; dans ce cas, le taux de changement sera minime. D'autre part, une valeur élevée de $lévy(\beta)$ montre que la solution est loin de la meilleure, alors des modifications majeures doivent être appliquées.
Nous avons divisé l'intervalle $[0, 1]$ en quatre intervalles pour décrire les quatre possibilités du mouvement. Si la valeur de Lévy est incluse dans l'intervalle :

— $[0, 0.25]$: Déplacer un produit ou une machine choisi de manière aléatoire de la cellule courante vers une autre cellule plus appropriée qui maximise l'efficacité du groupement. Par exemple, considérons une solution s d'une instance du problème avec 7 machines, 11 produits et 5 cellules.
$s = (4, 2, 5, \mathbf{1}, 2, 5, 3, 4, 4, 4, 1 \mid 3, 1, 2, 5, 2, 4, 5),\ Eff(s) = 40.74$
Le produit P_4 est sélectionné au hasard pour être déplacé de sa cellule (1) à l'une des 4 cellules possibles (2, 3, 4 et 5) qui améliore la fonction objective. Dans ce cas, le changement se fait en déplaçant le produit P_4 de la cellule 1 à la cellule 4 qui enregistre une meilleure efficacité comme illustré par s'.
$s' = (4, 2, 5, \mathbf{4}, 2, 5, 3, 4, 4, 4, 1 \mid 3, 1, 2, 5, 2, 4, 5),\ Eff(s') = 46.15$

— $[0.25, 0.5]$: Effectuer le déplacement de deux éléments (produits ou machines) choisis au hasard de leurs cellules courantes vers d'autres cellules plus appropriées. Par exemple, la solution s' est obtenue en effectuant un premier déplacement du produit P_4 de la cellule 1 vers la cellule 4 comme expliqué précédemment. Ensuite, une solution s" est obtenue à partir de s' en effectuant un deuxième déplacement du produit P_2 de la cellule 2 vers la cellule 3.
$s = (4, 2, 5, \mathbf{1}, 2, 5, 3, 4, 4, 4, 1 \mid 3, 1, 2, 5, 2, 4, 5),\ Eff(s) = 40.74$
$s' = (4, \underline{2}, 5, \mathbf{4}, 2, 5, 3, 4, 4, 4, 1 \mid 3, 1, 2, 5, 2, 4, 5),\ Eff(s') = 46.15$
s" $= (4, \underline{3}, 5, 4, 2, 5, 3, 4, 4, 4, 1 \mid 3, 1, 2, 5, 2, 4, 5),\ Eff(s") = 54.17$

— $[0.5, 0.75]$: Échanger les cellules entre deux produits ou deux machines ou entre un produit et une machine. La solution s' suivante est obtenue en échangeant les cellules 2 et 1 entre le produit P_7 et la machine M_5.
$s = (1, 3, 3, 3, 2, 5, \mathbf{2}, 4, 1, 1, 2 \mid 3, 2, 1, 4, \mathbf{1}, 5, 3),\ Eff(s) = 21.21$
$s' = (1, 3, 3, 3, 2, 5, \mathbf{1}, 4, 1, 1, 2 \mid 3, 2, 1, 4, \mathbf{2}, 5, 3),\ Eff(s') = 25.81$

— $[0.75, 1]$: Étant donné deux positions i et j choisies au hasard dans l'une des deux sections (produits ou machines), le mouvement consiste à inverser les éléments de la liste obtenue entre les positions i à j. La solution s' suivante est obtenue en inversant les éléments situés entre 6 et 10 dans la section des produits.
$s = (5, 1, 4, 1, 4, \underline{4, 3, 3, 4, 2}, 3 \mid 5, 4, 3, 2, 1, 5, 2),\ Eff(s) = 9.09$
$s' = (5, 1, 4, 1, 4, \underline{2, 4, 3, 3, 4}, 3 \mid 5, 4, 3, 2, 1, 5, 2),\ Eff(s') = 16.13$

3.4.2 Résultats expérimentaux

La performance de la méthode HCuckooSA2 est testée sur 35 instances de référence. Le langage Java est utilisé pour le codage de HCuckooSA2. Les tests sont effectués sur un ordinateur Pentium ® Dual Core CPU T4500 cadencé à 2.30 GHz et 2 GB de RAM.

3.4.2.1 Réglage de paramètres

Trois paramètres peuvent affecter la performance de la méthode proposée : la taille de la population *popsize*, le nombre maximal d'itérations et le pourcentage de nids à abandonner p_a. Durant nos tests, nous avons effectué plusieurs études statistiques pour déterminer les paramètres convenables à notre méthode.

La première étude est basée sur l'analyse de la variance (ANOVA) en utilisant le logiciel SPSS. Les résultats de l'étude statistique sont résumés dans une table, connue par la table d'ANOVA. Cette table affiche, pour chaque facteur, son degré de liberté (DF) ; la somme des carrés (SS) ; le carré moyen (MS) ; sa valeur de test de Fisher (F) et sa probabilité (P).

Le degré de liberté présente le nombre maximum de valeurs telles qu'aucune d'entre elles n'est calculable à partir des autres. Le test de Fisher, ou test F, permet de tester l'égalité de deux variances et il peut être exprimé en pourcentage. Par exemple un F de 95% ou 0,95 signifie que les deux variances sont très proches. La valeur de F est basée sur le degré de liberté et la somme des carrés.

D'après ANOVA, un facteur a un effet sur la fonction objectif s'il obtient une petite valeur de la probabilité P et une grande valeur de F. Dans la plupart des cas, si la probabilité P d'un facteur est inférieure à 0.05, ce facteur influence la fonction objectif. Pour plus de détails sur ANOVA, on peut référer le lecteur au (Muller et Fetterman, 2002).

Étant donné que les instances de petite taille sont faciles à résoudre, deux instances typiques ont été choisies pour l'analyse statistique : une instance de taille moyenne (l'instance 27 : 24 machines et 40 produits) et une instance de grande taille (l'instance 33 : 36 machines et 90 produits).

a) Analyse pour des instances de taille moyenne :

D'après le Tableau 3.4, la probabilité $P(p_a)$ est inférieure à 0.05. Nous concluons que p_a affecte l'efficacité de la solution. De plus, $P(popsize * p_a) < 0.05$ donc il y a une interaction entre *popsize* et p_a. Pour déterminer les valeurs appropriées des paramètres, nous devrions commencer par le paramètre ayant une valeur plus élevée de F. Selon la Figure 3.5, les valeurs convenables de p_a et *popsize* sont 0.2 (ou 20%) et 50, respectivement.

TABLE 3.4 – HCuckooSA2 : Table d'ANOVA pour une instance de taille moyenne

Source	DF	SS	MS	F	P
popsize	3	0.686	0.229	0.752	0.523
p_a	2	2.004	1.002	3.296	0.041
$popsize * p_a$	6	5.500	0.917	3.016	0.009
Error	108	32.830	0.304		
Total	120	244674.557			

DF : Degré de liberté ; SS : somme des carrés ; MS : carré moyen ; F : Test de Fisher ; P : probabilité

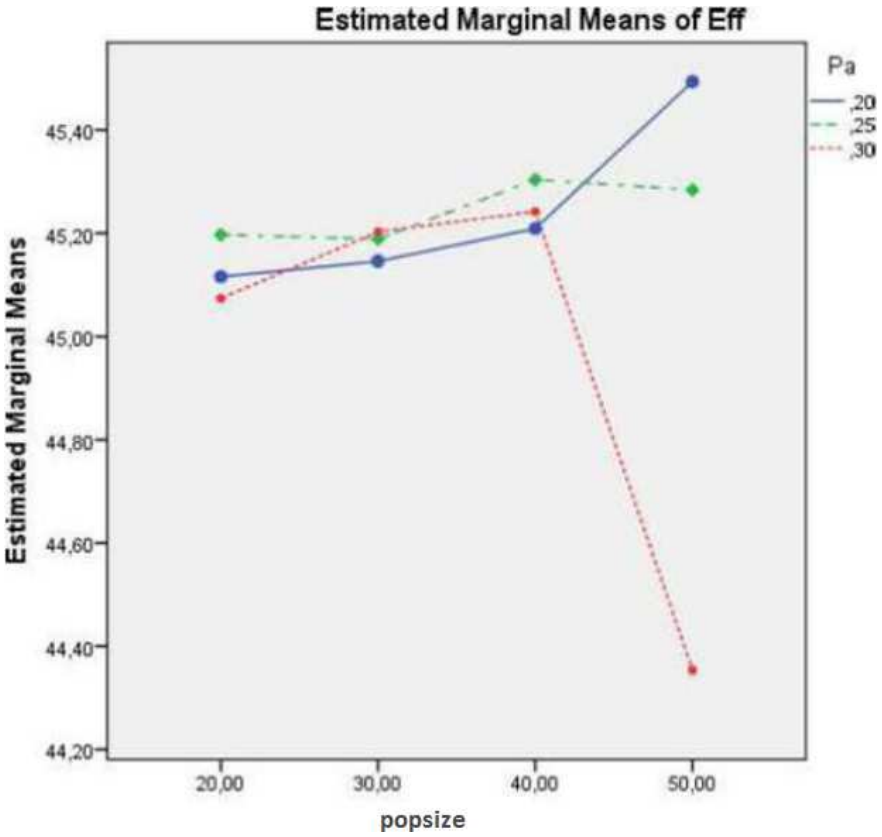


FIGURE 3.5 – HCuckooSA2 : Etude statistique d'ANOVA pour une instance de taille moyenne

b) Analyse pour des instances de grande taille :

Les résultats de l'étude statistique pour une instance de grande taille sont présentés dans le Tableau 3.5. Nous remarquons que les probabilités $P(p_a)$ et $P(popsize)$ sont inférieures à 0.05. En se basant sur la Figure 3.6, les valeurs $p_a = 0.2$ et $popsize = 50$ devraient être sélectionnées pour une meilleure performance de l'algorithme.

TABLE 3.5 – HCuckooSA2 : Table d'ANOVA pour une instance de grande taille

Source	DF	SS	MS	F	P
$popsize$	3	3.284	1.095	4.024	0.009
p_a	2	2.254	1.127	4.142	0.018
$popsize * p_a$	6	1.093	0.182	0.670	0.674
Error	108	29.378	0.272		
Total	120	265636,083			

DF : Degree of Freedom ; SS : Sum of Squares ; MS : Mean Square ; F : Fisher's test ; P : Probability

Pour confirmer que la combinaison des paramètres ($popsize = 50$ et $p_a = 0.2$) peut convenir à toute autre instance de référence, une deuxième étude est effectuée, où les différentes combinaisons de $popsize$ et p_a sont testées sur les 35 instances de référence. Nous avons exécuté la méthode HCuckooSA2 dix fois pour chaque instance. L'efficacité des solutions minimales, moyennes, et maxi-

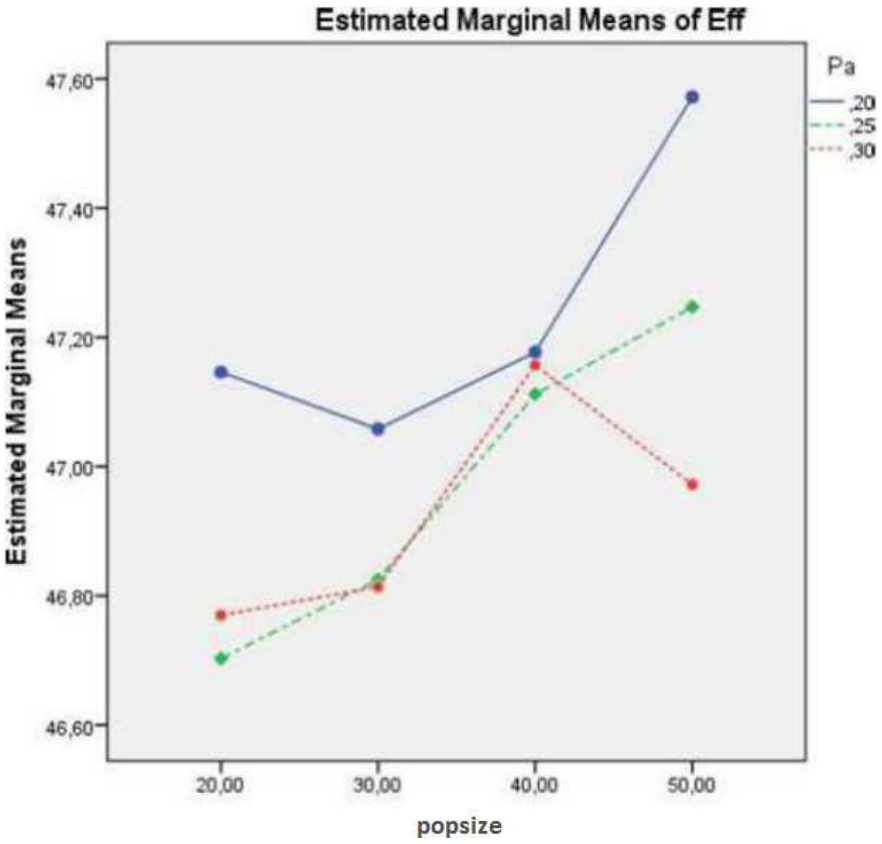


FIGURE 3.6 – HCuckooSA2 : Etude statistique d'ANOVA pour une instance de grande taille

males obtenues sont représentées dans le Tableau 3.6. Le nombre total trouvé des solutions les plus connues (Nb of BKS) pour chaque combinaison est également listé dans ce tableau. Les résultats ont montré que des meilleures performances sont obtenues lorsque $p_a = 0.2$ et $popsize = 50$.

TABLE 3.6 – HCuckooSA2 : Résultats des différentes combinaisons de paramètres

Combinaison	p_a	*popsize*	Eff_Min	Eff_Avg	Eff_Max	Nb of BKS
1	0.2	20	65.46	65.73	65.90	30
2	0.25	20	65.34	65.70	65.90	29
3	0.3	20	65.54	65.75	65.89	29
4	0.2	30	65.60	65.79	65.91	31
5	0.25	30	65.56	65.77	65.92	29
6	0.3	30	65.54	65.76	65.93	30
7	0.2	40	65.62	65.80	65.92	31
8	0.25	40	65.61	65.80	65.91	30
9	0.3	40	65.62	65.78	65.90	30
10	**0.2**	**50**	**65.69**	**65.83**	**65.94**	**32**
11	0.25	50	65.63	65.80	65.92	31
12	0.3	50	65.60	65.79	65.88	28

3.4.2.2 Résultats

Tout d'abord, nous allons évaluer la méthode sans et avec le mécanisme de recherche locale. Les deux méthodes seront notées par CuckooSA2 et HCuckooSA2, respectivement. Nous avons réalisé dix exécutions pour chaque instance avec les paramètres mentionnés précédemment. Le Tableau 3.7 présente le nombre de machines, le nombre de produits et le nombre de cellules de chaque instance. Il présente aussi l'efficacité des solutions minimales, moyennes et maximales atteintes et le temps de calcul obtenu en secondes. De plus, la valeur de l'écart (%gap) (Equation (3.6)) est rapportée dans ce tableau.

TABLE 3.7 – HCuckooSA2 : Résultats sans et avec la méthode de recherche locale

No.	M	P	C	CuckooSA2				HCuckooSA2				Best	$Gap(\%)$
				Eff_Min	Eff_Avg	Eff_Max	Time(s)	Eff_Min	Eff_Avg	Eff_Max	Time(s)		
1	5	7	2	**82.35**	**82.35**	**82.35**	0.26	**82.35**	**82.35**	**82.35**	0.44	**82.35**	0
2	5	7	2	**69.57**	**69.57**	**69.57**	0.28	**69.57**	**69.57**	**69.57**	0.40	**69.57**	0
3	5	18	2	**79.59**	**79.59**	**79.59**	0.34	**79.59**	**79.59**	**79.59**	0.48	**79.59**	0
4	6	8	2	**76.92**	**76.92**	**76.92**	0.35	**76.92**	**76.92**	**76.92**	0.50	**76.92**	0
5	7	11	5	**60.87**	**60.87**	**60.87**	1.21	**60.87**	**60.87**	**60.87**	1.54	**60.87**	0
6	7	11	4	**70.83**	**70.83**	**70.83**	0.74	**70.83**	**70.83**	**70.83**	1.15	**70.83**	0
7	8	12	4	**69.44**	**69.44**	**69.44**	0.95	**69.44**	**69.44**	**69.44**	1.41	**69.44**	0
8	8	20	3	**85.25**	**85.25**	**85.25**	0.90	**85.25**	**85.25**	**85.25**	1.28	**85.25**	0
9	8	20	2	**58.72**	**58.72**	**58.72**	0.50	**58.72**	**58.72**	**58.72**	0.68	**58.72**	0
10	10	10	5	**75.00**	**75.00**	**75.00**	1.77	**75.00**	**75.00**	**75.00**	1.89	**75.00**	0
11	10	15	3	**92.00**	**92.00**	**92.00**	1.02	**92.00**	**92.00**	**92.00**	1.29	**92.00**	0
12	14	24	7	67.65	69.65	**72.06**	5.44	**72.06**	**72.06**	**72.06**	14.57	**72.06**	0
13	14	24	7	67.65	70.24	**71.83**	5.54	**71.83**	**71.83**	**71.83**	14.92	**71.83**	0
14	16	24	8	48.45	49.43	51.06	7.12	**53.26**	**53.26**	**53.26**	25.54	**53.26**	0
15	16	30	6	53.19	59.5	65.65	2.53	**69.53**	**69.53**	**69.53**	18.24	**69.53**	0
16	16	43	8	35.29	38.4	40.38	2.98	57.14	57.47	**57.53**	43.45	**57.53**	0
17	18	24	9	49	51.19	53.54	8.27	**57.73**	**57.73**	**57.73**	38.71	**57.73**	0
18	20	20	5	39.31	40.34	41.01	3.36	42.36	42.81	**43.45**	12.64	**43.45**	0
19	20	23	7	42.96	44.43	45.52	3.97	50.4	50.76	**50.81**	28.68	**50.81**	0
20	20	35	5	47.09	51.51	56.52	1.90	**77.91**	**77.91**	**77.91**	19.24	**77.91**	0
21	20	35	5	36.48	39.78	41.71	1.92	**57.98**	**57.98**	**57.98**	18.93	**57.98**	0
22	24	40	7	40.72	44.22	48.19	4.27	**100.0**	**100.0**	**100.0**	40.98	**100.0**	0
23	24	40	7	36.1	38.41	41.71	4.13	**85.11**	**85.11**	**85.11**	45.83	**85.11**	0
24	24	40	7	32.51	34.42	35.89	4.04	**73.51**	**73.51**	**73.51**	48.70	**73.51**	0
25	24	40	11	30	30.67	32.1	10.23	52.74	53.11	**53.29**	137.98	**53.29**	0
26	24	40	12	29.59	30.36	31.01	12.26	47.62	48.12	**48.95**	165.25	**48.95**	0
27	24	40	12	29.27	30.14	31.52	12.08	44.83	45.49	46.21	173.55	**47.26**	2.22
28	27	27	5	39.3	41.67	45.48	1.96	**54.82**	**54.82**	**54.82**	23.13	**54.82**	0
29	28	46	10	23.66	25.03	26.44	9.53	46.75	46.9	47.08	159.35	**47.23**	0.32
30	30	41	14	29.17	30.85	31.82	18.84	62.76	63.11	**63.31**	336.20	**63.31**	0
31	30	50	13	24.3	24.79	25.91	17.74	58.38	59.40	59.77	324.65	**60.12**	0.58
32	30	50	14	23.08	23.73	24.31	20.80	50.00	50.49	**50.83**	397.58	**50.83**	0
33	36	90	17	14.62	15.35	16.97	44.81	47.37	47.57	**47.75**	991.51	**47.75**	0
34	37	53	3	39.46	40.46	42.14	1.44	**60.64**	**60.64**	**60.64**	22.34	**60.64**	0
35	40	100	10	13.53	13.87	14.27	21.05	**84.03**	**84.03**	**84.03**	329.63	**84.03**	0

Selon ce tableau, CuckooSA2 a atteint la meilleure solution connue pour 13 instances de référence. Quant aux instances restantes, les résultats obtenus sont proches des meilleures solutions connues. Cependant, nous avons introduit un mécanisme de recherche locale pour améliorer les résultats : HCuckooSA2. Les tests ont montré que HCuckooSA2 peut atteindre la meilleure solution connue pour 32 instances. Parmi ces instances, les résultats étaient stables pour 24 instances (1 à 15, 17, 20 à 24, 28, 34 et 35) au cours des 10 exécutions de l'algorithme (Eff_Min == Eff_Avg ==

Eff_Max), ce qui confirme la bonne performance de la méthode projetée. En outre, HCuckooSA2 a connu des déviations dans trois instances seulement où la plus élevée est égale à 2.22%. Ces résultats indiquent que notre méthode est prometteuse puisqu'elle est appliquée pour la première fois à ce type de problèmes. Concernant le temps de calcul nécessaire pour trouver la meilleure solution, les 11 premières instances sont résolues en moins de 2 secondes. La méthode a pu également atteindre la meilleure solution connue pour 15 instances importantes (12 à 24, 28 et 34) en moins de 50 secondes. Malheureusement, le temps de calcul augmente pour les instances restantes en raison de leur taille (M, P et C).

TABLE 3.8 – HCuckooSA2 : Résultats de la comparaison avec plusieurs méthodes

No.	SA	GA_LNS	HGBPSO	Dinkelbach	HCuckooSA2	Best
1	**82.35**	**82.35**	**82.35**	**82.35**	**82.35**	**82.35**
2	**69.57**	**69.57**	**69.57**	**69.57**	**69.57**	**69.57**
3	**79.59**	**79.59**	**79.59**	**79.59**	**79.59**	**79.59**
4	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**
5	**60.87**	**60.87**	**60.87**	**60.87**	**60.87**	**60.87**
6	**70.83**	**70.83**	**70.83**	**70.83**	**70.83**	**70.83**
7	**69.44**	**69.44**	**69.44**	**69.44**	**69.44**	**69.44**
8	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**
9	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**
10	**75.00**	**75.00**	**75.00**	**75.00**	**75.00**	**75.00**
11	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**
12	**72.06**	**72.06**	**72.06**	**72.06**	**72.06**	**72.06**
13	**71.83**	**71.83**	**71.83**	**71.83**	**71.83**	**71.83**
14	**53.26**	**53.26**	53.41	**53.26**	**53.26**	**53.26**
15	68.99	**69.53**	68.99	**69.53**	**69.53**	**69.53**
16	**57.53**	**57.53**	**57.53**	**57.53**	**57.53**	**57.53**
17	**57.73**	**57.73**	**57.73**	**57.73**	**57.73**	**57.73**
18	**43.45**	**43.45**	43.26	43.06	**43.45**	**43.45**
19	**50.81**	**50.81**	**50.81**	**50.81**	**50.81**	**50.81**
20	**77.91**	**77.91**	**77.91**	**77.91**	**77.91**	**77.91**
21	**57.98**	**57.98**	**57.98**	**57.98**	**57.98**	**57.98**
22	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
23	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**
24	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**
25	**53.29**	**53.29**	**53.29**	**53.29**	**53.29**	**53.29**
26	**48.95**	**48.95**	**48.95**	48.28	**48.95**	**48.95**
27	**47.26**	46.58	**47.26**	45.70	46.21	**47.26**
28	**54.82**	**54.82**	**54.82**	**54.82**	**54.82**	**54.82**
29	**47.23**	47.06	-	46.75	47.08	**47.23**
30	**63.31**	63.12	**63.31**	**63.31**	**63.31**	**63.31**
31	59.77	**60.12**	59.77	**60.12**	59.77	**60.12**
32	**50.83**	**50.83**	-	50.55	**50.83**	**50.83**
33	47.14	**47.75**	-	44.67	**47.75**	**47.75**
34	**60.64**	60.63	**60.64**	**60.64**	**60.64**	**60.64**
35	**84.03**	**84.03**	-	**84.03**	**84.03**	**84.03**
Nb Best	**32**	**31**	**27**	**29**	**32**	**35**

Pour exposer la performance de la méthode proposée, nous avons comparé les meilleures efficacités obtenues avec celles enregistrées par d'autres méthodes développées dans la littérature : le recuit simulé SA (Ying *et al.*, 2011), l'algorithme génétique GA_LNS (Elbenani *et al.*, 2012) et l'algorithme d'optimisation par essaims particulaires HGBPSO (Kashan *et al.*, 2014). De plus, nous avons utilisé une méthode exacte basée sur l'algorithme de Dinkelbach dans la phase de comparaison (Elbenani et Ferland, 2012). Le Tableau 3.8 présente les résultats de la comparaison. La dernière ligne fournit le nombre des meilleures solutions atteintes par chaque méthode.

Le résultat de la comparaison montre que SA et notre méthode sont classés les premiers en atteignant l'efficacité des meilleures solutions connues pour 32 instances. Concernant les trois instances restantes, les deux méthodes ont obtenu la même solution pour 31, qui est proche de la meilleure solution connue. De plus, la méthode proposée offre un meilleur résultat que SA pour l'instance 33. Cependant, SA fonctionne mieux dans 27 et 29. Mais, il faut noter que la méthode SA est appliquée plusieurs fois pour résoudre le problème de formation de cellules et elle a été finalisée pour donner des meilleurs résultats. Tandis que l'algorithme de recherche du coucou est un algorithme récemment développé et il est appliqué pour la première fois pour résoudre ce problème. Quant à l'algorithme de Dinkelbach, notre méthode a obtenu des meilleurs résultats pour les instances 18, 26, 27, 29, 32 et 33 en consommant moins de temps de calcul.

La Figure 3.7 présente les valeurs du gap calculées pour les quatre méthodes supérieures qui sont appliquées à toutes les instances. Notre méthode HCuckooSA2 a enregistré des écarts dans trois instances, le plus élevé est 2.22%. Ensuite, la méthode SA a présenté des écarts dans trois instances ; le plus élevé est enregistré par l'instance 33. La méthode GA_LNS a obtenu des écarts dans quatre instances et le plus élevé est égal à 1.44%. Enfin, l'algorithme de Dinkelbach a connu des écarts dans six instances ; la plus grande valeur concerne l'instance 33 et elle est égale à 6.45%.

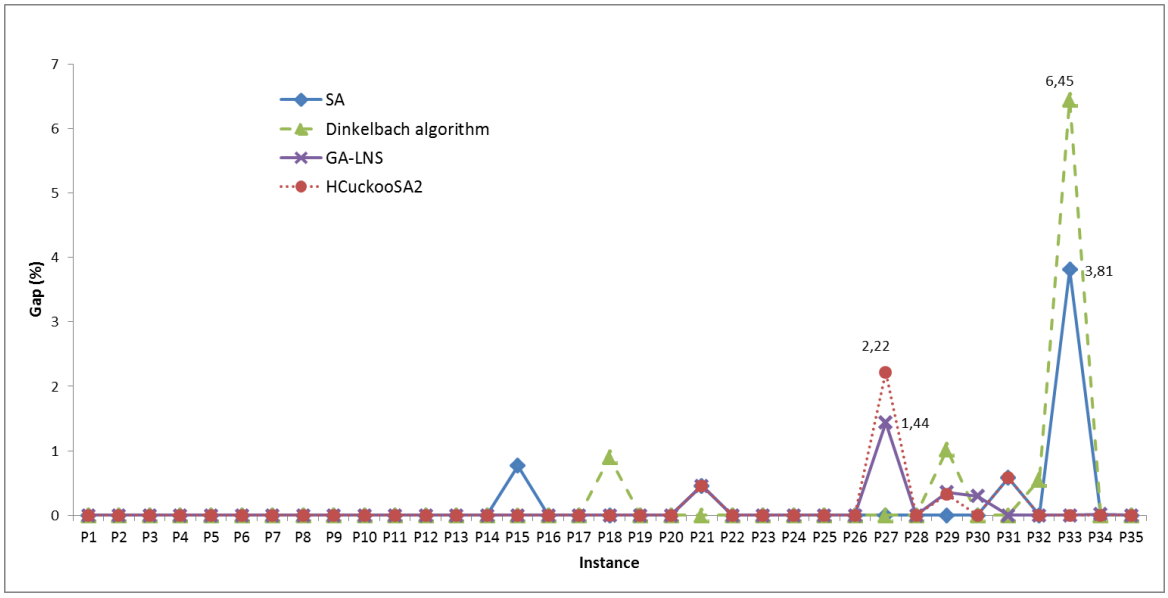


FIGURE 3.7 – HCuckooSA2 : Comparaison de gaps entre les méthodes

Les résultats numériques indiquent que notre méthode HCuckooSA2 génère de bons résultats pour les 35 instances de référence et que la performance de cette méthode est généralement constante en considérant la totalité des instances.

3.5 Conclusion

Dans ce chapitre, nous avons proposé deux méthodes basées sur l'algorithme de recherche du coucou dans le but de résoudre le problème de formation de cellules. La différence principale entre ces deux méthodes demeure dans la façon d'effectuer des déplacements dans l'espace de recherche via les vols de Lévy. Suite aux tests réalisés, nous constatons que les deux méthodes guident une exploitation intensifiée des régions prometteuses et une exploration de l'espace de recherche à l'échelle globale. La performance des méthodes proposées a été comparée avec plusieurs algorithmes de la littérature. Les résultats de comparaison montrent que nos méthodes surpassent la majorité de ces algorithmes lors de la résolution des instances du problème.

Chapitre

4 CFP avec cellules variables : Algorithme de sélection clonale

Sommaire

4.1 Introduction

Le nombre de cellules joue un rôle important dans la résolution du problème de formation de cellules. Dans les deux chapitres précédents, le nombre de cellules est prédéfini en tant que paramètre d'entrée. Ce nombre a été déterminé dans la littérature par des méthodes exactes. A travers ce chapitre, nous proposons une nouvelle approche basée sur l'algorithme de sélection clonale où le nombre de cellules n'est pas fixé à l'avance (Karoum et Elbenani, 2017a). Nous étudions le paramétrage de la méthode et son adaptation à la nouvelle version du problème. Ensuite, nous comparons les résultats obtenus avec d'autres algorithmes collectés de la littérature.

La méthode proposée est notée HClonalSA-VNC (Hybrid Clonal Selection Algorithm-Variant Number of Cells), et elle est principalement basée sur HClonalSA1 (Karoum *et al.*, 2016a). Les lecteurs peuvent trouver des ressemblances entre ces deux méthodes proposées ; Cependant, il existe

des différences importantes. Tout d'abord, le nombre de cellules est variable, ce qui peut affecter directement les mouvements intercellulaires des produits et les coûts de manutention. Deuxièmement, pour satisfaire à cette contrainte du nombre variable des cellules, notre représentation des solutions doit permettre l'ajout et la suppression des cellules. Par conséquent, un nouveau processus de réparation des solutions doit être appliqué. Pour renforcer la recherche, nous avons utilisé une méthode de recherche locale basée sur une stratégie de destruction pour mieux construire des solutions. Concernant le modèle mathématique utilisé pour évaluer la qualité de la solution, nous avons gardé le même modèle ($M_1(X, Y)$) utilisé dans le cas de la résolution du problème CFP avec un nombre fixe de cellules. La différence majeure entre les deux variantes du problème (cellules fixes et variables) réside dans la possibilité d'ajouter ou de supprimer des cellules.

4.2 Etat d'art

Un nombre limité d'auteurs ont étudié le cas d'un nombre variable de cellules lors de la résolution du problème de formation de cellules de production. Nous pouvons citer par exemple :

Brown et Sumichrast (2001) qui ont présenté une méthode basée sur un algorithme de regroupement génétique (grouping genetic algorithm) pour former les cellules de production et ils ont intégré une heuristique de remplacement au niveau de l'opérateur de croisement. Pour évaluer la performance de cette méthode, les auteurs ont utilisé les deux mesures les plus connues dans la littérature : l'efficience de groupement et l'efficacité de groupement.

James *et al.* (2007) ont également développé une méthode hybride basée sur un algorithme de groupement génétique (HGGA). Ils ont intégré la procédure de recherche locale définie par Gonçalves et Resende (2004). La méthode suggérée a pu augmenter l'efficacité des solutions et surpasser plusieurs méthodes bien connues, y compris les algorithmes de groupement génétiques classiques.

Wu *et al.* (2008) ont utilisé la méthode du recuit simulé pour minimiser le nombre des éléments exceptionnels et des éléments vides. Les cellules des produits sont formées par la construction d'une matrice de similarité entre les produits. La technique de recherche est guidée par un mouvement à la fois et un mouvement par échange afin de converger vers l'optimalité. Bien que cette méthode soit capable de trouver le nombre de cellules qui peut conduire à la meilleure solution, le processus de recherche itérative de la taille des cellules consomme trop de temps d'exécution.

Tunnukij et Hicks (2009) ont présenté une version améliorée d'un algorithme de regroupement génétique proposé par (Brown et Sumichrast, 2001). Leur amélioration consiste à remplacer l'heuristique de remplacement par un algorithme glouton (GH) et à utiliser une stratégie de roulette-élitiste à base de rangs (*rank-based roulette–elitist strategy*), qui est considérée comme un nouveau mécanisme pour créer des générations successives.

Noktehdan *et al.* (2010) ont proposé une version de groupement de l'algorithme d'évolution différentielle (GDE) pour résoudre le CFP où le nombre de cellules n'est pas prédéfini. De plus, ils ont combiné cet algorithme avec une méthode de recherche locale (HGDE) et ils ont utilisé un mécanisme analogue à celle des mécanismes de mutation.

Paydar et Saidi-Mehrabad (2013) ont développé un modèle de programmation mathématique qui permet d'identifier le nombre requis de cellules tout en maximisant l'efficacité du groupement comme fonction objectif. Pour résoudre le modèle proposé, les auteurs ont appliqué une méthode combinant un algorithme génétique et une approche de recherche à voisinage variable.

(Kashan *et al.*, 2014) ont présenté une nouvelle approche basée sur l'algorithme PSO (HGBPSO). Contrairement aux méthodes traditionnelles basées sur PSO et qui utilisent les opérateurs arith-

métiques, la version introduite par ces auteurs utilise des opérateurs de groupement appliqués aux cellules plutôt que sur les produits et les machines.

Noktehdan *et al.* (2016) ont modifié un algorithme du championnat de la ligue pour résoudre le problème (GLCA). L'algorithme proposé est basé sur une représentation de groupement utilisée dans un algorithme génétique. Une nouvelle méthode est également appliquée pour trouver le nombre de cellules initiales.

4.3 Adaptation de HClonalSA-VNC au CFP

Pour adapter HClonalSA-VNC à cette variante du problème, un nombre d'éléments doivent être spécifiés : la nouvelle représentation d'une solution, la procédure de génération d'une solution initiale, les opérateurs de mutation et de croisement appliqués et la méthode de recherche locale utilisée.

4.3.1 Représentation d'une solution

Dans HClonalSA-VNC, nous avons choisit de représenter notre solution sous forme d'un vecteur à 3 parties pour prendre en considération le nombre variable des cellules. Cette représentation se compose de trois sections : une section de produits ; une section de machines et une section de cellules. La même notation est déjà utilisée dans la littérature par (James *et al.*, 2007) :

$$s = (P_1, \ldots, P_P | M_1, \ldots, M_M | C_1, \ldots, C_C)$$

Où :

P_j est l'indice de la cellule contenant le produit $j, j = 1, \ldots, P$;
M_i est l'indice de la cellule comprenant la machine i, $i = 1, \ldots, M$;
C_k est le numéro de la cellule $k, k = 1, \ldots, C$ avec C un paramètre variable à ajuster.

La longueur des sections de produits et de machines est constante pour toutes les solutions et elle dépend de la taille du problème. D'autre part, la longueur de la section des cellules peut varier en fonction du nombre de cellules dans lesquelles les produits et les machines sont affectés. Après plusieurs tests, nous avons limité le nombre de cellules dans l'intervalle $[2, min(P, M)]$ avec P est le nombre de produits et M est le nombre de machines.

Exemple illustratif :

Prenons une instance ayant 11 produits et 7 machines. L'intervalle de C est donc $[2, 7]$. Deux solutions possibles s_1 et s_2 peuvent être générées par exemple :
s_1 $= (2, 2, 2, 1, 1, 1, 2, 1, 1, 1, 2 |\ 2, 2, 1, 1, 1, 1, 1 |\ 1, 2)$; le nombre de cellules dans s_1 est $C = 2$
s_2 $= (3, 2, 1, 1, 3, 3, 1, 1, 3, 2, 2 |\ 3, 2, 1, 1, 2, 3, 3 |\ 1, 2, 3)$; le nombre de cellules dans s_2 est $C = 3$

4.3.2 Génération d'une solution initiale

Pour générer une solution, nous choisissons un nombre c, aléatoirement, entre 2 et $min(P, M)$. Puis, nous affectons les produits et les machines aux c cellules en suivant les étapes introduites dans la Section 2.3.1.2 du Chapitre 2.

4.3.3 Mutation

Nous avons appliqué la mutation par échange (pair wise mutation) sur les clones des solutions choisies pour la sélection clonale (le pourcentage b de la population). Pour chaque solution, nous déterminons le nombre de ses clones en fonction de sa valeur d'affinité, la taille de la population et la somme des affinités de la population entière comme le montre l'Equation (2.2) du Chapitre 2. Ensuite, chaque clone subit un échange et le meilleur entre eux sera utilisé dans la phase de croisement. L'échange peut s'effectuer soit entre deux machines, entre deux produits ou entre un produit et une machine.

Considérons l'exemple présenté dans la Figure 4.1. En appliquant l'Equation (2.2), la solution courante est multipliée en deux clones. Chacun a subi la mutation par échange. Pour le premier clone, les cellules des deux produits P_1 et P_8 sont échangées ; l'efficacité ainsi enregistrée est de 24.49. Par contre, l'efficacité du clone 2, obtenue en échangeant les cellules du produit P_7 et de la machine M_6, a surpassé celles de la solution courante et du clone 1. Par conséquent, le deuxième clone muté sera utilisé dans la phase de croisement.

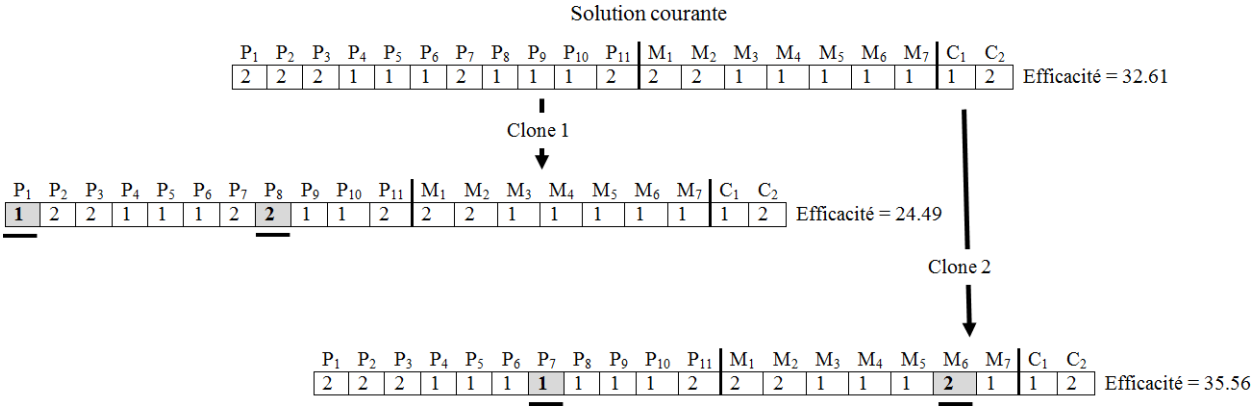


FIGURE 4.1 – HClonalSA-VNC : Exemple d'application du mutation par échange

4.3.4 Croisement uniforme

Nous avons appliqué le croisement uniforme sur les solutions obtenues par l'opérateur de mutation en considérant les sections de produits et de machines seulement.
Étant donné deux solutions parents :

$SP1 = (SP_1^1, ..., SP_P^1, SP_1^1, ..., SP_M^1)$
$SP2 = (SP_1^2, ..., SP_P^2, SP_1^2, ..., SP_M^2)$

Un vecteur de bits, appelé mask, ayant $(P + M)$ éléments est généré : $(S_1, ..., S_P, S_1, ..., S_M)$.
Les solutions enfants, $SE1 = (SE_1^1, ..., SE_P^1, SE_1^1, ..., SE_M^1)$ et $SE2 = (SE_1^2, ..., SE_P^2, SE_1^2, ..., SE_M^2)$, sont spécifiées comme suit :
Pour $j = 1, ..., P$:

$$\begin{cases} SE_j^1 = SP_j^1 \ et \ SE_j^2 = SP_j^2 & si \ S_j = 1 \\ SE_j^1 = SP_j^2 \ et \ SE_j^2 = SP_j^1 & si \ S_j = 0 \end{cases}$$

Pour $i = 1, ..., M$:

$$\begin{cases} SE_i^1 = SP_i^1 \ et \ SE_i^2 = SP_i^2 & si \ S_i = 1 \\ SE_i^1 = SP_i^2 \ et \ SE_i^2 = SP_i^1 & si \ S_i = 0 \end{cases}$$

Pour corriger les solutions non réalisables qui peuvent se produire suite à l'application des opérateurs génétiques, une heuristique de réparation semblable à celle proposée par (Brown et Sumichrast, 2001) est appliquée. Cette heuristique procède à l'élimination des cellules vides (cellules sans produits ou machines), puis à l'affectation des produits ou des machines de ces cellules vides aux cellules qui leurs conviennent le mieux.

Une de nos contributions apportées dans cette méthode réside dans la façon de déterminer la meilleure cellule :

— Si l'élément déplacé est un produit j, il sera attribué à une cellule C_j^* contenant le plus de machines possibles utilisant ce produit, tel que représenté dans l'Equation (4.1).

$$C_j^* = \underset{k}{\text{argmax}} \left\{ \frac{a_{1j}(k)}{a_{0j}(k) + a_{.j}} \right\} \tag{4.1}$$

Où :
argmax : Un argument qui maximise l'expression ;
$a_{1j}(k)$: Le nombre de machines dans la cellule k qui traite le produit j.
$a_{0j}(k)$: Le nombre de machines dans la cellule k qui ne traite pas le produit j.
$a_{.j}$: Le nombre total de machines qui traite le produit j.

— Si l'élément déplacé est une machine i, elle sera attribuée à une cellule C_i^* contenant le plus de produits possibles nécessitant cette machine, tel que indiqué dans l'Equation (4.2).

$$C_i^* = \underset{k}{\text{argmax}} \left\{ \frac{a_{i1}(k)}{a_{i0}(k) + a_{i.}} \right\} \tag{4.2}$$

Où :
$a_{i1}(k)$: Le nombre de produits dans la cellule k qui sont traités par la machine i.
$a_{i0}(k)$: Le nombre de produits dans la cellule k qui ne sont pas traités par la machine i
$a_{i.}$: Le nombre total de produits traités par la machine i.

Exemple explicatif :

Pour mieux comprendre le mécanisme utilisé, prenons l'exemple rapporté dans la Figure 4.2. Dans cet exemple, le croisement uniforme est appliqué sur les deux solutions (Parent 1 et Parent 2). La première solution (Parent 1) contient trois cellules et la deuxième solution (Parent 2) possède quatre cellules. Chaque cellule a au moins un produit et une machine :

Parent 1 (SP1) :

Cellules : $C_1 = \{\{P_2, P_3, P_{10}\}, \{M_2\}\}$
$C_2 = \{\{P_1, P_4, P_5, P_8, P_9\}, \{M_3, M_4, M_7\}\}$
$C_3 = \{\{P_6, P_7, P_{11}\}, \{M_1, M_5, M_6\}\}$

Parent 2 (SP2) :

Cellules : $C_1 = \{\{P_1, P_5, P_{11}\}, \{M_7\}\}$
$C_2 = \{\{P_6, P_7, P_{10}\}, \{M_1, M_3\}\}$
$C_3 = \{\{P_2, P_3, P_9\}, \{M_4, M_6\}\}$
$C_4 = \{\{P_4, P_8\}, \{M_2, M_5\}\}$

Deux solutions sont générées par cet opérateur du croisement (Enfant 1 et Enfant 2) :

Enfant 1 (SE1) :

Cellules : $C_1 = \{\{P_3, P_5\}, \{M_2, M_7\}\}$
$C_2 = \{\{P_1, P_4, P_6, P_7, P_8, P_{10}\}, \{M_1, M_3, M_4\}\}$
$C_3 = \{\{P_2, P_9, P_{11}\}, \{M_5, M_6\}\}$

Enfant 2 (SE2) :

Cellules : $C_1 = \{\{P_1, P_2, P_{10}, P_{11}\}, \{\}\}$
$C_2 = \{\{P_5, P_9\}, \{M_3, M_7\}\}$
$C_3 = \{\{P_3, P_6, P_7\}, \{M_1, M_4, M_6\}\}$
$C_4 = \{\{P_4, P_8\}, \{M_2, M_5\}\}$

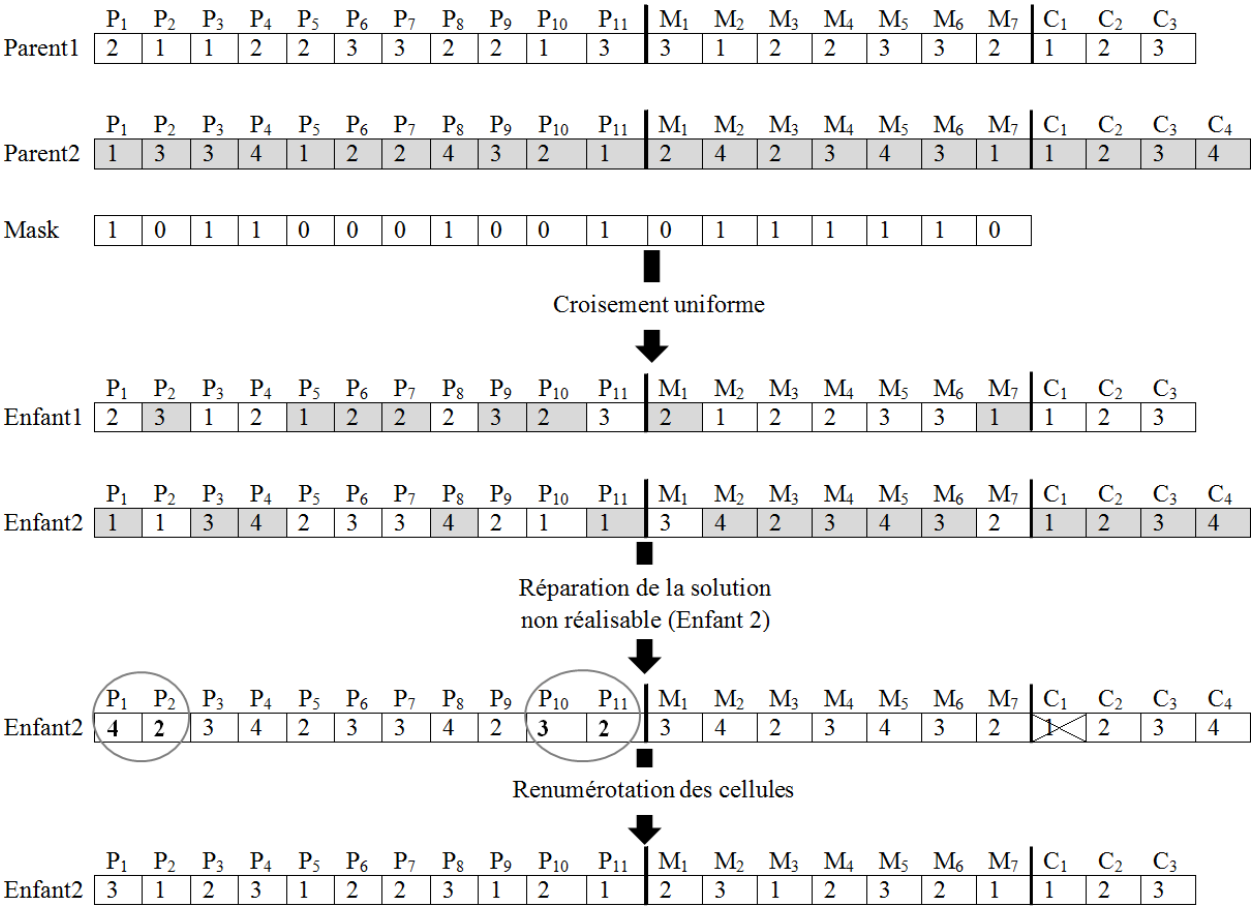


FIGURE 4.2 – HClonalSA-VNC : Exemple d'application du croisement uniforme

Nous pouvons remarquer que la deuxième solution est non réalisable du fait que la cellule 1 ne possède aucune machine pour traiter les produits. Dans ce cas, cette cellule sera supprimée et chacun de ces produits sera affecté à une cellule obtenue par l'application de l'Equation (4.1). En conséquence, le produit P_1 est affecté à la cellule 4 ; les produits P_2 et P_{11} sont déplacés vers la cellule 2 et le produit P_{10} est assigné à la cellule 3. Pour garantir une bonne représentation de la solution, les cellules sont renumérotées : la cellule 2 est renommée en cellule 1, la cellule 3 portera le numéro 2 et la cellule 4 recevra le numéro 3.

4.3.5 Recherche locale

Le mécanisme de recherche locale intégré dans HClonalSA-VNC est inspiré de (Elbenani *et al.*, 2012). Selon les cellules des machines de la solution initiale, chaque produit est affecté à la cellule convenable contenant le plus de machines l'utilisant (Equation (4.1)). Si la solution modifiée est meilleure que la solution initiale, elle la remplacera et le processus reprendra par la réaffectation des machines (une machine est affectée à la cellule contenant le plus de produits utilisant cette machine Equation (4.2)). Ce processus se poursuit par la réaffectation des produits, puis la réattribution des machines jusqu'à ce que la qualité de la nouvelle solution obtenue ne puisse être améliorée.

Cependant, si aucune modification n'est possible au niveau de la solution initiale, une stratégie de destruction et de récupération est appliquée pour modifier légèrement la solution. Cette stratégie consiste à détruire (supprimer) les produits ou les machines d'une cellule. La solution obtenue est rendue réalisable par la réaffectation des produits ou des machines supprimés dans des nouvelles cellules. Ce processus est répété un certain nombre de fois et permet surtout de faire modifier la structure de la solution obtenue pour échapper aux optima locaux et garantir l'exploration de nouvelles régions de l'espace des solutions. Les étapes de la recherche locale sont présentées dans l'algorithme 4.1.

Algorithme 4.1 HClonalSA-VNC : Recherche locale

Tant que critère d'arrêt est non satisfait **Faire** :

Etape 1 : Modification des cellules de machines et de produits (Intensification)

(1a) Modifier les cellules de produits selon les emplacements des machines à l'aide de l'Equation (4.1). Si aucun changement n'est possible, passer à la stratégie de destruction et de récupération (2a). Sinon, passer à l'étape (1b) ;

(1b) Modifier les cellules de machines en se basant sur les positions des produits et en utilisant l'Equation (4.2). Si aucun déplacement n'est appliqué, passer à la stratégie de destruction et de récupération (2b). Sinon, aller à l'étape (1a) ;

Etape 2 : Stratégie de destruction et de récupération (Diversification)

(2a) Supprimer les produits d'une cellule puis réaffecter les produits supprimés dans de nouvelles cellules. Ensuite, aller à l'étape (1b) ;

(2b) Supprimer les machines d'une cellule puis réassigner les machines supprimées dans de nouvelles cellules. Aller par la suite à l'étape (1a) ;

Fin Tant que

4.3.6 Algorithme du HClonalSA-VNC

La méthode HClonalSA-VNC commence par une phase d'initialisation où le nombre de produits P et le nombre de machines M sont déterminés. Ensuite, une population initiale de solutions réalisables est créée. Chaque machine i et produit j sont attribués de manière aléatoire à une seule cellule k afin de satisfaire les contraintes du problème (voir Chapitre 2, Section 2.3.1.2). Après la phase d'initialisation, la méthode proposée s'exécute jusqu'à atteindre un nombre spécifique d'itérations.

Dans chaque itération, un pourcentage b de solutions avec de grandes affinités est choisi et proliféré en plusieurs clones en fonction de leurs affinités. Les clones de chaque solution sélectionnée subissent un opérateur de mutation et celui qui a l'efficacité de groupement la plus élevée sera utilisé par l'opérateur de croisement pour produire de nouvelles solutions. Ensuite, un processus de réparation est effectué pour réparer les solutions non réalisables (présence de cellules vides ou cellules sans produits et ou sans machines) qui peuvent survenir lors de la phase de mutation ou de croisement. Puis, un mécanisme de recherche locale est appliqué à deux solutions choisies au hasard afin d'intensifier la recherche vers des régions prometteuses. Enfin, le processus de receptor editing est effectué pour remplacer les mauvaises solutions.

La différence principale entre cette méthode et la première approche HClonalSA1, proposée pour résoudre la première variante du CFP avec cellules fixes (Chapitre 2, Section 2.3), réside dans l'intégration de l'opérateur de mutation et l'utilisation d'une méthode de recherche locale basée sur une stratégie de destruction pour mieux construire des solutions. Les étapes de HClonalSA-VNC sont résumées dans l'algorithme 4.2.

Algorithme 4.2 Algorithme de HClonalSA-VNC

Etape 1 : Génération de la population initiale ;

Etape 2 : **Tant que** critère d'arrêt est non satisfait **Faire** :

1. Calcul de l'affinité de chaque solution en utilisant l'Equation (2.1) du Chapitre 2 ;
2. Sélection d'un pourcentage b de solutions de la population ayant les meilleures affinités ;
3. Sélection clonale : le processus de génération des copies de ces meilleures solutions est effectué en fonction de la valeur d'affinité, la taille de la population (*popsize*) et la somme des affinités de la population (Chapitre 2, Equation (2.2)) ;
4. **Application de la mutation sur les copies des solutions générées ;**
5. Utilisation du croisement uniforme sur les solutions résultantes de la mutation ;
6. **Application de la recherche locale sur deux solutions choisies aléatoirement à partir des solutions obtenues par le croisement ;**
7. Remplacement d'un pourcentage w de solutions, ayant de mauvaises affinités, de la population.

Fin Tant que

4.4 Résultats expérimentaux

La méthode HClonalSA-VNC proposée est testée sur 40 instances de référence. La taille de ces instances varie de 7 produits et 5 machines à 100 produits et 40 machines afin d'évaluer l'efficacité

de la méthode. Le langage Java est utilisé pour le codage de HClonalSA-VNC. Les tests sont réalisés sur un ordinateur Intel ® Core^{TM} i7 cadencé à 2.30 GHz et 8 GB de RAM.

4.4.1 Réglage de paramètres

Trois paramètres peuvent affecter la performance de la méthode proposée : la taille de la population (*popsize*), le nombre maximal d'itérations et le pourcentage b des solutions sélectionnées pour le processus de clonage. Parmi ces paramètres, la taille de la population et le nombre maximal d'itérations dépendent de la taille de l'instance. Pour les instances de petite taille, le nombre d'itérations et la taille de la population peuvent être assez réduits sans affecter la qualité de la solution. Au contraire, les valeurs de ces deux paramètres devraient être élevées pour des instances de grande taille.

Après avoir effectué des tests, nous avons choisi de fixer le nombre maximum d'itérations à 100. De plus, une étude statistique basée sur l'analyse de la variance (ANOVA) est conçue pour déterminer les facteurs importants qui influencent l'efficacité des solutions.

Selon des tests préliminaires, trois valeurs sont possibles pour le facteur b (0.2, 0.25, 0.3) et quatre valeurs pour le facteur *popsize* (20, 30, 40, 50) ont été considérées. Un test d'importance avec un niveau de confiance égal à 95% est utilisé pour déterminer l'importance de ces facteurs. Un facteur peut affecter l'efficacité de la solution s'il obtient une valeur élevée de F (Test de Fisher) et une valeur faible de probabilité P (ou Sig).

Nous avons utilisé une instance avec 40 produits et 24 machines (P26 dans le Tableau 4.2) pour l'étude statistique. Le Tableau 4.1 présente les résultats de cette étude. La probabilité P de la taille de la population est inférieure à 0.05 ($P(popsize) = 0.018$) ; nous concluons que le facteur popsize influence l'efficacité de la solution. De plus, $P(popsize * b) < 0.05$ démontre qu'il existe une interaction entre la taille de la population et le pourcentage b. La Figure 4.3 montre cette interaction et indique que les valeurs appropriées de b et *popsize* sont 0.2 (20%) et 50, respectivement.

TABLE 4.1 – HClonalSA-VNC : ANOVA table

Source	DF	SS	MS	F	P
popsize	3	1.820	0.607	3.520	0.018
b	2	0.328	0.164	0.951	0.389
$popsize * b$	6	3.004	0.501	2.906	0.012
Error	108	18.611	0.172		
Total	120	276231.190			

DF : Degré de liberté ; SS : somme des carrés ; MS : carré moyen ; F : Test de Fisher ; P : probabilité

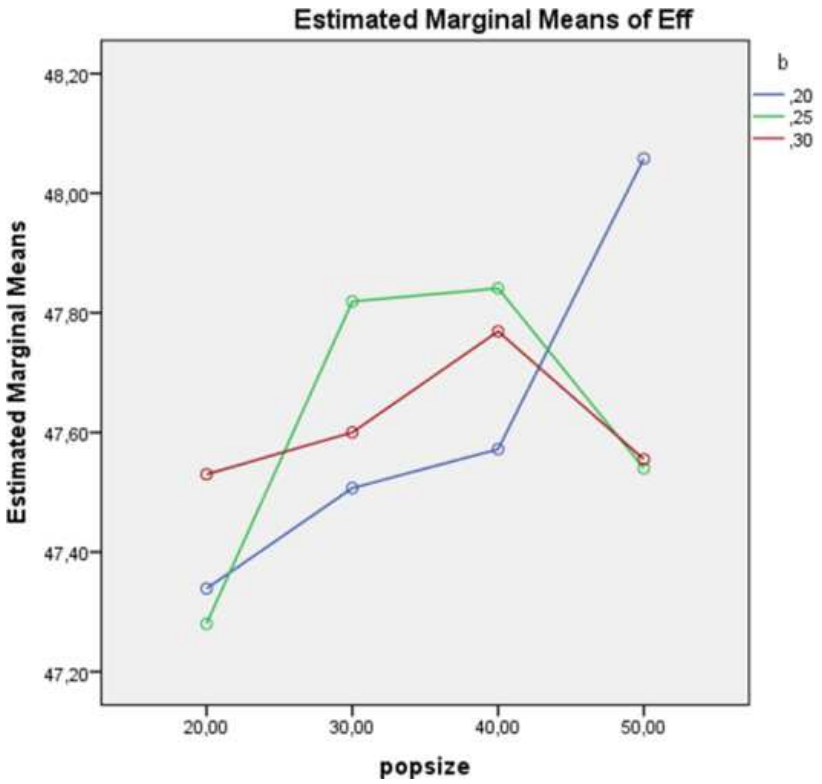


FIGURE 4.3 – HClonalSA-VNC : Etude statistique d'ANOVA

4.4.2 Résultats

La performance de notre méthode HClonalSA-VNC est évaluée sur 40 instances de référence. Dix exécutions distinctes sont effectuées pour chaque instance avec les paramètres mentionnés précédemment. Le Tableau 4.2 présente la source (la référence origine de l'instance), le nombre de machines et le nombre de produits de chaque instance. De plus, l'efficacité des meilleures solutions connues, l'efficacité des solutions minimales, moyennes et maximales atteintes par notre méthode, et le temps de calcul en secondes nécessaire sont respectivement rapportés dans ce tableau.

D'après les résultats présentés dans le Tableau 4.2, HClonalSA-VNC a pu atteindre l'efficacité des meilleures solutions connues pour 39 instances. Ces résultats sont stables pour 25 instances (1 à 13, 20, 22 à 24, 31, 33, 34, 36 et 40) au cours des 10 exécutions de l'algorithme. Concernant les instances restantes, les efficacités des solutions moyennes sont très proches de celles des meilleures solutions connues.

À propos du temps de calcul requis pour atteindre les meilleures solutions, les 11 premières instances sont résolues en moins d'une seconde et 14 instances (12 à 15, 17 à 19 et 32 à 38) sont résolues en moins de cinq secondes. En ce qui concerne le reste, le temps de calcul augmente avec la taille de l'instance. Les résultats obtenus montrent que la méthode proposée jouit d'une performance élevée pour trouver de très bonnes solutions tout en consommant un temps de calcul raisonnable.

TABLE 4.2 – HClonalSA-VNC : Résultats expérimentaux pour 40 instances de référence

No.	Source	M	P	Eff_Best	Eff_Min	Eff_Avg	Eff_Max	Time(s)
1	(King et Nakornchai, 1982)	5	7	**82.35**	82.35	82.35	**82.35**	0.02
2	(Waghodekar et Sahu, 1984)	5	7	**69.57**	69.57	69.57	**69.57**	0.02
3	(Seifoddini, 1989)	5	18	**79.59**	79.59	79.59	**79.59**	0.07
4	(Kusiak et Cho, 1992)	6	8	**76.92**	76.92	76.92	**76.92**	0.03
5	(Kusiak et Chow, 1987)	7	11	**60.87**	60.87	60.87	**60.87**	0.07
6	(Boctor, 1991)	7	11	**70.83**	70.83	70.83	**70.83**	0.09
7	(Seifoddini et Wolfe, 1986)	8	12	**69.44**	69.44	69.44	**69.44**	0.12
8	(Chandrasekharan et Rajagopalan, 1986b)	8	20	**85.25**	85.25	85.25	**85.25**	0.28
9	(Chandrasekharan et Rajagopalan, 1986a)	8	20	**58.72**	58.72	58.72	**58.72**	0.20
10	(Mosier et Taube, 1985a)	10	10	**75.00**	75.00	75.00	**75.00**	0.22
11	(Chan et Milner, 1982)	10	15	**92.00**	92.00	92.00	**92.00**	0.40
12	(Askin et Subramantan, 1987)	14	24	**72.06**	72.06	72.06	**72.06**	1.65
13	(Stanfel, 1985)	14	24	**71.83**	71.83	71.83	**71.83**	1.56
14	(McCormick *et al.*, 1972)	16	24	**53.26**	52.69	53.20	**53.26**	2.41
15	(Srinlvasan *et al.*, 1990)	16	30	**69.53**	69.4	69.50	**69.53**	2.97
16	(King, 1980)	16	43	**57.53**	57.32	57.32	**57.53**	8.12
17	(Carrie, 1973)	18	24	**57.73**	56.31	57.38	**57.73**	4.28
18	(Mosier et Taube, 1985b)	20	20	**43.45**	42.54	42.92	**43.45**	4.66
19	(Kumar *et al.*, 1986)	20	23	**50.81**	49.6	50.44	**50.81**	4.13
20	(Carrie, 1973)	20	35	**77.91**	77.91	77.91	**77.91**	11.60
21	(Boe et Cheng, 1991)	20	35	**57.98**	57.51	57.93	**57.98**	9.81
22	(Chandrasekharan et Rajagopalan, 1989)	24	40	**100.0**	100.0	100.0	**100.0**	25.47
23	(Chandrasekharan et Rajagopalan, 1989)	24	40	**85.11**	85.11	85.11	**85.11**	30.21
24	(Chandrasekharan et Rajagopalan, 1989)	24	40	**73.51**	73.51	73.51	**73.51**	31.83
25	(Chandrasekharan et Rajagopalan, 1989)	24	40	**53.29**	52.8	53.04	**53.29**	36.36
26	(Chandrasekharan et Rajagopalan, 1989)	24	40	**48.95**	47.95	48.11	**48.95**	36.27
27	(McCormick *et al.*, 1972)	27	27	**54.82**	54.45	54.77	**54.82**	16.57
28	(Kumar et Vannelli, 1986)	30	41	**63.31**	62.14	62.66	**63.31**	74.24
29	(Stanfel, 1985)	30	50	**60.12**	58.7	59.4	59.77	98.56
30	(McCormick *et al.*, 1972)	37	53	**60.64**	60.54	60.62	**60.64**	123.71
31	(Chandrasekharan et Rajagopalan, 1987)	40	100	**84.03**	84.03	84.03	**84.03**	690.81
32	(Chandrasekharan et Rajagopalan, 1989)	16	30	**52.29**	52.17	52.26	**52.29**	2.46
33	(Chandrasekharan et Rajagopalan, 1989)	16	30	**63.04**	63.04	63.04	**63.04**	2.40
34	(Chandrasekharan et Rajagopalan, 1989)	16	30	**68.38**	68.38	68.38	**68.38**	3.52
35	(Chandrasekharan et Rajagopalan, 1989)	16	30	**50.00**	49.29	49.55	**50.00**	2.92
36	(Chandrasekharan et Rajagopalan, 1989)	16	30	**72.73**	72.73	72.73	**72.73**	3.46
37	(Chandrasekharan et Rajagopalan, 1989)	16	30	**77.31**	77.31	77.31	**77.31**	2.88
38	(Chandrasekharan et Rajagopalan, 1989)	16	30	**73.24**	73.24	73.24	**73.24**	3.18
39	(Chandrasekharan et Rajagopalan, 1989)	20	35	**77.3**	77.3	77.3	**77.3**	10.95
40	(Chandrasekharan et Rajagopalan, 1989)	24	40	**72.37**	72.37	72.37	**72.37**	28.30

Pour confirmer la bonne performance de HClonalSA-VNC, les meilleures efficacités obtenues sont comparées avec celles obtenues par d'autres méthodes récemment développées pour optimiser la production cellulaire où le nombre de cellules n'est pas fixé à priori :

— L'algorithme de groupement génétique HGGA (James *et al.*, 2007) ;

— L'algorithme d'évolution différentielle HGDE (Noktehdan *et al.*, 2010) ;

— L'algorithme d'optimisation par essaims particulaires HGBPSO (Kashan *et al.*, 2014) ;

— L'algorithme du championnat de la ligue GLCA (Noktehdan *et al.*, 2016).

TABLE 4.3 – HClonalSA-VNC : Performance de la méthode HClonalSA-VNC par rapport à d'autres méthodes

No.	HGGA	HGDE	HGBPSO	GLCA	HClonalSA-VNC	Best
1	**82.35**	**82.35**	**82.35**	**82.35**	**82.35**	**82.35**
2	**69.57**	**69.57**	**69.57**	**69.57**	**69.57**	**69.57**
3	**79.59**	**79.59**	**79.59**	80.85[a]	**79.59**	**79.59**
4	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**	**76.92**
5	**60.87**	**60.87**	**60.87**	**60.87**	**60.87**	**60.87**
6	**70.83**	**70.83**	**70.83**	**70.83**	**70.83**	**70.83**
7	**69.44**	**69.44**	**69.44**	**69.44**	**69.44**	**69.44**
8	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**
9	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**
10	**75.00**	**75.00**	**75.00**	**75.00**	**75.00**	**75.00**
11	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**
12	**72.06**	**72.06**	**72.06**	73.53[a]	**72.06**	**72.06**
13	**71.83**	**71.83**	**71.83**	**71.83**	**71.83**	**71.83**
14	52.75	53.41	53.41	53.41	**53.26**	**53.26**
15	68.99	68.99	68.99	68.99	**69.53**	**69.53**
16	**57.53**	**57.53**	**57.53**	**57.53**	**57.53**	**57.53**
17	**57.73**	**57.73**	**57.73**	**57.73**	**57.73**	**57.73**
18	43.18	**43.45**	43.26	**43.45**	**43.45**	**43.45**
19	**50.81**	**50.81**	**50.81**	**50.81**	**50.81**	**50.81**
20	**77.91**	**77.91**	**77.91**	**77.91**	**77.91**	**77.91**
21	**57.98**	**57.98**	**57.98**	**57.98**	**57.98**	**57.98**
22	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
23	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**
24	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**
25	**53.29**	**53.29**	**53.29**	**53.29**	**53.29**	**53.29**
26	**48.95**	**48.95**	**48.95**	-	**48.95**	**48.95**
27	54.02	54.02	**54.82**	**54.82**	**54.82**	**54.82**
28	**63.31**	**63.31**	**63.31**	**63.31**	**63.31**	**63.31**
29	59.77	59.77	59.77	59.77	59.77	**60.12**
30	**60.64**	**60.64**	**60.64**	**60.64**	**60.64**	**60.64**
31	**84.03**	-	-	-	**84.03**	**84.03**
32	52.05	**52.29**	**52.29**	**52.29**	**52.29**	**52.29**
33	62.99	**63.04**	**63.04**	**63.04**	**63.04**	**63.04**
34	**68.38**	**68.38**	**68.38**	**68.38**	**68.38**	**68.38**
35	49.65	**50.00**	**50.00**	**50.00**	**50.00**	**50.00**
36	72.37	72.37	72.37	72.73[a]	**72.73**	**72.73**
37	**77.31**	**77.31**	**77.31**	**77.31**	**77.31**	**77.31**
38	**73.24**	**73.24**	**73.24**	**73.24**	**73.24**	**73.24**
39	**77.30**	**77.30**	**77.30**	-	**77.30**	**77.30**
40	**72.37**	**72.37**	**72.37**	-	**72.37**	**72.37**

[a] Solution avec singleton

TABLE 4.4 – HClonalSA-VNC : Résumé de la comparaison entre plusieurs méthodes

	HGGA	HGDE	HGBPSO	GLCA	HClonalSA-VNC
Nombre de meilleures solutions	31/40	35/39	34/39	30/35	39/40
Nombre de meilleures solutions en termes de pourcentage(%)	77.5	89.7	87.1	85.7	97.5

Les résultats de la comparaison sont rapportés dans le Tableau 4.3. La comparaison montre que la méthode proposée HClonalSA-VNC a surpassé HGGA, HGBPSO et HGDE dans les instances 15 et 36 en termes de qualité des solutions. Concernant la méthode GLCA, le singleton a apparu dans la solution obtenue pour l'instance 36, contrairement à notre méthode. De plus, notre méthode a pu trouver de meilleurs résultats que HGGA pour les instances 18, 32, 33 et 35, et elle a dépassé HGBPSO pour l'instance 18. HClonalSA-VNC n'a pas pu atteindre la meilleure solution trouvée dans la littérature pour une seule instance seulement (instance 29). A propos de l'instance 14, la solution optimale est déjà atteinte en utilisant une méthode exacte (Elbenani et Ferland, 2012) et elle est égale à 53.26 (la même valeur atteinte par notre méthode). En ce qui concerne les méthodes HGDE, HGBPSO et GLCA, elles n'ont pas utilisé l'instance d'origine (James *et al.*, 2007).

Le Tableau 4.4 résume les résultats de la comparaison entre les algorithmes utilisés et la méthode proposée et montre que notre méthode domine les autres méthodes en atteignant les solutions les plus connues de 97.5% des instances de tests.

Une deuxième comparaison entre HClonalSA-VNC et les autres méthodes est effectuée à base de l'écart trouvé durant les tests, tel que présenté dans la Figure 4.4. La valeur de l'écart (%gap) est calculée pour les instances résolues par les cinq méthodes.

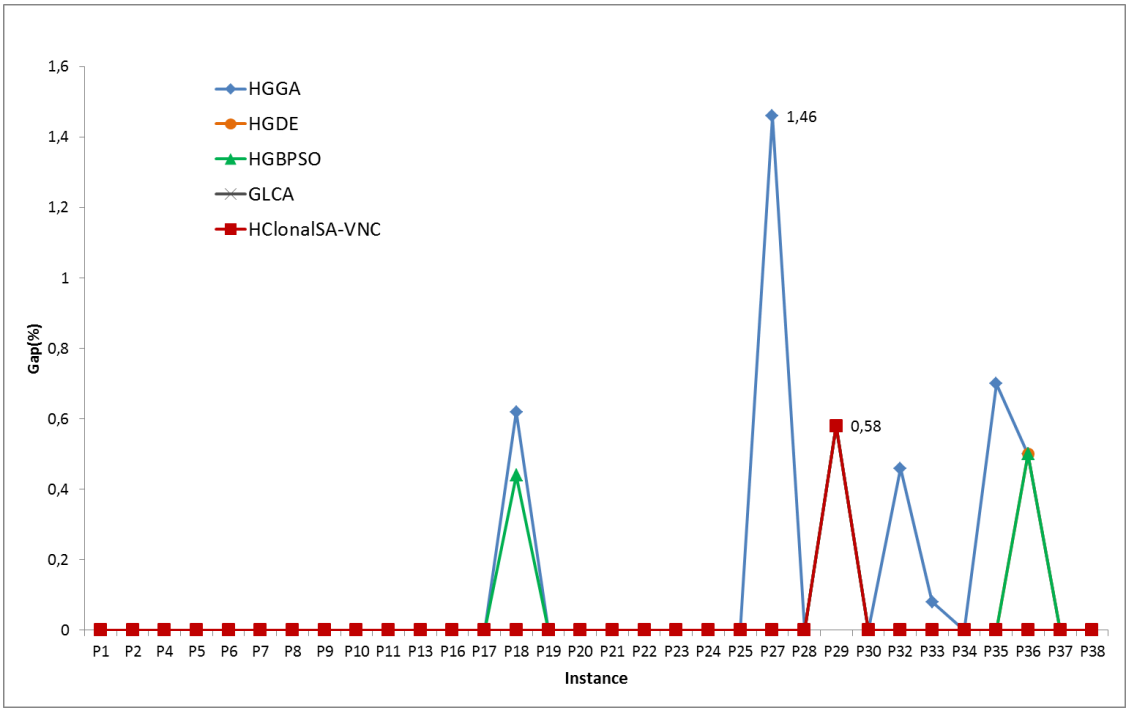


FIGURE 4.4 – HClonalSA-VNC : Comparaison de gaps entre les méthodes

D'après la Figure 4.4, les méthodes HClonalSA-VNC et GLCA ont enregistré un seul écart égal à 0.58% dans l'instance 29. La méthode HGDE a des écarts dans deux instances où le plus élevé est égal à 0.58% (instance 29). La méthode HGBPSO a présenté des écarts dans trois instances ; le plus grand écart est égal à 0.58%. Enfin, la méthode HGGA a enregistré des écarts dans sept instances ; la plus grande valeur concerne l'instance 27 et elle est égale à 1.46%.

Les tests effectués ont montré que notre méthode a pu atteindre l'efficacité des meilleures solutions connues pour 97.5% des instances testées. De plus, elle a surpassé d'autres méthodes au niveau de la qualité des solutions trouvées ce qui prouve sa bonne performance et son efficacité.

4.5 Conclusion

Durant ce chapitre, nous avons traité une autre version du problème de formation de cellules où le nombre de cellules n'est pas fixé à l'avance. Notre objectif est d'atteindre une meilleure organisation de la production avec possibilité de minimiser le nombre de cellules et réduire éventuellement le nombre des mouvements intercellulaires. Les résultats obtenus ont montré la bonne performance de notre méthode puisqu'elle a pu atteindre l'efficacité des meilleures solutions connues pour 97.5% des instances. Une étude comparative entre cette méthode et d'autres algorithmes a confirmé l'efficacité de notre méthode.

Jusqu'à présent, nous avons considéré que chaque produit a un seul scénario ou route de production, mais dans la réalité un produit peut avoir plusieurs routes de fabrication. Dans ce cas, le problème serait comment choisir la meilleure route qui permet d'éliminer/réduire les mouvements intercellulaires dans la solution finale. Ceci nous amène à une autre problématique reconnue en littérature par le problème de formation de cellules généralisé ou le problème de formation de cellules avec routes alternatives que nous allons aborder dans le chapitre suivant.

Chapitre

5 CFP généralisé : Algorithmes de sélection clonale et de recherche du coucou

Sommaire

5.1 Introduction

Comme nous l'avons déjà souligné, la formation des cellules de production se traduit par le groupement des machines en sous-groupes et les produits en familles. Le traitement de ce problème devient plus complexe avec la présence de plusieurs facteurs de production tels que les routes alternatives, le volume et le temps de production, la capacité et la fiabilité des machines, etc. Durant ce chapitre, nous allons traiter une autre version, reconnue par le problème de formation de cellules généralisé, qui prend en compte la possibilité d'avoir plusieurs routes de traitement pour chaque produit. Nous définissons quelques notions utilisées dans ce chapitre , puis nous listons des méthodes proposées pour résoudre le problème. Ensuite, nous présentons deux modèles mathématiques pour

traiter le problème généralisé. Le but du premier modèle est de minimiser le coût des mouvements intercellulaires et de maximiser la fiabilité des machines. Nous avons proposé une approche basée sur l'algorithme de sélection clonale pour le résoudre. Le deuxième modèle peut être considéré comme une extension du premier modèle prenant en compte le coût des mouvements intracellulaires et le coût du traitement des produits. Nous avons appliqué l'algorithme de recherche du coucou pour ce second modèle. Notre objectif est de rendre le problème plus réaliste en ajoutant différents facteurs de production et montrer ainsi la capacité de nos méthodes à répondre aux différentes variantes du problème de formation de cellules. Les deux méthodes proposées sont testées sur un ensemble d'instances. Les résultats obtenus sont comparés avec ceux d'une méthode Branch and Bound, en utilisant le logiciel LINGO.

5.2 Problème de formation de cellules généralisé

5.2.1 Définitions

Plusieurs chercheurs estiment que chaque produit a une seule route de production (ou de traitement) avec des opérations spécifiques, et que toutes les machines sont 100% fiables. Mais, dans l'environnement réel de production, un produit peut être fabriqué selon différentes routes (ou voies) avec un certain nombre d'opérations. Ces opérations peuvent être effectuées sur plusieurs machines alternatives, ce qui garantit une souplesse dans la conception du système de fabrication. La possibilité d'avoir des routes alternatives transforme le CFP en un problème de formation de cellules généralisé (GCFP : Generalized Cell Formation Problem). Kusiak (1987) a été l'un des premiers chercheurs à envisager ces routes alternatives. La prise en compte des routes alternatives offre plusieurs avantages, tels que la réduction du nombre des machines et la maximisation de leur utilisation, la souplesse et l'amélioration du taux de rendement du système, etc.
La considération des routes alternatives et de plusieurs facteurs de production rend le problème plus réaliste, mais aussi plus complexe. Par la suite, nous présentons les notions de quelques facteurs pris en compte dans notre travail.

Fiabilité des machines : Les machines sont des composantes essentielles dans les systèmes de production, mais elles peuvent être exposées à des pannes dues à l'utilisation. Ceci est difficile à surmonter aussi rapidement que le système nécessite même avec les routes alternatives. Ces pannes peuvent affecter gravement les mesures de performance du système et conduire à des problèmes de planification qui diminuent la productivité totale. Par conséquent, la considération de la fiabilité des machines est importante pour la conception du système de production cellulaire. Cependant, un nombre limité d'auteurs ont étudié la réduction du coût total due à la défaillance des machines, qui comprend généralement le coût de la réparation de la machine, le coût de la suspension de la production et le coût de la perte de capacité (Ameli et Arkat, 2008).

Volume de production : Les volumes de production (les demandes) peuvent influencer de manière significative la formation des cellules. En définissant ce paramètre, le temps requis pour réaliser chaque opération sur une machine peut être calculé. Le trafic et les mouvements entre les cellules sont proportionnels aux volumes de production. En outre, ce paramètre est nécessaire pour calculer le coût de production, et plus particulièrement les frais de transports.

Séquence des opérations : La séquence des opérations d'un produit représente l'ordre des opérations de fabrication nécessaires pour construire ce produit séquentiellement. La prise en compte de

cet ordre complique le problème de formation de cellules ; par conséquent de nombreux algorithmes ne traitent pas ce cas. Cependant, la séquence des opérations permet de calculer les mouvements intercellulaires et intracellulaires. Les méthodes de formation de cellules, sans considération de la séquence des opérations, peuvent déterminer les mouvements intercellulaires en fonction du nombre de cellules qu'un produit visitera durant sa fabrication. Néanmoins, le nombre de ces cellules peut être inférieur au nombre actuel des mouvements puisque le produit peut se déplacer entre les cellules plusieurs fois. De plus, la séquence des opérations peut également être utilisée comme mesure de similarité entre les produits pour identifier ceux qui peuvent être regroupés (Solimanpur *et al.*, 2004a).

5.2.2 Contraintes

Plusieurs contraintes doivent être respectées durant la formation des cellules de fabrication. Parmi ces contraintes, seules les contraintes liées aux machines et celles liées aux cellules seront étudiées dans cette section :

5.2.2.1 Contraintes sur les machines

- Capacité de la machine : l'une des exigences fondamentales dans la conception des systèmes de production cellulaire est le fait de considérer les capacités disponibles des machines. Ces capacités doivent être suffisantes pour satisfaire le volume de production requis par les produits.

- Proximité : Les contraintes de proximité des machines peuvent être divisées en deux types : contraintes de séparation et contraintes de collocation. Certaines machines doivent être séparées l'une de l'autre, tandis que d'autres machines doivent être placées ensemble pour des besoins techniques et des raisons de sécurité. Par exemple, les machines qui produisent du bruit, de la poussière, des vibrations, ou des températures élevées doivent être séparées des cellules d'assemblage électronique et des essais finaux qui nécessitent un calme et une précision. D'autre part, les machines qui partagent des ressources communes peuvent être placées dans la même cellule. Plusieurs auteurs ont considéré ces contraintes dans leurs modèles tels que (Sofianopoulou, 1999).

5.2.2.2 Contraintes sur les cellules

L'utilisation de cellules de grande taille rend la planification, l'ordonnancement et le contrôle de la production plus difficile. D'autre part, la décomposition du système de fabrication en cellules plus petites peut augmenter le trafic intercellulaire d'où la nécessité d'étudier plusieurs contraintes telles que le nombre et la taille des cellules. En pratique, le nombre de cellules serait spécifié selon des paramètres organisationnels tels que la taille des équipes de travailleurs, le type des produits, etc. La taille d'une cellule est principalement mesurée par le nombre de machines dans la cellule, mais elle peut également être exprimée selon le nombre d'opérateurs ou l'espace disponible. Au lieu de fixer la taille de cellule, elle peut être limitée par une valeur maximale ou/et une valeur minimale. Cela permettrait plus de flexibilité dans la conception du système de production.

5.2.3 Etat d'art

Par la suite, nous listons quelques méthodes proposées pour résoudre le problème de formation de cellules généralisé en considérant plusieurs facteurs de production.

5.2.3.1 Clustering

Yin et Yasuda (2002) ont développé un coefficient de similarité en considérant des routes alternatives, du volume de production, de la séquence et du temps des opérations. De plus, ils ont appliqué un algorithme heuristique à deux étapes pour résoudre le problème. La première étape consiste à regrouper les machines dans les cellules de production selon le coefficient de similarité proposé. La deuxième étape vise à améliorer les solutions et résoudre les problèmes liés à la capacité des machines par des approches prédéfinies.

Alhourani (2013) a développé un nouveau coefficient de similarité qui tient en compte l'existence des routes alternatives, la duplication et la capacité des machines, le volume de production, et la séquence des opérations. Il a également proposé un algorithme de clustering utilisant ce coefficient pour calculer la similarité entre les groupes de machines et minimiser le nombre des mouvements intercellulaires. Dans un récent travail (Alhourani, 2016), le même auteur a souligné l'importance de la fiabilité des machines dans la conception du système de production cellulaire. Il a présenté un deuxième coefficient de similarité incluant cette fiabilité et il a proposé une version modifiée de son algorithme pour minimiser le coût total du système.

5.2.3.2 Méthodes de programmation mathématique

Kusiak (1987) est l'un des premiers à développer des méthodes en programmation mathématique pour résoudre CFP avec la présence des gammes ou routes alternatives. Il a suggéré le modèle p-médian pour former simultanément les cellules de production en choisissant pour chaque produit la gamme qui donne une meilleure mise en cellules. Ce modèle est surtout testé sur des petites instances, ce qui est vu comme une limitation de son approche.

Caux *et al.* (2000) ont traité le problème de formation de cellules en y intégrant des plans de processus alternatifs et des contraintes sur la capacité des machines. Le but est de minimiser les mouvements intercellulaires en respectant ces contraintes. En effet, ils ont introduit une nouvelle approche combinant la méthode de recuit simulé pour la formation de cellules et l'approche Branch & Bound pour la sélection des routes.

Diaby et Nsakanda (2006) ont développé une formulation mathématique du problème et ils ont proposé une procédure heuristique basée sur la relaxation lagrangienne (*Lagrangean relaxation*) pour générer une solution presque optimale aux instances de grande taille. L'objectif est de minimiser le coût de manutention des matériaux et les frais d'installation, sous réserve de satisfaire toutes les demandes de produits et de ne pas dépasser les limites de capacité des machines.

Ameli et Arkat (2008) ont proposé un nouveau modèle de programmation mathématique en nombres entiers binaires pour le GCFP en tenant compte de la fiabilité des machines. Ils ont résolu le modèle par une méthode de programmation mathématique. En outre, ils ont ajouté un nouveau terme mesurant le coût de la fiabilité des machines en fonction du temps moyen entre deux pannes et du temps de production. Les mêmes auteurs ont contribué dans un autre travail (Ameli *et al.*, 2008a), dans lequel ils ont indiqué les aspects négatifs de certaines approches traditionnelles. Ils ont également présenté une nouvelle approche pour modéliser les effets de la non-fiabilité des machines et pour confirmer l'efficacité du nouveau terme proposé pour mesurer le coût.

5.2.3.3 Métaheuristiques

a) Métaheuristiques à parcours

- Recherche Tabou : Logendran et Karim (2003) ont souligné deux problèmes importants dans la conception des systèmes de fabrication cellulaire : la disponibilité d'emplacements alternatifs

pour les cellules et l'utilisation d'autres itinéraires pour déplacer les pièces avec une capacité limitée du transport de matériaux employés. Ils ont développé six versions de l'approche Tabou pour étudier l'impact de la mémoire à long terme et de la taille de la liste tabou (fixe ou variable) sur la qualité des solutions.

Wu *et al.* (2004) ont présenté une heuristique Tabou consistant d'une tenure tabou dynamique avec un mécanisme de mémoire à long terme. Le volume de production et la taille des cellules ont été pris en compte. Deux autres approches ont également été développées pour générer rapidement les solutions initiales : une approche grouper et assigner (GAA) et une approche aléatoire.

Rodrigues et Weller (2008) ont considéré la présence de routes alternatives pour les produits et la capacité de traitement des machines disponibles dans la formation des cellules. Ils ont appliqué une recherche Tabou pour assigner les machines aux cellules et une méthode Branch & Bound pour la sélection des routes.

Chung *et al.* (2011) ont introduit les problèmes liés à la fiabilité des machines, les séquences de production, le volume de production, et les routes alternatives dans la formation des cellules. Ils ont proposé une procédure composée de deux étapes : une construction initiale des solutions par des coefficients de similarité et une amélioration de ces solutions en utilisant la méthode Tabou et un opérateur de mutation.

- Recuit simulé : Sofianopoulou (1999) a présenté un modèle de programmation en nombres entiers non linéaire pour GCFP en considérant la présence de machines multiples. Il a utilisé une heuristique de recuit simulé bidimensionnel : la première dimension consiste à assigner aléatoirement chaque type de produit à un plan de traitement particulier et la deuxième dimension permet d'affecter les machines aux cellules selon les déplacements des produits et le nombre des mouvements intercellulaires. De plus, l'auteur a développé un modèle de programmation linéaire pour désigner une famille de produits.

Jayaswal et Adil (2004) ont proposé une méthode basée sur SA pour la formation des cellules, en tenant compte de la séquence des opérations, la duplication des machines, les routes alternatives. Le but est de minimiser les coûts des mouvements intercellulaires, les coûts d'investissement et de fonctionnement des machines. De plus, une heuristique de recherche locale est utilisée pour améliorer la solution produite par la méthode proposée.

Das *et al.* (2006) ont développé un modèle de programmation multi-objectif en nombres entiers mixtes pour la conception des systèmes de fabrication cellulaire avec intégration de la fiabilité des machines. Ils ont supposé que chaque produit a des routes alternatives de traitement et que chaque machine possède plusieurs copies similaires. Le but de leur modèle est de maximiser la fiabilité du système et de minimiser le coût total de la production. L'algorithme proposé combine les étapes de base du recuit simulé et les opérations de croisement et de mutation de l'algorithme génétique pour obtenir de meilleures solutions voisines.

Arkat *et al.* (2007) ont proposé une méthode qui tient compte des routes alternatives et du volume de production. Ils ont appliqué une procédure comprenant trois étapes principales : trouver le meilleur plan pour chaque produit, regrouper les machines en cellules et regrouper les produits en familles. A chaque étape, un algorithme basé sur SA a été appliqué. Les auteurs ont montré que les résultats obtenus demandent moins de temps que l'algorithme génétique.

Wu *et al.* (2007) et Wu *et al.* (2009) ont présenté des méthodes combinant un recuit simulé avec des opérateurs de l'algorithme génétique pour former les cellules de production. Dans leurs études, chaque produit a plusieurs scénarios de fabrication. Un mouvement d'insertion a été utilisé dans l'étape d'amélioration de solution afin d'accélérer la recherche et d'échapper à l'optimum local.

Jouzdani *et al.* (2014) ont présenté une version modifiée du modèle mathématique proposé par (Ameli *et al.*, 2008b) pour former les cellules en considérant de routes alternatives, de volume et de séquences de production. Le but est de maximiser la fiabilité des machines et de minimiser le coût d'installation des routes et de la manutention des matériaux. Les auteurs ont proposé une nouvelle version du recuit simulé pour résoudre leur modèle. Cette version utilise deux types de structure pour modéliser les relations cellule-machine et un type de structure pour la liaison entre les routes et les produits.

b) Métaheuristiques à population

- Algorithmes évolutionnaires : Lee *et al.* (1997) ont proposé un regroupement simultané des produits et des machines dans les cellules en considérant le volume de production, les séquences des opérations, et les routes alternatives. Ils ont développé une méthode basée sur l'algorithme génétique qui a la capacité de sélectionner une meilleure route pour chaque produit avant de regrouper les machines et les produits.

Moon et Gen (1999) ont proposé une approche basée sur GA pour la conception des cellules de fabrication en tenant compte du volume de production, la duplication et la capacité des machines, le temps de traitement, le nombre et la taille des cellules. Leur objectif est de minimiser le coût total d'investissement.

Vin *et al.* (2005) ont introduit un algorithme de groupement génétique multi-objectifs en considérant la séquence des opérations, le volume de production et les routes alternatives. L'évaluation des solutions reposait sur différents critères tels que la similarité entre les différents produits affectés à une machine, l'évaluation du coût et de la flexibilité des cellules par rapport aux contraintes liées à la capacité des machines.

Hu et Yasuda (2006) ont poursuivi une recherche basée sur des routes alternatives pour former des cellules de production et ils ont développé une méthode basée sur un algorithme de groupement génétique avec une nouvelle représentation de solutions et des techniques de croisement et de mutation spéciales qui produit des solutions efficaces.

Solimanpur et Foroughi (2011) ont proposé deux modèles mathématiques pour résoudre le problème. Le premier modèle spécifie l'itinéraire à sélectionner pour chaque produit et forme les familles de produits simultanément. La fonction objectif de ce modèle vise à minimiser la dissemblance totale entre les routes de traitement des produits situés dans la même cellule. Le deuxième modèle détermine les machines à attribuer à chaque cellule en fonction du temps de traitement, le volume de production, le budget et le coût des machines. La résolution de ces deux modèles est effectuée par un algorithme génétique. Les résultats étaient encourageants en termes de qualité de solutions et du temps de calcul par rapport à une méthode Branch & Bound.

Tavakkoli-Moghaddam *et al.* (2012) ont appliqué une méthode de recherche dispersée (Scatter search) pour former des cellules de production en y intégrant des routes alternatives, des séquences des opérations, du volume et du temps de production, ainsi que la possibilité d'avoir des machines identiques du même type. L'objectif de la méthode proposée est de minimiser le coût des mouvements intercellulaires en respectant des contraintes de la capacité des machines.

Rezazadeh et Khiali-Miab (2017) ont présenté un nouveau modèle mathématique pour la conception du système de production cellulaire en présence des routes alternatives. Ils ont appliqué un algorithme génétique à deux niveaux (two-layer GA) pour améliorer la fiabilité du système et le coût de production. Le premier niveau de cet algorithme tente à trouver une meilleure route de fabrication pour les produits, tandis que le deuxième niveau vise à renforcer la configuration des cellules en fonction des routes choisies au premier niveau.

- Optimisation par colonie de fourmis : Prabhaharan *et al.* (2005) ont proposé une approche basée sur ACO pour former des cellules afin de minimiser la variation totale de la charge cellulaire et les mouvements intercellulaires en tenant compte de la demande de production, les séquences des opérations, le temps de traitement et la capacité des machines. Les résultats obtenus ont surpassé ceux d'une méthode génétique.

Solimanpur *et al.* (2010) ont présenté un nouveau modèle pour résoudre le problème en tenant compte de la séquence des opérations et du volume de la production. Ils ont proposé ACO pour optimiser le système de production par la réduction des mouvements intercellulaires et des éléments vides. Par conséquent, les résultats obtenus étaient encourageants.

5.2.3.4 Méthodes de l'intelligence artificielle

- Réseaux de neurones : Zolfaghari et Liang (1998) ont considéré le temps de traitement, la capacité et la duplication des machines dans la formation des cellules. Une nouvelle mesure d'efficacité de groupement, qui tient compte du temps de traitement, est utilisée pour l'évaluation des solutions. Les auteurs ont adopté un algorithme de recuit simulé et une approche du réseau de neurones de Hopfield pour résoudre le problème dans un temps raisonnable.

- Logique floue : Won et Currie (2007) ont développé une approche floue de ART pour la formation des cellules, nommé Fuzzy ART/RRR-RSS (*Fuzzy ART/ReaRRangement-ReaSSignment*), où la séquence des opérations, le volume de production et la duplication des machines ont été prises en compte. L'approche proposée se compose de deux étapes : une étape de regroupement basée sur un algorithme flou de ART et une étape de réaffectation des produits et des machines exceptionnelles avec duplication des machines en cas de besoin. Les auteurs ont également introduit une nouvelle mesure de performance pour évaluer la qualité des solutions.

Papaioannou et Wilson (2009) ont proposé une formulation basée sur la programmation mathématique dans laquelle les produits et les machines sont assignés aux cellules simultanément, en prenant en compte la séquence des opérations, le coût d'installation, le taux d'utilisation et la duplication des machines. Les auteurs ont considéré le concept de flou dans certaines contraintes et dans la fonction objectif impliquée. En outre, ils ont utilisé un certain nombre d'opérateurs flous existants et des fonctions d'appartenance pour résoudre mathématiquement le modèle.

Le Tableau 5.1 résume les méthodes rapportées dans cette section et qui sont proposées pour résoudre le GCFP avec la présence d'autres facteurs de production. Le but est d'indiquer les différentes contraintes abordées par la majorité des auteurs.

D'après la littérature, un nombre limité d'auteurs ont considéré la fiabilité des machines dans leur modèle mathématique malgré le rôle important qu'elle joue dans la conception du système de production cellulaire. De plus, les chercheurs ont encouragé la duplication des machines de même type afin de minimiser les mouvements de produits entre les cellules. Cependant, la duplication de ces machines implique des investissements importants qui influencent le coût de la production.

Le but de notre prochain travail est de réduire le coût total dû à la défaillance des machines et les mouvements intercellulaires en utilisant presque les mêmes configurations proposées dans nos méthodes précédentes.

TABLE 5.1 – Liste des méthodes et des différents facteurs considérés dans la résolution de GCFP

Auteur	RA	SO	C_Inter	C_Intra	CI	CM	FM	DM	TP	VP	TC	Méthode
(Kusiak, 1987)	x											Programmation mathématique
(Liu et Wu, 1993)								x			x	Recuit simulé
(Lee *et al.*, 1997)	x	x								x		Algorithme génétique
(Zolfaghari et Liang, 1998)						x		x	x			SA + réseau de neurones Hopfield
(Sofianopoulou, 1999)	x	x						x			x	Recuit simulé
(Moon et Gen, 1999)	x					x		x		x	x	Algorithme génétique
(Caux *et al.*, 2000)	x					x						Branch & bound
(Yin et Yasuda, 2002)	x	x				x			x	x		Clustering
(Logendran et Karim, 2003)	x					x					x	Recherche Tabou
(Jayaswal et Adil, 2004)	x	x	x					x		x	x	Recuit simulé
(Wu *et al.*, 2004)										x	x	Recherche Tabou
(Prabhaharan *et al.*, 2005)		x				x			x	x		Optimisation par colonie de fourmis
(Vin *et al.*, 2005)	x	x				x				x		Algo de groupement génétique
(Diaby et Nsakanda, 2006)					x	x				x		Programmation mathématique
(Das *et al.*, 2006)	x		x				x	x	x	x	x	Recuit simulé
(Hu et Yasuda, 2006)	x		x	x						x		Algo de groupement génétique
(Won et Currie, 2007)		x						x		x		Algorithme flou de ART
(Arkat *et al.*, 2007)	x	x								x		Recuit simulé
(Ameli et Arkat, 2008)	x	x	x				x		x	x	x	Programmation mathématique
(Rodrigues et Weller, 2008)	x					x						Recherche Tabou
(Papaioannou et Wilson, 2009)		x						x			x	Logique floue
(Wu *et al.*, 2009)	x										x	Recuit simulé
(Solimanpur *et al.*, 2010)		x	x							x	x	Optimisation par colonie de fourmis
(Solimanpur et Foroughi, 2011)	x	x				x			x	x		Algorithme génétique
(Chung *et al.*, 2011)	x	x	x				x		x	x	x	Recherche Tabou
(Tavakkoli-Moghaddam *et al.*, 2012)	x	x	x			x		x	x	x	x	Recherche dispersée
(Alhourani, 2013)	x	x				x		x		x		Clustering
(Jouzdani *et al.*, 2014)	x	x	x	x			x		x	x	x	Recuit simulé
(Alhourani, 2016)	x	x				x	x	x	x	x		Clustering
(Rezazadeh et Khiali-Miab, 2017)	x	x	x				x		x	x	x	Algorithme génétique

RA : Routes alternatives ; SO : Séquence des opérations ; C_Inter : Coût des mouvements intercellulaires ; C_Intra : Coût des mouvements intracellulaires ; CI : Coût d'installation ; CM : Capacité des machines ; FM : Fiabilité des machines ; DM : Duplication des machines ; TP : Temps de production ; VP : Volume de production ; TC : Taille des cellules.

5.3 Algorithme de sélection clonale pour GCFP

Notre première méthode proposée pour résoudre le GCFP est basée sur l'algorithme de sélection clonale et elle est notée Hybrid Clonal Selection Algorithm-Generalized Cell Formation Problem (HClonalSA-GCFP) (Karoum et Elbenani, 2017b). L'objectif de cette méthode est de minimiser le coût des mouvements intercellulaires des produits et le coût des pannes de machines en tenant compte du volume de production et du temps de traitement de chaque produit. La méthode HClonalSA-GCFP est testée sur un ensemble d'instances. Les solutions obtenues sont comparées avec celles fournies par le logiciel LINGO. Par la suite, nous allons présenter le modèle mathématique utilisé ; introduire les modifications appliquées à l'algorithme de sélection clonale pour résoudre le GCFP, et résumer les résultats obtenus.

5.3.1 Description du problème

Pour résoudre le problème de formation de cellules généralisé, nous avons utilisé le modèle mathématique proposé par (Ameli et Arkat, 2008). Pour décrire ce modèle, nous allons tout d'abord présenter le problème en considérant la possibilité d'avoir des routes alternatives de traitement seulement, puis nous intégrons la fiabilité des machines et nous réécrivons la version finale du modèle avec ces contraintes.

5.3.1.1 CFP avec des routes alternatives de traitement

Le problème est formulé comme un modèle de programmation linéaire, noté $MG1(Y, Z)$, sous les hypothèses suivantes :

— Le nombre total de cellules est connu ;

— La limite inférieure et la limite supérieure pour le nombre de machines dans chaque cellule sont connues ;

— Chaque type de produit possède une ou plusieurs routes de traitement, mais une seule route peut être choisie ;

— Les opérations de chaque route de traitement sont effectuées sur différentes machines selon un ordre donné. De plus, la séquence d'opérations est importante dans le calcul du coût de manutention intercellulaire des matériaux, car elle peut influencer le nombre de fois le produit doit se déplacer entre les machines de différentes cellules ou entre les machines de la même cellule ;

— Le temps de traitement de chaque opération pour chaque type de produit sur chaque machine est déterminé ;

— Une seule machine de chaque type est utilisée. La duplication des machines de même type n'est pas prise en compte dans notre méthode ;

— Le volume de production pour chaque type de produit est connu ;

Les notations suivantes sont utilisées :

M	: nombre de machines.
P	: nombre de produits.
C	: nombre de cellules.
R	: nombre total de routes.
V_j	: volume de production du produit j.
R_j	: nombre de routes alternatives du produit j.
L_k	: limite inférieure (lower limit) du nombre de machines dans la cellule k.
U_k	: limite supérieure (upper limit) du nombre de machines dans la cellule k.
NM_{jr}	: nombre de machines dans la route r du produit j.
$\{u_{jr}^{(1)}, u_{jr}^{(2)}, ..., u_{jr}^{(NM_{jr})}\}$	: indice de la machine dans la route r du produit j.
A_{jr}	: coût du mouvement intercellulaire du produit j dans la route r.

L'objectif du modèle $MG1(Y,Z)$ est de minimiser le coût des mouvements intercellulaires ($IntercellularCost$). Une expression mathématique de ce modèle est présentée comme suit :

$$MG1(Y,Z): \quad \min \ IntercellularCost = \sum_{j=1}^{P}\sum_{r=1}^{R_j}\sum_{i=1}^{NM_{jr}-1}\sum_{k=1}^{C} A_{jr}V_jZ_{jr}Y_{(u_{jr}^{(i)})k}\left(1-Y_{\left(u_{jr}^{(i+1)}\right)k}\right) \quad (5.1)$$

Sous contraintes :

$$\sum_{r=1}^{R_j} Z_{jr} = 1, \quad j = 1,2,...,P \quad (5.2)$$

$$\sum_{i=1}^{M} Y_{ik} \leq U_k, \quad k = 1,2,...,C \quad (5.3)$$

$$\sum_{i=1}^{M} Y_{ik} \geq L_k, \quad k = 1,2,...,C \quad (5.4)$$

$$\sum_{k=1}^{C} Y_{ik} = 1, \quad i = 1,2,...,M \quad (5.5)$$

Variables de décision :

$$Y_{ik} = \begin{cases} 1, & \text{si la machine } i \text{ est dans la cellule } k \\ 0, & \text{sinon.} \end{cases}$$

$$Z_{jr} = \begin{cases} 1, & \text{si la route } r \text{ du produit } j \text{ est sélectionnée} \\ 0, & \text{sinon.} \end{cases}$$

La fonction objectif du modèle (Equation (5.1)) permet de minimiser le coût de déplacement des produits entre les différentes cellules en tenant compte des volumes de production, des séquences d'opérations et des routes alternatives. La contrainte (5.2) impose qu'une seule route de traitement doit être choisie pour chaque produit. Les inégalités (5.3) et (5.4) imposent respectivement les limites supérieure et inférieure pour le nombre de machines dans chaque cellule. La contrainte (5.5) indique que chaque machine ne sera affectée qu'à une seule cellule.

Les variables (Y et Z) sont des variables binaires. Étant donné que la fonction objectif est sous une forme non linéaire, Ameli et Arkat (2008) ont proposé une autre variable binaire (X) et un ensemble de contraintes pour linéariser cette fonction objectif. L'équation de transformation est la suivante :

$$X_{jrikek} = Z_{jr} \times Y_{ik} \times \left(1 - Y_{ek}\right) \tag{5.6}$$

Où :

$$X_{jrikek} = \begin{cases} 1, & \text{Si la route } r \text{ du produit } j \text{ est choisie, la machine } i \text{ est dans} \\ & \quad \text{la cellule } k \text{ et la machine } e \text{ n'existe pas dans la cellule } k. \\ 0, & \text{Sinon.} \end{cases}$$

Contraintes de linéarisation :

$$X_{jrikek} \leq Z_{jr} \tag{5.7}$$

$$X_{jrikek} \leq Y_{ik} \tag{5.8}$$

$$X_{jrikek} \leq (1 - Y_{ek}) \tag{5.9}$$

$$Z_{jr} + Y_{ik} + (1 - Y_{ek}) - X_{jrikek} \leq 2 \tag{5.10}$$

$$Z_{jr}, Y_{ik}, X_{jrikek} \in \{0, 1\} \tag{5.11}$$

Les contraintes (5.7) à (5.9) garantissent que si l'une des variables binaires primaires (Z_{jr}, Y_{ik} ou $(1-Y_{ek})$) est égale à 0, la nouvelle variable correspondante est forcée de prendre 0. La contrainte (5.10) assure que si toutes ces mêmes variables primaires prennent une valeur égale à 1, leur nouvelle variable correspondante prend la valeur 1. Enfin, la contrainte (5.11) indique que les variables sont binaires. La fonction objectif peut être réécrite comme suit :

$$MG2(X,Y,Z): \quad \min \ IntercellularCost = \sum_{j=1}^{P} \sum_{r=1}^{R_j} \sum_{i=1}^{NM_{jr}-1} \sum_{k=1}^{C} A_{jr} V_j X_{jr(u_{jr}^{(i)})k(u_{jr}^{(i+1)})k} \tag{5.12}$$

Sous contraintes : Equations (5.2-5.5) et Equations (5.7-5.11)

5.3.1.2 Fiabilité des machines

Pour intégrer la fiabilité des machines dans le modèle précédent, les hypothèses supplémentaires suivantes sont prises en compte :

— Le temps de panne pour chaque machine suit une distribution exponentielle avec un taux de défaillance connu ;

— Le coût de panne pour chaque machine est connu.

Le coût de panne d'une machine comprend généralement le coût de réparation de cette machine, le coût de suspension de la production, etc.

Le traitement de la fiabilité (reliability) d'une machine dans la phase de conception d'un système de fabrication consiste à évaluer le temps moyen entre deux pannes (MTBF) de cette machine. La division du temps de fonctionnement d'une machine donnée sur son MTBF permet d'obtenir le nombre de ses pannes. En multipliant cette quantité par le coût de panne, nous obtiendrons le coût

de la non-fiabilité de la machine.

Pour résumer, le modèle mathématique qui minimise le coût total des mouvements intercellulaires ($IntercellularCost$) et des pannes de machines ($BreakdownCost$) est présenté, par (Ameli et Arkat, 2008), comme suit :

$$\begin{aligned} MG2(X,Y,Z): \quad \min\ TotalCost &= IntercellularCost + BreakdownCost \\ &= \sum_{j=1}^{P}\sum_{r=1}^{R_j}\sum_{i=1}^{NM_{jr}-1}\sum_{k=1}^{C} A_{jr}V_j X_{jr(u_{jr}^{(i)})k(u_{jr}^{(i+1)})k} \\ &+ \sum_{j=1}^{P}\sum_{r=1}^{R_j}\sum_{i=1}^{NM_{jr}} Z_{jr}\frac{V_j T_{j(u_{jr}^{(i)})} B_{(u_{jr}^{(i)})}}{MTBF_{(u_{jr}^{(i)})}} \end{aligned} \tag{5.13}$$

Sous contraintes :

$$\sum_{r=1}^{R_j} Z_{jr} = 1, \quad j = 1,2,...,P \tag{5.14}$$

$$\sum_{i=1}^{M} Y_{ik} \leq U_k, \quad k = 1,2,...,C \tag{5.15}$$

$$\sum_{i=1}^{M} Y_{ik} \geq L_k, \quad k = 1,2,...,C \tag{5.16}$$

$$\sum_{k=1}^{C} Y_{ik} = 1, \quad i = 1,2,...,M \tag{5.17}$$

$$X_{jrikek} \leq Z_{jr} \tag{5.18}$$

$$X_{jrikek} \leq Y_{ik} \tag{5.19}$$

$$X_{jrikek} \leq (1 - Y_{ek}) \tag{5.20}$$

$$Z_{jr} + Y_{ik} + (1 - Y_{ek}) - X_{jrikek} \leq 2 \tag{5.21}$$

$$Z_{jr}, Y_{ik}, X_{jrikek} \in \{0,1\} \tag{5.22}$$

Où :

M	: nombre de machines.
P	: nombre de produits.
C	: nombre de cellules.
R	: nombre total de routes.
V_j	: volume de production du produit j.
R_j	: nombre de routes alternatives du produit j.
L_k	: limite inférieure (lower limit) du nombre de machines dans la cellule k.
U_k	: limite supérieure (upper limit) du nombre de machines dans la cellule k.
NM_{jr}	: nombre de machines dans la route r du produit j.
$\{u_{jr}^{(1)}, u_{jr}^{(2)}, ..., u_{jr}^{(NM_{jr})}\}$	: indice de la machine dans la route r du produit j.
A_{jr}	: coût du mouvement intercellulaire du produit j dans la route r.
T_{ji}	: temps de traitement du produit j sur la machine i.
B_i	: coût de panne de la machine i.
$MTBF_i$	: temps moyen entre les pannes de la machine i.

5.3.1.3 Exemple numérique

Pour clarifier le problème introduit, nous présentons un exemple proposé par (Bhide *et al.*, 2005) et légèrement modifié par (Jouzdani *et al.*, 2014) et (Alhourani, 2016). Cet exemple est obtenu à partir d'une usine d'engrenages. Il se compose de 8 produits ($P = 8$), 9 machines ($M = 9$), 2 cellules ($C = 2$) et 20 routes de traitement ($R = 20$: P_1 a 3 routes ; P_2 a 3 routes ; les produits P_3, P_4, P_5, P_6 et P_8 ont 2 routes ; et P_7 a 4 routes ; au total il y'a 20 routes de traitement). Le nombre minimal de machines nécessaires dans chaque cellule est égal à deux, et le nombre maximal autorisé est égal à six.

Le Tableau 5.2 présente le temps moyen entre les pannes MTBF et le coût de panne de chaque machine.

TABLE 5.2 – Exemple de GCFP : Fiabilté des machines

Machine	MTBF(h)	coût de la panne
1	90	900
2	51	2000
3	73	2000
4	60	1600
5	76	1500
6	62	1800
7	71	1400
8	58	1700
9	65	1500

Le Tableau 5.3 présente la matrice d'incidence machine-produit liée à l'exemple. La première ligne et la deuxième ligne de ce tableau représentent les demandes (le volume de production) et les routes alternatives de chaque produit. La dernière ligne affiche le coût des mouvements intercellulaires (Inter_C) de chaque produit dans chaque route. Les nombres dans cette matrice sont sous la forme "la séquence d'opérations (le temps de traitement)" ; c'est-à-dire, ils montrent la séquence d'opérations nécessaires pour la fabrication des produits le long des routes et le temps de traitement de chaque opération.

TABLE 5.3 – Exemple de GCFP : Matrice d'incidence

Séquence de (temps traitement (min))	Produits (volume)																					
Machines	P1(75)			P2(130)			P3(110)		P4(145)		P5(110)		P6(105)		P7(140)				P8(115)			
	R1	R2	R3	R1	R2	R3	R1	R2	R1	R2	R1	R2	R1	R2	R1	R2	R3	R4	R1	R2		
M1	1(5)			1(4)			1(4)	1(4)	1(5)	1(5)		1(4)		1(4)	1(5)			1(5)	1(4)			
M2		1(5)	1(5)		1(5)	1(5)					1(3)		1(5)			1(3)	1(3)	2(3)		1(4)		
M3							2(3)	2(3)	2(3)	2(3)												
M4	2(3)			2(4)				3(3)	3(5)	3(5)		2(4)			2(4)		2(4)					
M5	3(4)		2(4)	3(3)	2(3)	2(3)	3(4)			4(4)					3(3)		3(3)					
M6		2(5)														2(4)		3(4)	2(3)	2(4)		
M7											2(5)	3(5)	2(5)	2(5)								
M8		3(4)		4(4)		3(4)	4(3)	4(3)	5(4)				3(5)	3(5)			4(5)					
M9	4(5)		3(5)		3(3)						3(5)	4(5)			4(5)	3(5)		4(5)				
Inter_C	375	375	0	1300	0	650	1100	0	1450		1100	550	525	0	700	0	700	700	575	0		

Le produit P_1, par exemple, a 3 routes alternatives ($R1, R2, R3$ donc $R_1 = 3$) ; sa troisième route comprend trois opérations dans trois machines différentes (M_2, M_5 et M_9). La première opération dans la machine M_2 nécessite 5 minutes de traitement ($T_{1(u_{13}^{(1)})} = 5$). Le coût de mouvements intercellulaires de ce produit P_1 dans la route 3 est égal à $A_{13} = 0$ et le volume de production est $V_1 = 75$.

L'exemple proposé est résolu à l'aide du logiciel LINGO en utilisant la méthode Branch and Bound. Le Tableau 5.4 présente la solution obtenue avec un coût de pannes égal à 284530.7 et un coût de mouvements intercellulaires nul.
- La première cellule de production contient les machines {M_2, M_7, M_9} et les produits {P_1, P_5, P_7}.
- La deuxième cellule comprend les machines {M_1, M_3, M_4, M_5, M_6, M_8} et les produits {P_2, P_3, P_4, P_6, P_8}.
Une seule route est désignée pour chaque produit. La troisième route est choisie pour fabriquer le produit P_1, la première route est sélectionnée pour le produit P_5 et ainsi de suite.
Le calcul du coût des mouvements intercellulaires du produit P_1, par exemple, est effectué en utilisant l'Equation (5.12) :
$IntercellularCost_1 = \sum_{i=1}^{NM_{13}-1=2} \sum_{k=1}^{C} A_{13} V_1 X_{13(u_{13}^{(i)})k(u_{13}^{(i+1)})k} = \{A_{13}V_1(X_{132151}+X_{132252}+X_{135191}+X_{135292})\}$
Où :
- $A_{13} = 0$
- $V_1 = 75$
- $X_{132151} = 1$ car la route 3 du produit P_1 est choisie, la machine M_2 est dans la cellule 1 et la machine M_5 n'est pas dans la même cellule. Par conséquent, un déplacement intercellulaire est réalisé.
- $X_{132252} = 0$ car la machine M_2 n'est pas dans la cellule 2 et la machine M_5 est dans la cellule 2.
- $X_{135191} = 0$ puisque les machines M_5 et M_9 ne sont pas affectées à la cellule 1.
- $X_{135292} = 0$ car la machine M_5 est dans la cellule 2 et la machine M_9 est également dans la même cellule 2.
Donc : $IntercellularCost_1 = 0 \times 75 \times (1 + 0 + 0 + 0) = 0$

TABLE 5.4 – Exemple de GCFP : Résultats obtenus par LINGO

		Produit (Route)							
		P1(R3)	P5(R1)	P7(R2)	P2(R1)	P3(R1)	P4(R1)	P6(R2)	P8(R1)
Machines	M2	1	1	1					
	M7		2					2	
	M9	3	3	3					
	M1				1	1	1	1	1
	M3					2	2		
	M4				2		3		
	M5	2			3	3			
	M6			2					2
	M8				4	4	4	3	

5.3.2 Adaptation de HClonalSA-GCFP au GCFP

Pour adapter HClonalSA-GCFP à notre problème, un nombre d'éléments doivent être spécifiés : la représentation et la génération d'une solution, les opérateurs de mutation et de croisement appliqués et la méthode de recherche locale utilisée.

5.3.2.1 Représentation d'une solution

Chaque solution comporte deux sections ; la première section indique la route de production choisie pour chaque produit et la deuxième section spécifie la cellule de chaque machine : $s = (r_1, \ldots, r_P \mid c_1, \ldots, c_M)$. La solution est codée sous forme d'un vecteur de longueur $P + M$ où P est le nombre de produits et M est le nombre de machines.

Puisque les positions des produits n'affectent pas la fonction objectif, elles ne sont pas considérées dans la représentation de la solution. Après l'obtention de la solution finale, chaque produit sera affecté à la cellule contenant le plus grand nombre de machines traitant ce produit selon la route choisie (Ameli *et al.*, 2008b).

Par exemple, la solution obtenue précédemment par LINGO pour l'exemple numérique (Tableau 5.4) sera présentée comme suit : $s = (3, 1, 1, 1, 1, 2, 2, 1 \mid 2, 1, 2, 2, 2, 2, 1, 2, 1)$.
Cette représentation montre que le produit P_1 est affecté à la route 3 ($r_1 = 3$) ; les produits P_2, P_3, P_4, P_5 et P_8 sont traités dans leurs premières routes ($r_2 = 1, r_3 = 1, r_4 = 1, r_5 = 1, r_8 = 1$) ; et chacun des produits P_6 et P_7 est affecté à sa deuxième route de traitement ($r_6 = 2, r_7 = 2$). Concernant les emplacements des machines, les machines M_1, M_3, M_4, M_5, M_6 et M_8 sont assignées à la cellule 2 ($c_1 = c_3 = c_4 = c_5 = c_6 = c_8 = 2$) ; les machines M_2, M_7 et M_9 sont déplacées dans la cellule 1 ($c_2 = c_7 = c_9 = 1$).

5.3.2.2 Génération d'une solution

Les solutions de la population initiale sont générées aléatoirement. Chaque produit j est affecté à l'une de ses routes de traitement R_j de manière aléatoire et chaque machine i est assignée à une cellule k. Les étapes nécessaires pour générer une solution sont présentées dans l'algorithme 5.1.

Algorithme 5.1 HClonalSA-GCFP : Génération d'une solution

Etape 1 : Affectation des produits aux routes
  Pour chaque produit j ($j = 1, \ldots, P$) **Faire**
    Générer aléatoirement un nombre r entre 1 et le nombre de routes de ce produit R_j
    Affecter le produit j à la route r
  Fin Pour

Etape 2 : Affectation des machines aux cellules
  Pour chaque machine i ($i = 1, \ldots, M$) **Faire**
    Générer aléatoirement un nombre k entre 1 et le nombre de cellules C
    Assigner la machine i à la cellule k
  Fin Pour

La génération aléatoire des solutions initiales peut produire des solutions non réalisables qui violent les limites inférieure et supérieure des cellules. Par conséquent, un processus de réparation est appliqué pour corriger ces solutions. Ce processus implique le déplacement des machines en

tenant compte de la limite supérieure U_k et la limite inférieure L_k de chaque cellule k et en suivant les étapes de l'algorithme 5.2.

Algorithme 5.2 HClonalSA-GCFP : Processus de réparation des solutions non réalisables

Pour chaque cellule k ($k = 1, \ldots, C$) de la solution s à réparer **Faire**
  Si le nombre de machines de la cellule k (noté n_k) est inférieure à L_k ($n_k < L_k$) **Alors**
    Déplacer $(L_k - n_k)$ machines à partir des cellules ayant un surplus en tenant compte de l'efficacité de la solution s.
  Fin Si
  Si le nombre de machines est supérieure à U_k ($n_k > U_k$) **Alors**
    Déplacer $(n_k - U_k)$ machines à partir de la cellule k vers d'autres cellules en tenant compte de la limite supérieure et de l'efficacité de la solution s.
  Fin Si
Fin Pour

5.3.2.3 Mutation et croisement

L'opérateur de mutation est appliqué aux solutions clonées en modifiant la route d'un produit ou la cellule d'une machine choisie au hasard comme illustré dans l'exemple suivant :
$s = (3, \mathbf{1}, 1, 1, 1, 2, 2, 1 \mid 2, 1, 2, 2, 2, 2, 1, 2, 1)$
s' est obtenue en changeant la route de production du produit P_2 de 1 à 3, comme montré en gras.
$s' = (3, \mathbf{3}, 1, 1, 1, 2, 2, 1 \mid 2, 1, 2, 2, 2, 2, 1, 2, 1)$
Ensuite, les solutions résultantes de la mutation vont subir un croisement uniforme. Le fonctionnement de cet opérateur est déjà expliqué dans Chapitre 2, Section 2.3.1.5.
Les modifications apportées par ces deux opérateurs peuvent produire des solutions non réalisables. Par conséquent, le processus de réparation, présenté par l'algorithme 5.2, est appliquée pour corriger ces solutions.

5.3.2.4 Recherche locale

Pour intensifier la recherche, nous avons utilisé la méthode de recherche locale introduite par (Gonçalves et Resende, 2004) (Chapitre 2, Section 2.4.1.2). Sur la base des routes choisies pour le traitement des produits, nous modifions les cellules de machines. Chaque machine sera affectée à la cellule qui améliore le plus la fonction objectif du modèle $MG2(X, Y, Z)$ (Equation (5.13)). Si la solution ainsi modifiée est meilleure que la solution courante, cette dernière va être supprimée et remplacée par la solution modifiée. Ensuite, nous modifions la route du traitement de chaque produit en fonction de la position des machines. Cette opération se répète par la modification des routes, puis la réaffectation des machines jusqu'à ce que la qualité de la nouvelle solution ne dépasse pas la qualité de la dernière solution trouvée. Il est à noter que l'ordre du choix de l'élément (machine ou route) à étudier est arbitraire.

5.3.2.5 Algorithme du HClonalSA-GCFP

Dans cette méthode HClonalSA-GCFP, nous avons suivi les mêmes étapes utilisées dans la méthode HClonalSA-VNC (voir Chapitre 4, Section 4.3.6). Le but est de montrer la capacité de notre proposition à résoudre plusieurs variantes du problème de formation de cellules. Un rappel de

la méthode est présenté dans l'algorithme 5.3

Algorithme 5.3 Algorithme de HClonalSA-GCFP

Etape 1 : Génération de la population initiale ;

Etape 2 : **Tant que** critère d'arrêt est non satisfait **Faire** :

1. Calcul de l'affinité de chaque solution en utilisant la fonction objectif du modèle $MG2(X, Y, Z)$ dans l'Equation (2.1) du Chapitre 2 ;
2. Sélection d'un pourcentage b de solutions de la population ayant les meilleures affinités ;
3. Sélection clonale : le processus de génération des copies de ces meilleures solutions est effectué en fonction de la valeur d'affinité, la taille de la population (*popsize*) et la somme des affinités de la population (Chapitre 2, Equation (2.2)) ;
4. Application de la mutation sur les copies générées ;
5. Utilisation du croisement uniforme sur les solutions résultantes de la mutation ;
6. Application de la recherche locale sur deux solutions choisies aléatoirement à partir des solutions obtenues par le croisement ;
7. Remplacement d'un pourcentage w des solutions, ayant des mauvaises affinités, de la population.

Fin Tant que

5.3.3 Résultats expérimentaux

Pour évaluer l'efficacité de la méthode proposée, nous avons préparé un ensemble d'instances de référence. Un nombre limité des chercheurs ont considéré le coût de panne des machines et le coût des mouvements intercellulaires ; par conséquent, la majorité des données sont générées aléatoirement. 16 instances sont utilisées y compris deux (5, 7) collectées de la littérature comme indiqué dans le Tableau 5.5. Les autres instances sont préparées en ajoutant les données manquantes telles que le coût de panne des machines (CP), le coût des mouvements intercellulaires des produits dans chaque route (C_inter), le volume de production (VP) et le temps de production (TP).

Le langage Java est utilisé pour le codage de HClonalSA-GCFP. Les tests sont effectués sur un ordinateur Intel ® Core^{TM} i7 clock at 2.50 GHz et 8 GB de RAM.

5.3.3.1 Réglage de paramètres

Trois paramètres peuvent affecter la performance de la méthode proposée : la taille de la population (*popsize*), le nombre maximal d'itérations et le pourcentage b des solutions sélectionnées pour le processus de clonage. Nous avons défini le nombre maximum d'itérations à 500 itérations. Une étude statistique basée sur l'analyse de la variance (ANOVA) est conçue, en utilisant le logiciel SPSS, pour déterminer les facteurs importants qui influencent l'efficacité des solutions. Puisque les instances de grande taille sont difficiles à résoudre, nous avons choisi une instance avec 30 machines, 70 produits et 149 routes alternatives (instance 16 dans Tableau 5.5) pour l'étude statistique. Le Tableau 5.6 présente les résultats de cette étude. La probabilité P du pourcentage b est inférieur à

TABLE 5.5 – HClonalSA-GCFP : Instances de référence

No.	Source	Taille (M x P x R)	Données générées aléatoirement
1	(Kusiak, 1987)	4 x 5 x 11	CP, VP , C_inter et TP
2	(Moon et Chi, 1992)	6 x 6 x 13	CP, VP , C_inter et TP
3	(Sankaran et Kasilingam, 1990)	6 x 10 x 20	CP et TP
4	(Lee *et al.*, 1997)	8 x 13 x 26	CP, C_inter et TP
5	(Alhourani, 2016)	9 x 8 x 20	—
6	(Ouk Kim *et al.*, 2004)	10 x 10 x 25	CP et C_inter
7	(Alhourani, 2016)	10 x 16 x 32	—
8	(Alhourani, 2013)	12 x 12 x 20	CP, VP et C_inter
9	(Sofianopoulou, 1999)	12 x 20 x 26	CP, VP , C_inter et TP
10	(Sofianopoulou, 1999)	14 x 20 x 26	CP, VP , C_inter et TP
11	(Ameli et Arkat, 2008)	17 x 30 x 63	C_inter
12	(Sofianopoulou, 1999)	18 x 30 x 59	CP, VP , C_inter et TP
13	(Nagi *et al.*, 1990)	20 x 20 x 51	CP et C_inter
14	(Won et Kim, 1997)	26 x 28 x 71	CP, VP , C_inter et TP
15	(Lee *et al.*, 1997)	30 x 40 x 89	CP, C_inter et TP
16	(Hu et Yasuda, 2006)	30 x 70 x 149	CP, VP , C_inter et TP

0.05 ($P(b) = 0.004$) ; nous concluons que le facteur b influence l'efficacité de la solution. De plus, $P(popsize * b) < 0.05$ démontre qu'il existe une interaction entre la taille de la population et le pourcentage b. Pour déterminer les valeurs appropriées des facteurs, nous devrions commencer par celui ayant la valeur F la plus élevée. Selon la Figure 5.1, les valeurs appropriées de b et *popsize* sont 0.25(25%) et 30, respectivement.

TABLE 5.6 – HClonalSA-GCFP : Table d'ANOVA

Source	DF	SS	MS	F	P
popsize	3	4.90 E+8	1.63 E+8	1.75	0.161
b	2	1.08 E+9	5.42 E+8	5.82	0.004
$popsize * b$	6	1.40 E+9	2.34 E+8	2.51	0.026
Erreur	108	1.00 E+10	9.32 E+7		
Total	120	3.05 E+15			

DF : DF : Degré de liberté ; SS : somme des carrés ; MS : carré moyen ; F : Test de Fisher ; P : probabilité

5.3.3.2 Résultats

Nous avons effectué dix exécutions distinctes de l'algorithme pour chaque instance avec les paramètres mentionnés précédemment. Les meilleurs résultats enregistrés par HClonalSA-GCFP sont comparés aux solutions optimales obtenues par la méthode Branch & Bound à l'aide du logiciel LINGO 16. Le Tableau 5.7, présente les résultats de la comparaison. Pour chaque instance de référence, le coût total des mouvements intercellulaires (TC_inter), le coût total de pannes des machines

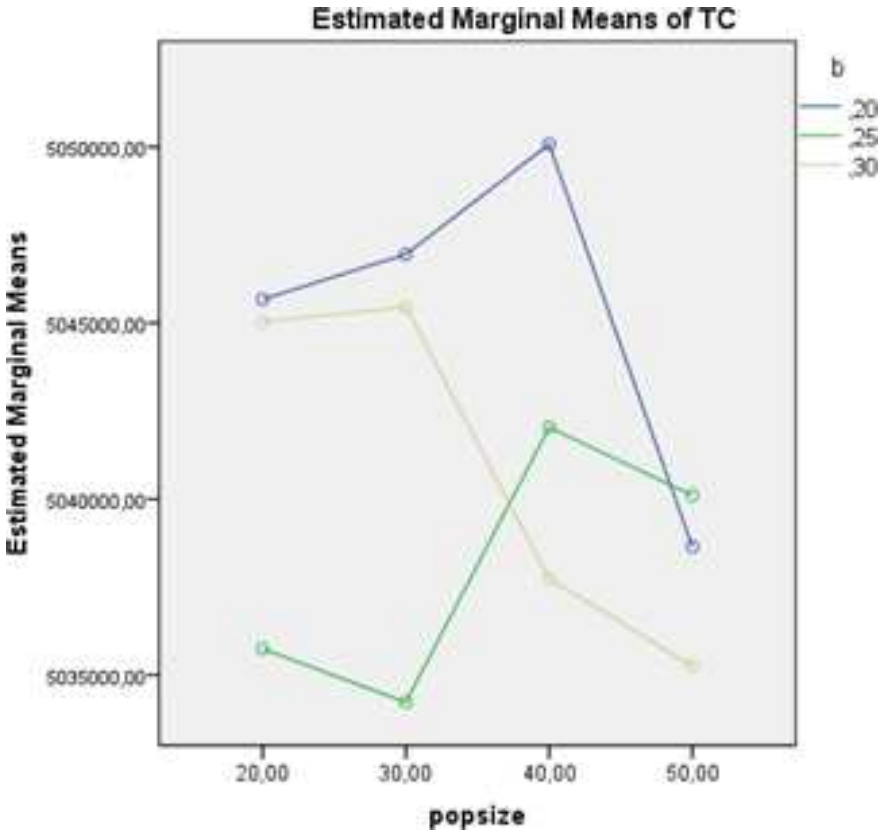


FIGURE 5.1 – HClonalSA-GCFP : Etude statistique d'ANOVA

(TCP), la somme des coûts (TC = TC_inter+ TCP) et le temps d'exécution (CPU(s)) sont présentés dans ce tableau, respectivement. Le temps d'exécution maximum de LINGO est fixé à 72 heures.

D'après le Tableau 5.7, la méthode proposée a pu atteindre l'optimum global pour 14 instances (1-13, 15) en moins de trois secondes ; tandis que LINGO a nécessité un temps de calcul important pour résoudre la majorité de ces instances. Les résultats obtenus confirment la capacité de HClonalSA-GCFP à résoudre le problème en consommant un temps de calcul raisonnable.

Concernant l'instance 14, LINGO n'a pas pu trouver la solution optimale après plus de 72 heures d'exécution. Cependant, notre méthode a atteint la même valeur obtenue par LINGO en moins de 2 secondes. Quant à l'instance 16, dû à sa grande taille, LINGO n'a pas pu trouver la solution optimale après 72 heures d'exécution. D'autre part, HClonalSA-GCFP a atteint une meilleure solution en moins de 8 secondes, ce qui surpasse la solution de LINGO en termes de qualité de solution et de temps de calcul requis.

Pour montrer le rôle de la méthode de recherche locale dans notre algorithme, nous avons comparé les meilleurs résultats de HClonalSA-GCFP avec ceux obtenus sans intégration de la recherche locale : ClonalSA-GCFP. Le Tableau 5.8 résume les résultats obtenus. ClonalSA-GCFP a pu atteindre l'optimum global pour 12 instances. Concernant les 4 instances restantes (13-16), les résultats obtenus sont proches des solutions les plus connues. Cependant, pour améliorer la performance de l'algorithme proposé, nous avons gardé le mécanisme de recherche locale. Pour confirmer l'efficacité

TABLE 5.7 – HClonalSA -GCFP : Résultats de la comparaison entre LINGO et HClonalSA-GCFP

No.	LINGO 16.0 (B&B)				HClonalSA-GCFP			
	TC_inter	TCP	TC	CPU(s)	TC_inter	TCP	TC	CPU(s)
1	5250	54609.9	**59859.9**[a]	0.53	5250	54609.9	**59859.9**	0.04
2	0	68260.24	**68260.24**[a]	0.23	0	68260.24	**68260.24**	0.05
3	60000	1955554	**2015554**[a]	0.26	60000	1955553.8	**2015554**	0.11
4	0	258234	**258234**[a]	0.84	0	258234	**258234**	0.1
5	0	284530.7	**284530.7**[a]	0.47	0	284530.7	**284530.7**	0.14
6	2000	50436.48	**52436.48**[a]	0.78	2000	50436.48	**52436.48**	0.14
7	6000	2341.32	**8341.32**[a]	1.47	6000	2341.32	**8341.32**	0.2
8	3350	71828.74	**75178.74**[a]	31.03	3350	71828.74	**75178.74**	0.27
9	21580	80482.3	**102062.3**[a]	32.06	21580	80482.3	**102062.3**	0.56
10	4840	80249.89	**85089.89**[a]	51.37	4840	80249.89	**85089.89**	0.52
11	40125	43458.38	**83583.38**[a]	68	40125	43458.38	**83583.38**	0.6
12	191400	993056.5	**1184456**[a]	346.34	191400	993056.5	**1184456**	1.45
13	0	443.02	**443.02**[a]	65.26	0	443.02	**443.02**	0.49
14	56000	873198.3	**929198.3**	-	56000	873198.3	**929198.3**	1.63
15	6800	1127687	**1134487**[a]	428	6800	1127687	**1134487**	1.71
16	672080	4366524	5038604	-	641330	4389076.19	**5030406.19**	7.35

[a] Optimum global

de la méthode proposée, une deuxième comparaison est effectuée à base de l'écart trouvé durant les tests. La valeur de l'écart est calculée en fonction de la limite inférieure trouvée par LINGO pour chaque instance de test en utilisant l'Equation (5.23). La limite inférieure, désignée par LB, est utilisée comme critère d'optimalité des solutions. Le Tableau 5.9 et la Figure 5.2 résument les résultats des trois méthodes.

$$\%gap = \frac{TC - LB}{LB} \times 100 \tag{5.23}$$

La méthode HClonalSA-GCFP a enregistré des écarts dans deux instances où le plus élevé est égal à 11.13% (instance 16). LINGO a également des écarts dans les mêmes instances avec une valeur plus élevée dans l'instance 16. ClonalSA-GCFP a présenté des écarts dans quatre instances de référence.

D'après nos tests, notre méthode est plus efficace que ClonalSA-GCFP et LINGO dans la résolution du GCFP. Il est à noter que nous n'avons pas pu comparer nos résultats avec d'autres méthodes ayant utilisées des instances de test aléatoires afin de valider leurs modèles mathématiques faute de disponibilité de ces instances malgré nos nombreuses consultations aux auteurs concernés.

La contribution de notre travail réside dans les points suivants :

— L'adaptation de l'algorithme de sélection clonale au GCFP.

— L'amélioration de l'algorithme pour la résolution des instances de grande taille.

— La résolution du GCFP dans un temps de calcul très raisonnable en comparaison avec la méthode B&B.

TABLE 5.8 – HClonalSA -GCFP : Résultats de la comparaison entre ClonalSA-GCFP et HClonalSA-GCFP

No.	ClonalSA-GCFP				HClonalSA-GCFP			
	TC_inter	TCP	TC	CPU(s)	TC_inter	TCP	TC	CPU(s)
1	5250	54609.9	**59859.9**	0.03	5250	54609.9	**59859.9**	0.04
2	0	68260.24	**68260.24**	0.03	0	68260.24	**68260.24**	0.05
3	60000	1955554	**2015554**	0.03	60000	1955553.8	**2015554**	0.11
4	0	258234	**258234**	0.05	0	258234	**258234**	0.1
5	0	284530.7	**284530.7**	0.01	0	284530.7	**284530.7**	0.14
6	2000	50436.48	**52436.48**	0.04	2000	50436.48	**52436.48**	0.14
7	6000	2341.32	**8341.32**	0.07	6000	2341.32	**8341.32**	0.2
8	3350	71828.74	**75178.74**	0.07	3350	71828.74	**75178.74**	0.27
9	21580	80482.3	**102062.3**	0.15	21580	80482.3	**102062.3**	0.56
10	4840	80249.89	**85089.89**	0.17	4840	80249.89	**85089.89**	0.52
11	40125	43458.38	**83583.38**	0.14	40125	43458.38	**83583.38**	0.6
12	191400	993056.5	**1184456**	0.22	191400	993056.5	**1184456**	1.45
13	0	443.42	443.42	0.12	0	443.02	**443.02**	0.49
14	58200	873854.13	932054.13	0.3	56000	873198.3	**929198.3**	1.63
15	29600	1123968.69	1153568.69	0.67	6800	1127687	**1134487**	1.71
16	678000	4354675.8	5032675.8	1.8	641330	4389076.19	**5030406.19**	7.35

TABLE 5.9 – HClonalSA-GCFP : Comparaison entre LB, LINGO, ClonalSA-GCFP et HClonalSA-GCFP

No.	*LB*	LINGO	ClonalSA-GCFP	HClonalSA-GCFP
1	**59859.9**	**59859.9**	**59859.9**	**59859.9**
2	**68260.24**	**68260.24**	**68260.24**	**68260.24**
3	**2015554**	**2015554**	**2015554**	**2015554**
4	**258234**	**258234**	**258234**	**258234**
5	**284530.7**	**284530.7**	**284530.7**	**284530.7**
6	**52436.48**	**52436.48**	**52436.48**	**52436.48**
7	**8341.32**	**8341.32**	**8341.32**	**8341.32**
8	**75178.74**	**75178.74**	**75178.74**	**75178.74**
9	**102062.3**	**102062.3**	**102062.3**	**102062.3**
10	**85089.89**	**85089.89**	**85089.89**	**85089.89**
11	**83583.38**	**83583.38**	**83583.38**	**83583.38**
12	**1184456**	**1184456**	**1184456**	**1184456**
13	**443.02**	**443.02**	443.42	**443.02**
14	**920612.1**	929198.3	932054.13	929198.3
15	**1134487**	**1134487**	1153568.69	**1134487**
16	**4526388**	5038604	5032675.8	5030406.19

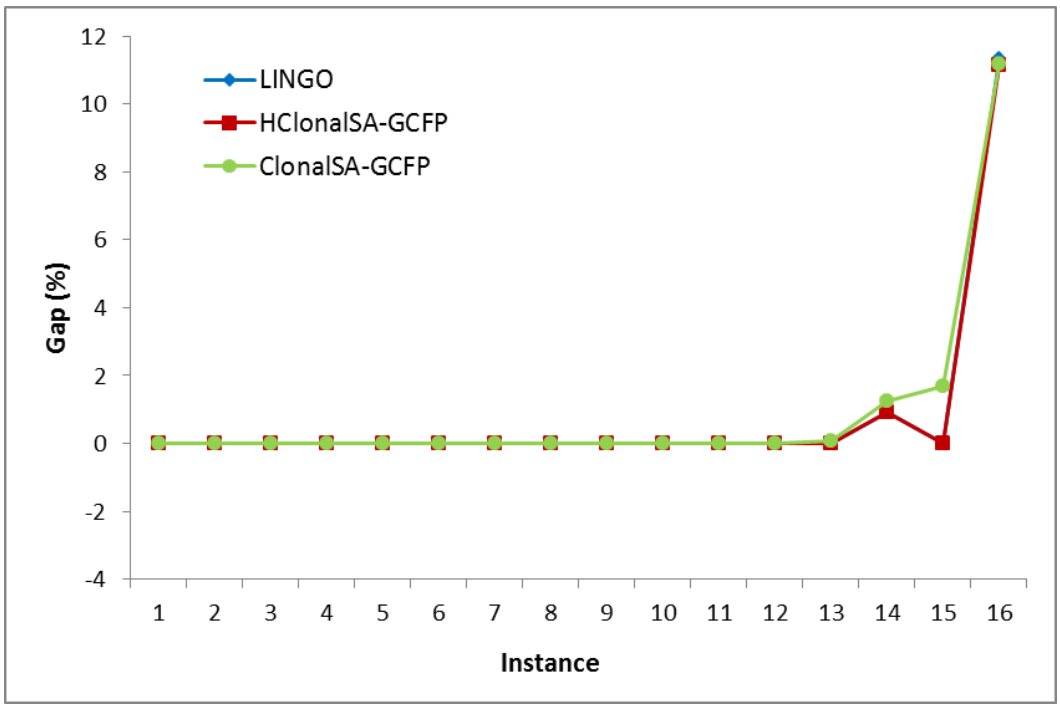


FIGURE 5.2 – HClonalSA-GCFP : Comparaison de Gap entre LINGO, ClonalSA-GCFP et HClonalSA-GCFP

5.4 Algorithme de recherche du coucou pour GCFP

Notre deuxième méthode consiste à minimiser le coût des mouvements intercellulaires et intracellulaires des produits, le coût de panne des machines et le coût d'installation en tenant compte de plusieurs facteurs de production. Cette méthode est basée sur l'algorithme de recherche du coucou et elle est notée Hybrid Cuckoo Search Algorithm-Generalized Cell Formation Problem (HCuckooSA-GCFP) (Karoum et Elbenani, 2018). Les tests sont réalisés sur un ensemble d'instances de référence. Les solutions obtenues sont comparées avec celles fournies par le logiciel LINGO. Par la suite, nous allons présenter le modèle mathématique utilisé et introduire les modifications appliquées à l'algorithme de recherche du coucou. Puis, nous allons interpréter les résultats obtenus.

5.4.1 Description du problème

Le modèle mathématique utilisé est inspiré de celui de (Jouzdani *et al.*, 2014) et il est basé sur les hypothèses introduites dans la Section 5.3.1. Ce modèle peut être considéré comme une extension du premier modèle, déjà introduit $MG2(X, Y, Z)$, par intégration des coûts des mouvements intracellulaires et d'installation. Dans ce contexte, nous avons gardé les mêmes notations avec ajout de deux notations concernant les nouveaux coûts traités :

M	: nombre de machines.
P	: nombre de produits.
C	: nombre de cellules.
R	: nombre total de routes.
V_j	: volume de production du produit j.
R_j	: nombre de routes alternatives du produit j.
L_k	: limite inférieure (lower limit) du nombre de machines dans la cellule k.
U_k	: limite supérieure (upper limit) du nombre de machines dans la cellule k.
NM_{jr}	: nombre de machines dans la route r du produit j.
$\{u_{jr}^{(1)}, u_{jr}^{(2)}, ..., u_{jr}^{(NM_{jr})}\}$	: indice de la machine dans la route r du produit j.
A_{jr}	: coût du mouvement intercellulaire du produit j dans la route r.
T_{ji}	: temps de traitement du produit j sur la machine i.
B_i	: coût de panne de la machine i.
$MTBF_i$	: temps moyen entre les pannes de la machine i.
A_j	**: coût du mouvement intracellulaire du produit j ;**
α_{jr}	**: coût d'installation de la route r du produit j.**

Le coût d'installation des routes est égale à :

$$SetupCost = \sum_{j=1}^{P} \sum_{r=1}^{R_j} Z_{jr} \alpha_{jr} \tag{5.24}$$

Le coût des mouvements intracellulaires est formulé comme suit :

$$IntracellularCost = \sum_{j=1}^{P} \sum_{r=1}^{R_j} \sum_{i=1}^{NM_{jr}-1} \sum_{k=1}^{C} A_j V_j Z_{jr} Y_{(u_{jr}^{(i)})k} Y_{\left(u_{jr}^{(i+1)}\right)k} \tag{5.25}$$

Avec :

$$Y_{ik} = \begin{cases} 1, & \text{si la machine } i \text{ est dans la cellule } k \\ 0, & \text{sinon.} \end{cases}$$

$$Z_{jr} = \begin{cases} 1, & \text{si la route } r \text{ du produit } j \text{ est sélectionnée} \\ 0, & \text{sinon.} \end{cases}$$

Pour linéariser cette équation du coût, une nouvelle variable binaire est introduite :

$$W_{jriek} = Z_{jr} \times Y_{ik} \times Y_{ek} \tag{5.26}$$

Où :

$$W_{jriek} = \begin{cases} 1, & \text{si la route } r \text{ du produit } j \text{ est sélectionnée, et} \\ & \quad \text{les machines } i \text{ et } e \text{ sont dans la cellule } k \\ 0, & \text{sinon.} \end{cases}$$

En utilisant la nouvelle variable, le coût des mouvements intracellulaires est reformulé comme suit :

$$IntracellularCost = \sum_{j=1}^{P} \sum_{r=1}^{R_j} \sum_{i=1}^{NM_{jr}-1} \sum_{k=1}^{C} A_j V_j W_{jr(u_{jr}^{(i)})(u_{jr}^{(i+1)})k} \tag{5.27}$$

Le but du modèle mathématique, noté $MG3(X,Y,Z,W)$, est de minimiser le coût des mouvements intercellulaires ($IntercellularCost$) et des mouvements intracellulaires ($IntracellularCost$), le coût de panne des machines ($BreakdownCost$) et le coût d'installation ($SetupCost$). Ce modèle est présenté comme suit :

$$\begin{aligned} MG3(X,Y,Z,W): \ \min \ TotalCost &= MG2(X,Y,Z) + IntracellularCost + SetupCost \\ &= IntercellularCost + BreakdownCost + \\ &\quad IntracellularCost + SetupCost \\ &= \sum_{j=1}^{P} \sum_{r=1}^{R_j} \sum_{i=1}^{NM_{jr}-1} \sum_{k=1}^{C} A_{jr} V_j X_{jr(u_{jr}^{(i)})k(u_{jr}^{(i+1)})k} \\ &\quad + \sum_{j=1}^{P} \sum_{r=1}^{R_j} \sum_{i=1}^{NM_{jr}} Z_{jr} \frac{V_j T_{j(u_{jr}^{(i)})} B_{(u_{jr}^{(i)})}}{MTBF_{(u_{jr}^{(i)})}} \\ &\quad + \sum_{j=1}^{P} \sum_{r=1}^{R_j} \sum_{i=1}^{NM_{jr}-1} \sum_{k=1}^{C} A_j V_j W_{jr(u_{jr}^{(i)})(u_{jr}^{(i+1)})k} \\ &\quad + \sum_{j=1}^{P} \sum_{r=1}^{R_j} Z_{jr} \alpha_{jr} \end{aligned} \tag{5.28}$$

Sous contraintes :

$$\sum_{r=1}^{R_j} Z_{jr} = 1, \quad j = 1, 2, ..., P \tag{5.29}$$

$$\sum_{i=1}^{M} Y_{ik} \leq U_k, \quad k = 1, 2, ..., C \tag{5.30}$$

$$\sum_{i=1}^{M} Y_{ik} \geq L_k, \quad k = 1, 2, ..., C \tag{5.31}$$

$$\sum_{k=1}^{C} Y_{ik} = 1, \quad i = 1, 2, ..., M \tag{5.32}$$

$$Z_{jr} + Y_{(u_{jr}^{(i)})k} + (1 - Y_{(u_{jr}^{(i+1)})k}) - 3X_{jr(u_{jr}^{(i)})k(u_{jr}^{(i+1)})k} \geq 0 \tag{5.33}$$

$$Z_{jr} + Y_{(u_{jr}^{(i)})k} + (1 - Y_{(u_{jr}^{(i+1)})k}) - X_{jr(u_{jr}^{(i)})k(u_{jr}^{(i+1)})k} \leq 2 \tag{5.34}$$

$$Z_{jr} + Y_{(u_{jr}^{(i)})k} + Y_{(u_{jr}^{(i+1)})k} - 3W_{jr(u_{jr}^{(i)})(u_{jr}^{(i+1)})k} \geq 0 \tag{5.35}$$

$$Z_{jr} + Y_{(u_{jr}^{(i)})k} + Y_{(u_{jr}^{(i+1)})k} - W_{jr(u_{jr}^{(i)})(u_{jr}^{(i+1)})k} \leq 2 \tag{5.36}$$

$$Z_{jr}, Y_{ik}, X_{jrikek}, W_{jriek} \in \{0, 1\} \tag{5.37}$$

La première partie de la fonction objectif (5.28) calcule le coût du déplacement des produits entre les différentes cellules. La deuxième partie représente le coût de panne des machines. La troisième partie calcule le coût du déplacement des produits dans la même cellule et la dernière partie calcule le coût d'installation. La contrainte (5.29) impose qu'une seule route de traitement doit être choisie pour chaque produit. Les inégalités (5.30) et (5.31) imposent respectivement les limites supérieure et inférieure pour le nombre de machines dans chaque cellule. La contrainte (5.32) indique que chaque machine sera affectée à une seule cellule. Les contraintes (5.33) et (5.35) garantissent que si au moins une des variables binaires primaires est égale à zéro, sa nouvelle variable prend également une valeur égale à zéro. Les contraintes (5.34) et (5.36) assurent que si toutes les variables primaires prennent une valeur égale à 1, leurs nouvelles variables correspondantes prennent la même valeur. Enfin, la contrainte (5.37) indique que les variables sont binaires.

5.4.2 Adaptation de HCuckooSA-GCFP au GCFP

Notre objectif est de montrer la capacité de la première approche HCuckooSA1 (Chapitre 3, Section 3.3) à résoudre le problème de formation de cellules généralisé en prenant en compte plusieurs facteurs de production tels que les routes alternatives, la fiabilité des machines, le volume et le temps de production.

Nous avons gardé le même codage présenté dans la Section 5.3.2.1 pour la représentation des solutions et nous avons généré les solutions aléatoirement comme indiqué dans la Section 5.3.2.2. Concernant l'adaptation de l'algorithme de recherche du coucou au GCFP, nous avons utilisé les modifications apportées antérieurement au vol de Lévy (Chapitre 3, Section 3.3.1.2) où la différence entre deux positions $(s_b^{(t)} - s_l^{(t)})$ présente les mouvements nécessaires pour se déplacer de la solution courante $s_l^{(t)}$ vers la meilleure solution obtenue $s_b^{(t)}$. Pour chacun des mouvements nécessaires, un nombre aléatoire entre 0 et 1 est généré ; si sa valeur est inférieure au coefficient $Coef = \frac{\mu}{|\nu|^{1/\beta}}$, le mouvement sera appliqué à la solution courante. Les étapes de la méthode sont présentées dans l'algorithme 5.4.

Algorithme 5.4 Algorithme de HCuckooSA-GCFP

Générer une population initiale de *popsize* solutions (nids)
Tant que critère d'arrêt est non satisfait **Faire**
  Générer une solution s_1 du coucou en utilisant le vol de Lévy
  Calculer sa qualité/fitness $F(s_1)$
  Appliquer la recherche locale Goncalves à la solution s_1
  Choisir au hasard une solution s_2 de la population
  Si $F(s_1) > F(s_2)$ **Alors**
    la nouvelle solution s_1 remplace la solution s_2
  Fin Si
  Abandonner une fraction p_a des mauvaises solutions dans la population et générer des nouvelles solutions
  Classer les solutions de la population
Fin Tant que

5.4.3 Résultats expérimentaux

Notre méthode est testée sur la majorité des instances indiquées dans le Tableau 5.5 avec ajout du coût des mouvements intracellulaires (C_intra) et du coût d'installation (CI). Le langage Java est également utilisé pour le codage de HCuckooSA-GCFP. Les tests sont effectués sur un ordinateur Intel ® Core^{TM} i7 cadencé à 2.50 GHz et 8 GB de RAM.

5.4.3.1 Réglage de paramètres

Nous devons déterminer les valeurs des trois paramètres suivantes : la taille de la population *popsize*, le nombre d'itérations et la fraction p_a des solutions à abandonner. Après avoir effectué des tests préliminaires, nous avons considéré trois valeurs possibles pour le nombre d'itérations (50, 100, 200), cinq valeurs pour *popsize* (15, 20, 30, 40, 50) et quatre valeurs pour p_a (15%, 20%, 25%, 30%). Les résultats ont montré que les valeurs suivantes semblent appropriées pour trouver les meilleures solutions tout en consommant un temps de calcul raisonnable :

— Taille de la population ou le nombre de nids : $popsize = 20$;

— Nombre d'itérations : 100 ;

— Fraction des solutions (nids) à abandonner : $p_a = 20\%$;

5.4.3.2 Résultats

Les résultats de la résolution des instances en utilisant la méthode B&B et notre méthode HCuckooSA-GCFP sont fournis dans le Tableau 5.10 et le Tableau 5.11, respectivement. Pour chaque instance de référence, le coût total des mouvements intercellulaires (TC_inter), le coût total des mouvements intracellulaires (TC_intra), le coût total des pannes de machines (TCP), le coût d'installation (TCI), la somme de ces coûts (TC) et le temps d'exécution (CPU(s)) sont présentés dans ces deux tableaux. Le temps d'exécution maximum de LINGO est fixé à 72 heures.

D'après le Tableau 5.10, LINGO a pu trouver l'optimum global de 10 instances. Pour les 9 premières instances, les solutions sont obtenues en moins de 5 minutes. Quant à l'instance P10, LINGO a nécessité une longue durée (plus que 6 heures) pour obtenir une solution réalisable. A propos des instances de 11 à 14, les solutions présentées dans le tableau sont atteintes après 72 heures. Malheureusement, LINGO n'a pas pu trouver des solutions réalisables pour les instances 12

TABLE 5.10 – HCuckooSA-GCFP : Résultats obtenus par LINGO

No.	TC_inter	TC_intra	TCP	TCI	TC	CPU(s)
1	24,500	0	54,609.90	3,490	82,599.90[a]	0.44
2	24,500	0	68,260.24	4,140	96,900.24[a]	0.60
3	325,200	172,000	1,985,939	7,790	2,490,929[a]	0.76
4	74,100	130,00	262,516.7	12,750	362,366.7[a]	48.49
5	0	61,950	305,578.1	8,845	376,373.1[a]	3.57
6	13,760	6,660	51,591.28	9,810	81,821.28[a]	50.80
7	42,500	49,900	2,377.99	15,210	109,988[a]	120.49
8	21,860	1,200	72,410.46	11,100	106,570.5[a]	87.13
9	41,210	13,880	82,835.29	15,085	153,010.3[a]	255.62
10	12,670	10,860	80,923.05	16,820	121,273.1[a]	24,367.49
11	3,250	0	450.11	13,760	17,460.11	-
12	261,950	0	882,864.2	21,100	1,165,914[b]	-
13	320415	0	1,250,646	33,805	1,604,866	-
14	1611400	772,815	4,372,591	53,220	6,810,026[b]	-

[a] Optimum global
[b] solution non réalisable

et 14 durant cette période.

Concernant HCuckooSA-GCFP, le Tableau 5.11 montre que notre méthode a obtenu les mêmes solutions optimales que LINGO pour les 10 premières instances en moins de 24 ms. De plus, HCuckooSA-GCFP a pris environ une seconde pour trouver des solutions similaires à celles obtenues par la méthode B&B après 72 heures d'exécution. En outre, la méthode proposée est capable de trouver des solutions meilleures et réalisables pour les instances 12 et 14 en moins de deux secondes. On peut conclure que notre méthode est plus efficace que LINGO en termes de qualité des solutions et du temps de calcul requis.

TABLE 5.11 – HCuckooSA-GCFP : Résultats obtenus par notre méthode HCuckooSA-GCFP

No.	TC_inter	TC_intra	TCP	TCI	TC	CPU(s)
1	24,500	0	54,609.90	3,490	82,599.90	0.14
2	24,500	0	68,260.24	4,140	96,900.24	0.14
3	325,200	172,000	1,985,939	7,790	2,490,929	0.15
4	74,100	13,000	262,516.7	12,750	362,366.7	0.16
5	0	61,950	305,578.1	8,845	376,373.1	0.15
6	13,760	6,660	51,591.28	9,810	81,821.28	0.17
7	42,500	49,900	2,377.99	15,210	109,988	0.17
8	21,860	1,200	72,410.46	11,100	106,570.5	0.17
9	41,210	13,880	82,835.29	15,085	153,010.3	0.20
10	12,670	10,860	80,923.05	16,820	121,273.1	0.23
11	3,250	0	450.11	13,760	17,460.11	0.21
12	231,150	0	909,694.80	21,150	1,161,994.80	0.34
13	320,415	0	1,250,646	33,805	1,604,866	0.46
14	1,575,630	480,440	4,377,346.46	53,930	6,487,346.46	1.51

TABLE 5.12 – HCuckooSA -GCFP : comparaison entre LB, LINGO et HCuckooSA-GCFP

No.	LB	LINGO	HCuckooSA-GCFP
1	82,599.90	82,599.90[a]	82,599.90
2	96,900.24	96,900.24[a]	96,900.24
3	2,490,929	2,490,929[a]	2,490,929
4	362,366.7	362,366.7[a]	362,366.7
5	376,373.1	376,373.1[a]	376,373.1
6	81,821.28	81,821.28[a]	81,821.28
7	109,988	109,988[a]	109,988
8	106,570.5	106,570.5[a]	106,570.5
9	153,010.3	153,010.3[a]	153,010.3
10	121,273.1	121,273.1[a]	121,273.1
11	15481.57	17,460.11	17,460.11
12	891690.8	1,165,914[b]	1,161,994.80
13	1160130	1,604,866	1,604,866
14	4540103	6,810,026[b]	6,487,346.46

[a] Optimum global
[b] solution non réalisable

Pour confirmer l'efficacité de la méthode proposée, la valeur de l'écart est calculée en fonction de la limite inférieure trouvée par LINGO pour chaque instance en utilisant l'Equation (5.23). La limite inférieure désignée par LB est utilisée comme critère d'optimalité des solutions. Le Tableau 5.12 et la Figure 5.3 résument les résultats obtenus. Ces résultats ont montré la supériorité de notre méthode par rapport à la méthode B&B.

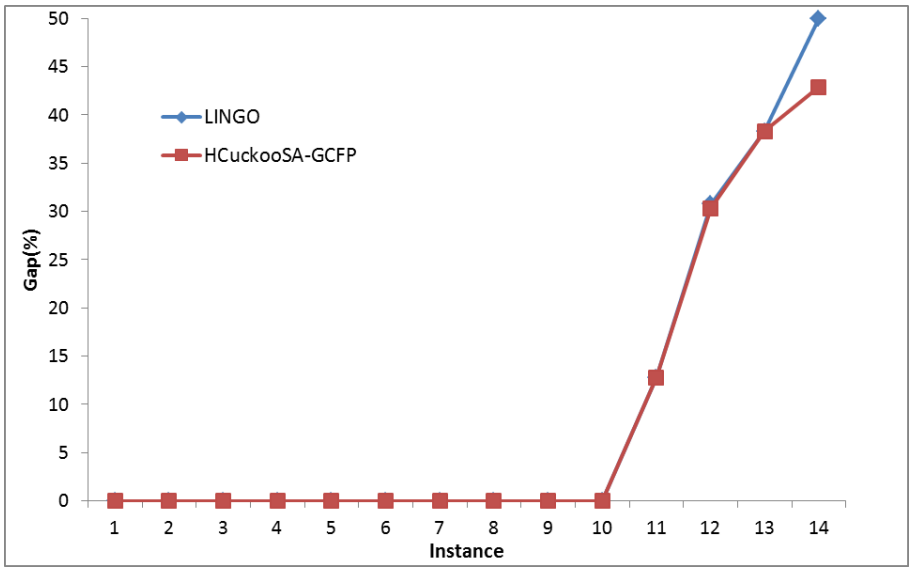


FIGURE 5.3 – HCuckooSA-GCFP : Comparaison de Gap entre LINGO et HCuckooSA-GCFP

Malheureusement, nous n'avons pas pu comparer nos résultats avec d'autres travaux puisque la majorité des auteurs génèrent les instances de façon aléatoire afin de s'adapter à leurs modèles mathématiques. Nous avons donc implémenté la méthode de recuit simulé (SA) proposée par Jouzdani *et al.* (2014) pour résoudre le même modèle mathématique. Les résultats obtenus sont résumés dans

le Tableau 5.13 avec ceux de la méthode B&B et notre méthode HCuckooSA-GCFP. Comme indiqué dans le tableau, la méthode SA a pu atteindre l'optimum global de 8 instances. En ce qui concerne les autres instances, notre méthode a trouvé de meilleurs résultats que SA dans un temps de calcul très court. HCuckooSA-GCFP a pu résoudre toutes les instances en moins de deux secondes tandis que la méthode SA a nécessité environ trois minutes. La Figure 5.4 présente le temps de calcul requis par ces deux méthodes. Les résultats obtenus ont montré la supériorité de notre méthode en termes de qualité de solutions et de temps CPU.

TABLE 5.13 – HCuckooSA-GCFP : Résultats de la comparaison entre LINGO, SA et HCuckooSA-GCFP

No.	LINGO		SA		HCuckooSA-GCFP	
	TC	CPU(s)	TC	CPU(s)	TC	CPU(s)
1	**82,599.90**	0.44	**82,599.90**	0.44	**82,599.90**	0.14
2	**96,900.24**	0.60	**96,900.24**	1.15	**96,900.24**	0.14
3	**2,490,929**	0.76	**2,490,929**	5.24	**2,490,929**	0.15
4	**362,366.7**	48.49	**362,366.7**	16.13	**362,366.7**	0.16
5	**376,373.1**	3.57	**376,373.1**	43.40	**376,373.1**	0.15
6	**81,821.28**	50.80	**81,821.28**	48.30	**81,821.28**	0.17
7	**109,988**	120.49	110,048	52.55	**109,988**	0.17
8	**106,570.5**	87.13	**106,570.5**	53.24	**106,570.5**	0.17
9	**153,010.3**	255.62	**153,010.3**	55.62	**153,010.3**	0.20
10	**121,273.1**	24,367.49	123,615.11	58.16	**121,273.1**	0.23
11	**17,460.11**	-	17,510.29	84.21	**17,460.11**	0.21
12	1,165,914	-	1,269,515.27	110.20	**1,161,994.80**	0.34
13	**1,604,866**	-	1,737,596.21	112.11	**1,604,866**	0.46
14	6,810,026	-	6,882,847.64	143.41	**6,487,346.46**	1.51

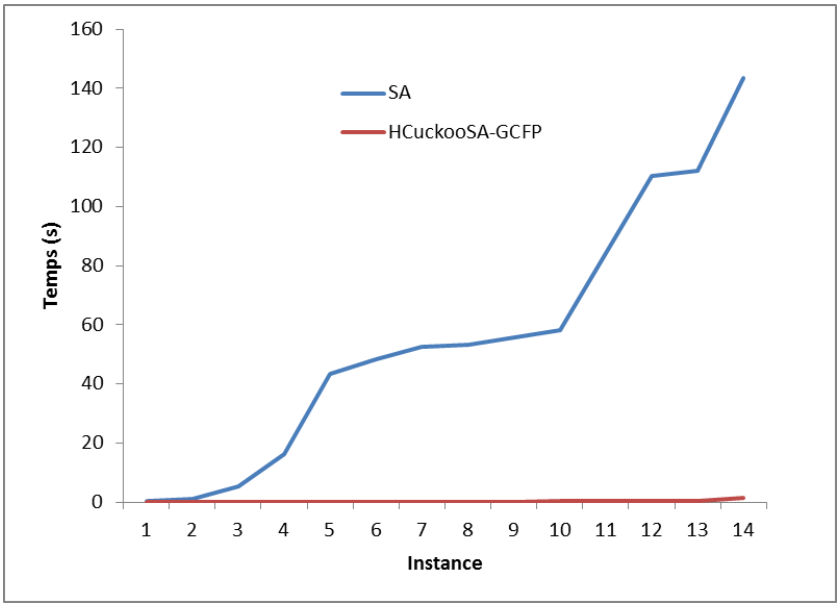


FIGURE 5.4 – HCuckooSA-GCFP : Comparaison de temps entre SA et HCuckooSA-GCFP

5.5 Comparaison entre HClonalSA-GCFP et HCuckooSA-GCFP

L'objectif principal de notre travail est de montrer la capacité de nos approches à résoudre le problème de formation de cellules en prenant en compte plusieurs facteurs de production. Dans ce contexte, nous avons appliqué la méthode HClonalSA-GCFP, déjà développée en Section 5.3 pour la résolution du modèle $MG2(X, Y, Z)$, pour résoudre le modèle mathématique $MG3(X, Y, Z, W)$. Ce dernier modèle vise à minimiser le coût des mouvements intercellulaires et des mouvements intracellulaires, le coût de panne des machines et le coût d'installation.

Nous avons testé HClonalSA-GCFP sur un ensemble d'instances. Les résultats obtenus sont présentés dans le Tableau 5.14. Pour chaque instance, le coût total des mouvements intercellulaires (TC_inter), le coût total des mouvements intracellulaires (TC_intra), le coût total des pannes de machines (TCP), le coût d'installation (TCI), la somme de ces coûts (TC) et le temps d'exécution (CPU(s)) sont rapportés dans ce tableau.

Table 5.14 – HClonalSA-GCFP : Résultats obtenus par la méthode HClonalSA-GCFP

No.	TC_inter	TC_intra	TCP	TCI	TC	CPU(s)
1	24,500	0	54,609.90	3,490	82,599.90	0.02
2	24,500	0	68,260.24	4,140	96,900.24	0.02
3	325,200	172,000	1,985,939	7,790	2,490,929	0.03
4	74,100	13,000	262,516.7	12,750	362,366.7	0.12
5	0	61,950	305,578.1	8,845	376,373.1	0.04
6	13,760	6,660	51,591.28	9,810	81,821.28	0.06
7	42,500	49,900	2,377.99	15,210	109,988	0.15
8	21,860	1,200	72,410.46	11,100	106,570.5	0.17
9	41,210	13,880	82,835.29	15,085	153,010.3	0.19
10	12,670	10,860	80,923.05	16,820	121,273.1	0.25
11	3,250	0	450.11	13,760	17,460.11	0.30
12	231,150	0	909,694.80	21,150	1,161,994.80	0.5
13	320,415	0	1,250,646	33,805	1,604,866	0.52
14	1,575,630	480,440	4,377,346.46	53,930	6,487,346.46	1.8

Pour évaluer la qualité des solutions atteintes, nous avons comparé les résultats obtenus avec ceux enregistrés par la méthode HCuckooSA-GCFP et par LINGO. Le Tableau 5.15 montre le coût total et le temps d'exécution trouvé par chaque méthode. D'après ce tableau, la méthode HClonalSA-GCFP a obtenu les mêmes solutions que HCuckooSA-GCFP pour 14 instances en moins de 2 secondes. Elle a pu également atteindre les solutions optimales pour les 10 premières instances. De plus, elle a trouvé des solutions meilleures et réalisables pour les instances 12 et 14 en moins de deux secondes. Pour résumer, les résultats obtenus lors des tests réalisés ont confirmé encore une fois la bonne performance de nos deux méthodes en termes de qualité des solutions et de temps d'exécution.

TABLE 5.15 – Résultats de la comparaison entre HClonalSA-GCFP, HCuckooSA-GCFP et LINGO

No.	LINGO		HCuckooSA-GCFP		HClonalSA-GCFP	
	TC	CPU(s)	TC	CPU(s)	TC	CPU(s)
1	82,599.90	0.44	82,599.90	0.14	82,599.90	0.02
2	96,900.24	0.60	96,900.24	0.14	96,900.24	0.02
3	2,490,929	0.76	2,490,929	0.15	2,490,929	0.03
4	362,366.7	48.49	362,366.7	0.16	362,366.7	0.12
5	376,373.1	3.57	376,373.1	0.15	376,373.1	0.04
6	81,821.28	50.80	81,821.28	0.17	81,821.28	0.06
7	109,988	120.49	109,988	0.17	109,988	0.15
8	106,570.5	87.13	106,570.5	0.17	106,570.5	0.17
9	153,010.3	255.62	153,010.3	0.20	153,010.3	0.19
10	121,273.1	24,367.49	121,273.1	0.23	121,273.1	0.25
11	17,460.11	-	17,460.11	0.21	17,460.11	0.30
12	1,165,914	-	1,161,994.80	0.34	1,161,994.80	0.5
13	1,604,866	-	1,604,866	0.46	1,604,866	0.52
14	6,810,026	-	6,487,346.46	1.51	6,487,346.46	1.8

5.6 Conclusion

L'intégration de plusieurs facteurs de production rend le problème de formation de cellules plus réaliste, mais aussi plus complexe. Dans ce chapitre, nous avons proposé deux méthodes basées sur l'algorithme de sélection clonale et l'algorithme de recherche du coucou pour résoudre le problème avec la présence des routes alternatives et de la fiabilité des machines. Le but est d'optimiser le coût total de production.

Les méthodes proposées sont évaluées à l'aide d'un ensemble d'instances de référence. Les résultats obtenus sont comparés à ceux produits par l'algorithme B&B, en utilisant le logiciel LINGO. Les résultats de la comparaison justifient la bonne performance des deux méthodes en termes de qualité des solutions et du temps de calcul requis. Malheureusement, nous n'avons pas pu comparer nos résultats avec d'autres travaux puisque la majorité des auteurs génèrent les instances de test de façon aléatoire afin de s'adapter à leurs modèles mathématiques.

Pour les futures recherches, une extension des méthodes pourrait être appliquée à d'autres variantes du problème de formation de cellules en tenant compte des facteurs supplémentaires tels que les coûts d'affectation des ressources humaines.

Conclusion générale

Synthèse

Dans cette thèse, nous avons étudié le problème de formation de cellules de production qui est considéré comme le premier problème rencontré pour réaliser un système de production cellulaire. Notre travail est essentiellement basé sur la proposition de méthodes approchées permettant de résoudre efficacement trois variantes du problème abordé. Nos approches sont des adaptations de deux métaheuristiques inspirées de la nature : l'algorithme de sélection clonale et l'algorithme de recherche du coucou. À notre meilleure connaissance, nous sommes les premiers à utiliser ces deux algorithmes pour résoudre le problème. Chaque approche proposée est développée différemment pour s'adapter à la variante correspondante du problème.

La première variante présente la version simple du problème avec un nombre de cellules fixe à l'avance. Nous avons proposé deux approches basées sur l'algorithme de sélection clonale pour résoudre cette variante. La première approche (HClonalSA1) combine les caractéristiques de l'algorithme avec une méthode de recherche Tabou pour intensifier la recherche autour des régions prometteuses. La deuxième approche (HClonalSA2) présente une nouvelle façon pour utiliser l'algorithme de sélection clonale en intégrant d'autres types d'opérateurs génétiques et un mécanisme de recherche (Recherche Goncalves) visant à renforcer la qualité des solutions obtenues. De plus, nous avons adapté l'algorithme de recherche du coucou de deux manières différentes en lien avec le processus de vol de Lévy (HCuckooSA1 et HCuckooSA2). Les approches développées sont évaluées sur un ensemble d'instances qui varie entre 30 et 35 instances. Les résultats de tests ont montré que nos approches sont comparables, meilleures dans certains cas à un ensemble d'approches récentes tirées de la littérature.

La deuxième variante du problème traite le cas où le nombre de cellules est variable. La différence majeure entre cette variante et la première variante réside dans la possibilité d'ajouter ou de supprimer des cellules. Pour résoudre cette variante, nous avons introduit une nouvelle approche basée sur l'algorithme de sélection clonale (HClonalSA-VNC). Nous avons évalué cette approche sur 40 instances de référence. Les résultats obtenus ont montré la bonne performance de notre approche puisqu'elle a pu atteindre les meilleures solutions connues pour 97.5% des instances. Une étude comparative entre HClonalSA-VNC et d'autres algorithmes a confirmé l'efficacité de notre approche.

La troisième variante (problème de formation de cellules généralisé) prend en compte la possibilité d'avoir plusieurs routes de traitement pour chaque produit et d'intégrer différents facteurs de production. L'intégration de ces facteurs rend le problème plus réaliste mais aussi plus complexe. Nous avons appliqué deux méthodes pour former les cellules en considérant les nouvelles contraintes de cette troisième variante. La première méthode est basée sur l'algorithme de sélection clonale (HClonalSA-GCFP) et elle vise à minimiser le coût des mouvements intercellulaires et à maximiser la fiabilité des machines. La deuxième méthode est une adaptation de l'algorithme de recherche du coucou (HCuckooSA-GCFP) et elle est basée sur les mêmes idées appliquées dans l'approche (HCuckooSA1) tout en tenant compte des nouvelles spécificités du problème. Le but est de montrer la capacité de nos approches à résoudre d'autres variantes du problème de formation de cellules. HCuckooSA-GCFP consiste à minimiser le coût des mouvements intercellulaires et intracellulaires, le coût de panne des machines et le coût d'installation. Les deux approches proposées sont testées sur un ensemble d'instances. Les résultats obtenus sont comparés avec une méthode Branch and Bound, en utilisant le logiciel LINGO. Les résultats de la comparaison ont confirmé la bonne performance de nos méthodes en termes de qualité des solutions et du temps de calcul requis.

Perspectives

Suite aux résultats obtenus par nos méthodes, nous avons constaté qu'il y'a une possibilité d'étendre notre travail pour résoudre d'autres variantes du problème de formation de cellules telles que le problème de formation cellulaire dynamique où des changements périodiques sont prévus au niveau des demandes de produits. En outre, le rôle important joué par les employés dans l'exécution des tâches sur les machines rend leur affectation aux cellules un facteur important pour une utilisation complète des systèmes de fabrication cellulaire. Nous pouvons donc étudier ce cas en tenant compte, par exemple, du type, des capacités et du temps disponible pour chaque employé, et ainsi que du salaire et du nombre des travailleurs alloués.

Bien que l'utilisation des métaheuristiques permette de réduire considérablement le temps de calcul, ce dernier peut augmenter pour les instances de grande taille. Nos futures recherches peuvent se concentrer également sur le développement des versions parallèles de nos méthodes pour résoudre le problème de formation de cellules et d'autres problèmes d'optimisation combinatoire plus particulièrement les problèmes à grande échelle. Le but est d'améliorer à la fois la qualité des solutions et le temps d'exécution.

Une autre piste à suivre est celle du Smart Manufacturing (SM - littéralement fabrication intelligente) qui représente une approche technologique utilisant des machines-outils connectées à Internet pour surveiller le processus de production. Son déploiement implique d'intégrer des capteurs aux machines afin de collecter des données sur leur état de fonctionnement et leurs performances. Dans ce contexte nous visons une étude approfondie.

Annexe

A Algorithme de sélection clonale pour Team Orienteering Problem

A.1 Introduction

Dans cette annexe, nous décrivons un nouveau problème d'optimisation combinatoire connu par Team orienteering problem ou le problème d'orientation d'équipe. Ce problème est considéré comme une variante du célèbre problème de tournées de véhicules. Notre but est de montrer la capacité de nos approches à résoudre d'autres problèmes. Dans ce contexte, la méthode proposée pour résoudre le Team Orienteering Problem (Karoum et Elbenani, 2016) est basée sur les mêmes idées appliquées dans l'approche HClonalSA2 (Chapitre 2, Section 2.4) tout en tenant compte des nouvelles spécificités du problème. Au début de cette annexe, nous commençons par définir le problème, ses applications et l'état d'art des méthodes proposées pour le résoudre. Ensuite, nous décrivons en détail la méthode proposée. Puis, nous rapportons les résultats obtenus et nous les comparons avec les meilleures solutions connues dans la littérature.

A.2 Team Orienteering Problem TOP

Le problème d'orientation de l'équipe ou team orienteering problem (TOP) consiste à déterminer un ensemble de chemins disjoints partants tous d'un même point de départ vers la même destination dans un intervalle de temps spécifié. Le but principal est de maximiser le profit total reçu par visiter un sous-ensemble d'emplacements en respectant le temps donné.

Ce problème a été décrit pour la première fois par (Butt et Cavalier, 1994) sous le nom du problème du voyageur de commerce sélectif (STSP). Plus tard, Chao *et al.* (1996) ont introduit son nom actuel (team orienteering problem) et ils ont créé un ensemble d'instances utilisées actuellement comme instances de référence pour le TOP.

Le TOP est une extension du problème d'orientation (OP) où l'équipe ne possède qu'un seul membre. Il est également considéré comme une variante du problème de tournées de véhicules (VRP) pour lequel chaque véhicule n'est pas forcé de visiter tous les clients qui lui sont affectés. Étant une extension de ces problèmes, TOP est un problème NP difficile (Chao *et al.*, 1996).

Soit une équipe de m membres et un ensemble de n lieux (points ou clients). Chaque lieu a un profit associé. Les points de départ et d'arrivée ont un profit nul. Le temps de voyage t_{ij}, entre les lieux i et j, est connu à l'avance. Une fois qu'un membre de l'équipe atteint un lieu et collecte le profit, les membres restants ne pourraient en bénéficier. Il est difficile de visiter tous les lieux car le temps disponible donné à l'équipe est limité à un temps spécifié T_{max}. Par conséquent, l'équipe tente de visiter les endroits où le gain total peut être maximisé autant que possible.

La Figure A.1 montre un exemple résolu de TOP avec trois membres qui suivent trois chemins disjoints et qui doivent partir du dépôt D vers l'arrivée A en maximisant le score récolté lors de la visite d'un lieu. Dans notre exemple, on a 14 lieux au total, mais certains d'entre eux ne seront pas visités, cela est dû à la limitation du temps de parcours et par le fait que chaque lieu doit être visité au plus une seule fois.

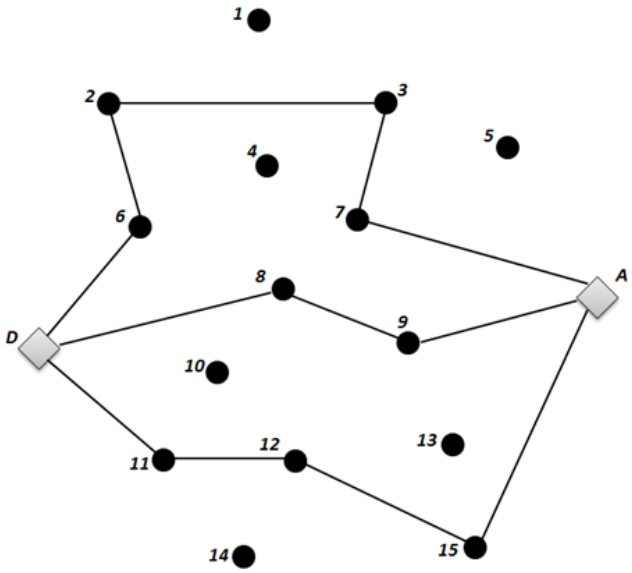


FIGURE A.1 – TOP : Exemple avec une équipe de trois membres

Plusieurs applications du TOP ont été publiées dans la littérature, y compris le recrutement d'athlètes (Butt et Cavalier, 1994), la version multi-véhicules du problème de livraison de carburant domestique (Chao *et al.*, 1996), le routage des techniciens pour le service clientèle (Tang et Miller-Hooks, 2005) et la planification de voyage touristique (Vansteenwegen *et al.*, 2009).

A.2.1 État de l'art pour TOP

Un nombre limité d'approches exactes ont été appliquées dans la littérature pour résoudre le TOP. Une approche a été introduite par (Butt et Ryan, 1999), qui ont utilisé une technique basée sur la génération de colonnes et des techniques de stockage des lieux visités pour améliorer le temps.

Dang *et al.* (2013a) ont introduit un algorithme Branch-and-Cut pour résoudre le problème. Cet algorithme est basé sur une formulation linéaire avec un nombre polynomial de variables binaires. Durant la résolution du TOP, les auteurs ont défini un ensemble de propriétés de dominance et d'inégalités valides.

Keshtkaran *et al.* (2016) ont développé un algorithme de Branch-and-Price pour trouver des solutions optimales du TOP. Cet algorithme est basé sur celui proposé par (Boussier *et al.*, 2007) en intégrant des nouvelles fonctionnalités, telles que la méthode utilisée pour résoudre le sous-problème du prix.

Grâce à l'efficacité des métaheuristiques dans la résolution des problèmes d'optimisation combinatoire, plusieurs auteurs ont utilisé ces algorithmes pour résoudre le TOP, nous pouvons citer, par exemple :

Tang et Miller-Hooks (2005) qui ont développé une méthode intégrant la recherche Tabou dans une procédure de mémoire adaptative pour résoudre le TOP. Cette méthode alterne entre la réalisation de petits et de grands mouvements pendant la phase d'amélioration de la solution ce qui permet l'évolution efficace de la recherche sans dégradation significative de la qualité de la solution.

Archetti *et al.* (2007) qui ont utilisé deux algorithmes de recherche Tabou (TS avec stratégie de pénalité et TS avec stratégie réalisable) et deux versions d'un algorithme de recherche du voisinage variable (VNS rapide et VNS lent). Les algorithmes proposés ont pu améliorer la solution la plus connue de 128 instances de référence.

Ke *et al.* (2008) ont appliqué l'approche d'optimisation des colonies de fourmis ACO pour résoudre TOP. De plus, ils ont utilisé quatre méthodes pour construire les solutions : les méthodes séquentielles, déterministes, aléatoires et simultanées.

Bouly *et al.* (2010) ont proposé un algorithme mémétique pour résoudre le problème. Dans cet algorithme, les auteurs ont combiné l'algorithme génétique avec des techniques de recherche locale et ils ont utilisé une procédure de division optimale pour l'évaluation des solutions.

Dang *et al.* (2013b) ont développé une nouvelle méthode basée sur l'optimisation par essaims particulaires PSO pour le TOP. La contribution principale de leur méthode demeure dans la rapidité du processus d'évaluation qui permet d'examiner un plus grand nombre de solutions voisines et d'explorer plus rapidement l'espace de recherche en un temps de calcul raisonnable.

Expósito *et al.* (2016) ont modélisé le problème de conception des voyages touristiques comme étant un problème d'orientation d'équipe dans le but de maximiser le nombre de points importants à visiter. Ils ont également proposé une méthode basée sur la procédure GRASP pour résoudre ce problème avec un bon mécanisme de construction des solutions et une méthode de recherche locale pour améliorer la qualité des solutions.

Pour plus de détails sur les méthodes proposées pour le problème d'orientation d'équipe TOP, on peut référer le lecteur au (Gunawan *et al.*, 2016).

A.2.2 Formulation mathématique

Le but du problème d'orientation d'équipe TOP est de construire M chemins qui maximisent le profit en respectant la contrainte du temps de chaque chemin. Formellement, ce problème peut être défini sous forme d'un graphe $G = (V, E)$, où $V = \{1, \ldots, n\}$ est l'ensemble des sommets qui représentent les lieux et $E = \{(i, j) | i, j \in V\}$ est l'ensemble des arcs connectant les lieux.
Chaque sommet $i \in V$ est associé à un profit r_i. Tous les chemins doivent commencer au sommet 1 et se terminer au sommet n. Le temps nécessaire pour passer du sommet i à j est connu pour chaque paire de sommets. Pour chaque trajet, le temps de déplacement ne peut pas dépasser une limite de temps prédéfinie T_{max}.
Les variables de décision permettant de modéliser le problème d'orientation d'équipe sont les suivantes :

$$x_{ijk} = \begin{cases} 1, & \text{si le sommet } i \text{ est visité avant le sommet } j \text{ dans le chemin } k \\ 0, & \text{sinon} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{si } i\text{-ième sommet est visité dans le chemin } k \\ 0, & \text{sinon} \end{cases}$$

Le modèle mathématique utilisé pour ce problème est celui proposé par (Tang et Miller-Hooks, 2005) :

$$f(x,y) = \max \sum_{k=1}^{m} \sum_{i=2}^{n-1} r_i y_{ik} \tag{A.1}$$

Sous contraintes :

$$\sum_{k=1}^{m} \sum_{j=2}^{n-1} x_{1jk} = \sum_{k=1}^{m} \sum_{i=2}^{n-1} x_{ink} = m \tag{A.2}$$

$$\sum_{i<j} x_{ijk} + \sum_{i>j} x_{jik} = 2y_{jk} \ (j = 2, ..., n-1; k = 1, ..., m) \tag{A.3}$$

$$\sum_{k=1}^{m} y_{ik} \leq 1 \ (i = 2, ..., n-1) \tag{A.4}$$

$$\sum_{i=1}^{n-1} \sum_{j>i} t_{ij} x_{ijk} \leq T_{max} \ (k = 1, ..., m) \tag{A.5}$$

$$\sum_{i,j \in U;\ i<j} x_{ijk} \leq |U| - 1 \ (U \subset V\{1,n\}; 2 \leq |U| \leq n-2) \tag{A.6}$$

$$x_{ijk}, y_{jk} \in \{0,1\} \ \forall i,j = 1, ..., n; \ \forall k = 1, ..., m \tag{A.7}$$

La fonction objectif (A.1) maximise le profit total récolté. La contrainte (A.2) garantit que chaque chemin ou tour commence au sommet de départ et se termine au sommet d'arrivée. La contrainte (A.3) assure la connectivité de chaque chemin. La contrainte (A.4) garantit que chaque sommet est visité au plus une seule fois (sauf les sommets 1 et n). La contrainte (A.5) garantit que la limitation de temps de chaque chemin n'est pas violée. La contrainte (A.6) élimine les sous-chemins. Enfin, la contrainte (A.7) indique que les variables sont binaires.

A.3 Algorithme de sélection clonale pour TOP

A.3.1 Adaptation de ClonalSA au TOP

L'adaptation de ClonalSA au TOP nécessite la spécification de certains éléments : la représentation d'une solution, la génération d'une population initiale, les équations de la sélection clonale, l'opérateur du mutation appliqué et la méthode de recherche locale utilisée.

A.3.1.1 Représentation d'une solution

Considérons l'exemple suivant pour expliquer la représentation de la solution adoptée dans notre méthode. Le Tableau A.1 présente une instance TOP avec 21 sommets à visiter par deux membres d'équipe en temps maximal égale à $T_{max} = 7.5$. La tournée commence par le sommet 1 et se termine par le sommet 21. Pour chaque sommet ou lieu, le Tableau A.1 représente les coordonnées (x, y) et le profit r (reward).

Une solution valide de cette instance est présentée sous forme d'un chromosome avec deux chemins commençant par l'emplacement 1 et se terminant par l'emplacement 21 :

$$s = \begin{cases} (1, 12, 11, 13, 14, 21) & \text{:chemin 1 pour le membre 1} \\ (1, 6, 7, 21) & \text{:chemin 2 pour le membre 2} \end{cases}$$

TABLE A.1 – Instance de TOP avec deux chemins et 21 lieux

No.	x	y	r	No.	x	y	r
1	4.600	7.100	0	12	5.400	8.200	10
2	5.700	11.400	20	13	5.800	6.800	10
3	4.400	12.300	20	14	6.700	5.800	25
4	2.800	14.300	30	15	13.800	13.100	40
5	3.200	10.300	15	16	14.100	14.200	40
6	3.500	9.800	15	17	11.200	13.600	30
7	4.400	8.400	10	18	9.700	16.400	30
8	7.800	11.000	20	19	9.500	18.800	50
9	8.800	9.800	20	20	4.700	16.800	30
10	7.700	8.200	20	21	5.000	5.600	0
11	6.300	7.900	15				

Le temps de déplacement de 1 à 21 pour chaque membre est égal à la somme des distances entre les différents emplacements visités tout au long de son chemin. Le temps enregistré par le premier membre est égal à 6.57 (= 1.36 + 0.94 + 1.21 + 1.35 + 1.71) et le temps de parcours du second est égal à 7.44 (= 2.91 + 1.66 + 2.86). Le score total de la solution proposée est la somme des scores collectés par les deux membres et il est égal à 85 (= 60 + 25).

A.3.1.2 Population initiale

Les solutions initiales sont générées d'une manière aléatoire. La création d'un nouveau chromosome aléatoire est équivalente à la création d'un groupe d'itinéraires ou de chemins aléatoires valides. Tous ces itinéraires comprennent les points de départ et d'arrivée.

Pour former un chemin, les points sont ajoutés à partir d'une liste de disponibilité commune à tous les chemins. La tentative d'ajouter un point à un itinéraire n'est possible que si elle ne dépasse pas le temps maximal donné. Une fois le point est ajouté à un chemin, il est ensuite retiré de la liste de disponibilité et marqué comme visité. Un chromosome doit contenir autant de chemins que le nombre de membres de l'équipe.

A.3.1.3 Sélection clonale

La valeur d'affinité d'une solution s est basée sur la fonction objectif $f(x, y)$, de l'Equation (A.1), telle que représentée dans l'Equation (A.8).

$$\mathit{Affinite}(s) = \frac{1}{f(x,y)} \tag{A.8}$$

Puisque le but de la fonction objectif est de maximiser le profit donc les solutions ayant la plus petite affinité dans la population seront sélectionnées pour proliférer en plusieurs clones ou copies. Le nombre de clones d'une solution s est calculé en fonction de sa valeur d'affinité, de la taille de la population (*popsize*) et de la somme des affinités de l'ensemble de la population (Chapitre 2, Equation (2.2)).

A.3.1.4 Mutation

La mutation par échange est appliquée aux clones générés. Elle consiste à supprimer un point aléatoire d'un chemin choisi au hasard de la solution. Ensuite, tenter d'insérer un ou plusieurs points parmi ceux de la liste de disponibilité. Les points disponibles sont vérifiés un par un dans un ordre aléatoire.

Par exemple, soit s la solution actuelle :

$$s = \begin{cases} (1, 12, 11, 13, 14, 21) & \text{:chemin 1} \\ (1, 6, \mathbf{7}, 21) & \text{:chemin 2} \end{cases}$$

Le point 7 du chemin 2 est choisi aléatoirement pour être supprimé. Un test est effectué pour choisir d'autres points parmi la liste de disponibilité qui inclut les points non-visités : 2, 3, 4, 5, 8, 9, 10, 15, 16, 17, 18, 19 et 20. Ces points sont testés un par un dans un ordre aléatoire.
s' est obtenue en supprimant le point 7 parmi les points visités par le deuxième membre, puis en ajoutant les deux points 5 et 3.

$$s' = \begin{cases} (1, 12, 11, 13, 14, 21) & \text{:chemin 1} \\ (1, 6, \mathbf{5}, \mathbf{3}, 21) & \text{:chemin 2} \end{cases}$$

A.3.1.5 Recherche locale

Pour intensifier la recherche vers des meilleures solutions, un mécanisme de recherche locale est intégré dans notre méthode. La génération de nouvelles solutions voisines est basée sur l'application des mouvements d'insertion. Elle consiste à tester toutes les insertions possibles des points non-visités entre deux points consécutifs dans la solution. Seules les insertions qui ne violent aucune contrainte sont prises en considération.

A.3.1.6 Algorithme du ClonalSA-TOP

La méthode proposée est notée ClonalSA-TOP (*Clonal Selection Algorithm - Team Orienteering Problem*) et elle est basée sur l'algorithme de sélection clonale. Chaque itération commence par sélectionner un pourcentage b des solutions avec les meilleures affinités. Ensuite, les solutions sélectionnées se multiplient en clones. Ces derniers subissent un opérateur de mutation, et les meilleurs d'entre eux remplacent les solutions déjà sélectionnées pour la prolifération. Ensuite, un mécanisme de recherche locale basé sur la recherche Goncalves (voir Chapitre 2, Section 2.4.1.2) est appliqué aux solutions obtenues par l'opérateur de croisement. Enfin, un processus de receptor editing est effectué en remplaçant un pourcentage w des mauvaises solutions dans la population par des nouvelles solutions. L'algorithme A.1 décrit notre méthode.

Algorithme A.1 Algorithme de ClonalSA-TOP

Etape 1 : Génération de la population initiale ;

Etape 2 : **Tant que** critère d'arrêt est non satisfait **Faire** :

1. Calcul de l'affinité de chaque solution en utilisant l'Equation (2.1) du Chapitre 2 ;
2. Sélection d'un pourcentage b de solutions de la population ayant les meilleures affinités ;
3. Sélection clonale : le processus de génération des copies de ces meilleures solutions est effectué en fonction de la valeur d'affinité, la taille de la population (*popsize*) et la somme des affinités de la population (Chapitre 2, Equation (2.2)).
4. Application de la mutation sur les copies des solutions générées ;
5. Application de la recherche locale sur les solutions obtenues par le croisement ;
6. Remplacement d'un pourcentage w des solutions, ayant des mauvaises affinités, de la population.

Fin Tant que

A.3.2 Résultats expérimentaux

L'algorithme proposé est testé sur un ensemble d'instances de référence collectées de la littérature. La taille de ces instances varie de petite à grande afin d'évaluer la qualité de l'algorithme.

La méthode proposée ClonalSA-TOP est codée en Java. Les tests sont réalisés sur un ordinateur Pentium ® Dual Core CPU T4500 cadencé à 2.30 GHz et 2 GB de RAM.
Les résultats obtenus, après plusieurs tests, ont indiqué que les valeurs des paramètres suivantes sont appropriés pour notre méthode : une taille de population égale à 20, 500 itérations, un pourcentage b des solutions choisies pour le processus de clonage égal à 20%, et un pourcentage w des mauvaises solutions à remplacer égal à 20%.

Le Tableau A.2 présente le nombre de lieux ou de points (n), le nombre de membres de l'équipe (m) et le temps maximal T_{max} de chaque instance de référence. En outre, la meilleure solution connue dans la littérature, la meilleure solution atteinte par notre méthode, et le temps de calcul requis en secondes sont rapportés dans ce tableau.

Les résultats présentés dans ce tableau montrent que la méthode proposée a pu atteindre les meilleures solutions connues de la majorité des instances étudiées, sauf cinq instances où les solutions obtenues sont très proches des meilleures solutions connues. Concernant le temps de calcul nécessaire pour trouver la meilleure solution, la méthode proposée est capable de résoudre ces instances en moins d'une seconde. Les résultats obtenus montrent le bon fonctionnement de l'algorithme du point de vue de la qualité des solutions et du temps de calcul raisonnable.

TABLE A.2 – ClonalSA-TOP : Résultats expérimentaux

Instance	n	m	T_{max}	Best Sol.	ClonalSA-TOP	Temps (s)	Instance	n	m	T_{max}	Best Sol.	ClonalSA-TOP	Temps (s)
p1.2.l	32	2	30.0	**195**	185	0.187	p3.2.c	33	2	12.5	**180**	**180**	0.172
P1.2.m	32	2	32.5	**215**	200	0.187	p3.3.b	33	3	6.7	**90**	**90**	0.140
p1.3.k	32	3	18.3	**135**	**135**	0.187	p3.3.c	33	3	8.3	**120**	**120**	0.141
p1.4.k	32	4	13.8	**100**	**100**	0.187	p3.3.d	33	3	10.0	**170**	**170**	0.187
p2.2.a	21	2	7.5	**90**	**90**	0.109	p3.3.e	33	3	11.7	**200**	**200**	0.187
p2.2.b	21	2	10	**120**	**120**	0.125	p3.3.f	33	3	13.3	**230**	**230**	0.203
p2.2.c	21	2	11.5	**140**	**140**	0.124	p3.3.h	33	3	16.7	**300**	**300**	0.203
p2.2.d	21	2	12.5	**160**	**160**	0.125	p3.4.d	33	4	7.5	**100**	**100**	0.141
p2.2.f	21	2	15.0	**200**	**200**	0.125	p3.4.e	33	4	8.8	**140**	**140**	0.156
p2.2.g	21	2	16.0	**200**	**200**	0.125	p3.4.f	33	4	10.0	**190**	**190**	0.187
p2.2.h	21	2	17.5	**230**	**230**	0.109	p3.4.g	33	4	11.2	**220**	**220**	0.202
p2.3.a	21	3	5.0	**70**	**70**	0.109	p3.4.h	33	4	12.5	**240**	**240**	0.203
p2.3.b	21	3	6.7	**70**	**70**	0.109	p3.4.i	33	4	13.8	**270**	**270**	0.203
p2.3.c	21	3	7.7	**105**	**105**	0.125	p3.4.m	33	4	18.8	**390**	**390**	0.203
p2.3.d	21	3	8.3	**105**	**105**	0.109	p5.2.f	66	2	15.0	**240**	**240**	0.359
p2.3.e	21	3	9.0	**120**	**120**	0.110	p5.4.f	66	4	7.5	**80**	**80**	0.312
p2.3.f	21	3	10.0	**120**	**120**	0.124	p5.4.g	66	4	8.8	**140**	**140**	0.390
p2.3.g	21	3	10.7	**145**	**145**	0.125	p5.4.h	66	4	10.0	**140**	**140**	0.405
p2.3.i	21	3	12.7	**200**	**200**	0.109	p6.2.d	64	2	15.0	**192**	**192**	0.343
p2.3.j	21	3	13.3	**200**	**200**	0.125	p6.2.e	64	2	17.5	**360**	342	0.421
p2.3.k	21	3	15.0	**200**	**200**	0.125	p6.2.f	64	2	20.0	**588**	582	0.490
p2.4.b	21	4	5.0	**70**	**70**	0.109	p6.3.g	64	3	15.0	**282**	270	0.437
p2.4.f	21	4	7.5	**105**	**105**	0.125	p7.2.c	102	2	30.0	**101**	**101**	0.374
p2.4.g	21	4	8.0	**105**	**105**	0.109	p7.3.d	102	3	26.7	**117**	**117**	0.452
p2.4.h	21	4	8.8	**120**	**120**	0.125	p7.3.e	102	3	33.3	**175**	**175**	0.578
p2.4.i	21	4	9.5	**120**	**120**	0.125	p7.4.d	102	4	20.0	**79**	**79**	0.359
p2.4.j	21	4	10.0	**120**	**120**	0.125	p7.4.e	102	4	25.0	**123**	**123**	0.452
p2.4.k	21	4	11.2	**180**	**180**	0.125	p7.4.f	102	4	30.0	**164**	**164**	0.624
p3.2.b	33	2	10.0	**150**	**150**	0.156							

A.3.3 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode basée sur l'algorithme de sélection clonale pour résoudre le Team Orienteering Problem. À notre connaissance, cet algorithme est appliqué pour la première fois au TOP. Les résultats obtenus ont montré l'efficacité de la méthode proposée puisqu'elle a pu atteindre les meilleures solutions connues pour 91.22% des instances tout en consommant un temps de calcul raisonnable.

Pour les futures recherches, une extension de l'approche pourrait être appliquée à d'autres variantes du Team Orienteering Problem en tenant compte des facteurs supplémentaires tels que les fenêtres temporelles multiples (*multiple time windows*) et les capacités des membres de l'équipe.

Productions scientifiques

Articles acceptés

Karoum Bouchra, El Khattabi Noussaima, Elbenani Bouazza and El Imrani Ameur Abdelhakim, "An efficient artificial immune system algorithm for the cell formation problem", Journal of Computational Methods in Sciences and Engineering (2016) : vol. 16, no 4, pp. 733-744, IOS Press.

Karoum Bouchra and Elbenani Bouazza, "A hybrid clonal algorithm for the cell formation problem with variant number of cells", Production Engineering (2017) : vol. 11, no 1, pp. 19-28, Springer.

Karoum Bouchra and Elbenani Youssef Bouazza, "A clonal selection algorithm for the generalized cell formation problem considering machine reliability and alternative routings", Production Engineering (2017) : vol. 11, no 4-5, pp. 545-556, Springer.

Karoum Bouchra and Elbenani Youssef Bouazza, "Optimization of the Material Handling Costs and the Machine Reliability in Cellular Manufacturing System Using Cuckoo search Algorithm", Neural Computing and Applications (2018), Springer. https ://doi.org/10.1007/s00521-017-3302-3

Articles en révision

Karoum Bouchra and Elbenani Youssef Bouazza, "Discrete cuckoo search algorithm for solving the cell formation problem (submitted)", International Journal of Manufacturing Research (2017), Inderscience.

Conférences internationales

Karoum Bouchra, Bouazza Elbenani, and El Imrani Abdelhakim Ameur, "Clonal selection algorithm for the cell formation problem", Mediterranean Conference on Information & Communication Technologies (MedICT'2015), May 7-9, 2015.

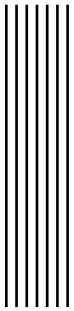
Karoum Bouchra, Elbenani Bouazza, El Imrani Abdelhakim Ameur and El Khattabi Noussaima, "Manufacturing Cell Formation Problem using Hybrid Cuckoo Search Algorithm", 11th Metaheuristics International Conference (MIC'15), June 7-10, 2015.

Karoum Bouchra and Elbenani Bouazza, "Clonal selection algorithm for the team orienteering problem", In : 2016 IEEE 11th International Conference on Intelligent Systems : Theories and Applications (SITA), doi : 10.1109/SITA.2016.7772274, Oct 19-20, 2016.

Chapitres de livres

Karoum Bouchra, Elbenani Bouazza, and El Imrani Abdelhakim Ameur (2016) "Clonal Selection Algorithm for the Cell Formation Problem. " In : El Oualkadi A., Choubani F., El Moussati A. (eds) Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015. Lecture Notes in Electrical Engineering, vol 380. Springer, Cham.

Karoum Bouchra, Elbenani Bouazza, El Khattabi Noussaima and El Imrani Ameur Abdelhakim, (2018) "Manufacturing Cell Formation Problem Using Hybrid Cuckoo Search Algorithm." In : Amodeo L., Talbi EG., Yalaoui F. (eds) Recent Developments in Metaheuristics. Operations Research/Computer Science Interfaces Series, vol 62. Springer, Cham.



Bibliographie

Aarts E. et Korst J. : *Simulated annealing and boltzmann machines. A stochastic approach to combinatorial optimization and neural computing*. New York, NY ; John Wiley and Sons Inc., 1990.

Abdelaziz A. Y. et Ali E. S. : Cuckoo search algorithm based load frequency controller design for nonlinear interconnected power system. *International Journal of Electrical Power & Energy Systems*, 73:632–643, 2015.

Adenso-Díaz B., Lozano S. et Eguía I. : Part-machine grouping using weighted similarity coefficients. *Computers & Industrial Engineering*, 48(3):553–570, 2005.

Alhourani F. : Clustering algorithm for solving group technology problem with multiple process routings. *Computers & Industrial Engineering*, 66(4):781–790, 2013.

Alhourani F. : Cellular manufacturing system design considering machines reliability and parts alternative process routings. *International Journal of Production Research*, 54(3):846–863, 2016.

Ameli M. S. J. et Arkat J. : Cell formation with alternative process routings and machine reliability consideration. *The International Journal of Advanced Manufacturing Technology*, 35(7-8):761–768, 2008.

Ameli M. S. J., Arkat J. et Barzinpour F. : Modelling the effects of machine breakdowns in the generalized cell formation problem. *The International Journal of Advanced Manufacturing Technology*, 39(7-8):838–850, 2008a.

Ameli M. S. J., Arkat J. et Sakri M. S. : Applying metaheuristics in the generalized cell formation problem considering machine reliability. *Journal of the Chinese Institute of Industrial Engineers*, 25(4):261–274, 2008b.

Anderberg M. R. : Cluster analysis for applications. monographs and textbooks on probability and mathematical statistics, 1973.

Andres C. et Lozano S. : A particle swarm optimization algorithm for part–machine grouping. *Robotics and Computer-Integrated Manufacturing*, 22(5):468–474, 2006.

Anvari M., Mehrabad M. S. et Barzinpour F. : Machine–part cell formation using a hybrid particle swarm optimization. *The International Journal of Advanced Manufacturing Technology*, 47(5): 745–754, 2010.

Archetti C., Hertz A. et Speranza M. G. : Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, 2007.

Arkat J., Hosseini L. et Farahani M. H. : Minimization of exceptional elements and voids in the cell formation problem using a multi-objective genetic algorithm. *Expert Systems with Applications*, 38(8):9597–9602, 2011.

Arkat J., Saidi M. et Abbasi B. : Applying simulated annealing to cellular manufacturing system design. *The International Journal of Advanced Manufacturing Technology*, 32(5-6):531–536, 2007.

Ashayeri J., Heuts R. et Tammel B. : A modified simple heuristic for the p-median problem, with facilities design applications. *Robotics and computer-integrated manufacturing*, 21(4):451–464, 2005.

Askin R. G. et Standridge C. R. : *Modeling and analysis of manufacturing systems*. John Wiley & Sons Inc, 1993.

Askin R. G. et Subramantan S. P. : A cost-based heuristic for group technology configuration. *International Journal of Production Research*, 25(1):101–113, 1987.

Ateme-Nguema B. et Dao T. : Quantized hopfield networks and tabu search for manufacturing cell formation problems. *International Journal of Production Economics*, 121(1):88–98, 2009.

Ateme-Nguema B. H. et Dao T. M. : Optimization of cellular manufacturing systems design using the hybrid approach based on the ant colony and tabu search techniques. *In IEEE International Conference on Industrial Engineering and Engineering Management, 2007*, pages 668–673. IEEE, 2007.

Attila Islier A. : Group technology by an ant system algorithm. *International Journal of Production Research*, 43(5):913–932, 2005.

Balamurugan R., Natarajan A. M. et Premalatha K. : A modified harmony search method for biclustering microarray gene expression data. *International Journal of Data Mining and Bioinformatics*, 16(4):269–289, 2016.

Barthelemy P., Bertolotti J. et Wiersma D. S. : A lévy flight for light. *Nature*, 453:495–498, 2008.

Bhide P., Bhandwale A. et Kesavadas T. : Cell formation using multiple process plans. *Journal of Intelligent manufacturing*, 16(1):53–65, 2005.

Boctor F. F. : A linear formulation of the machine-part cell formation problem. *International Journal of Production Research*, 29(2):343–356, 1991.

Boe W. J. et Cheng C. H. : A close neighbour algorithm for designing cellular manufacturing systems. *International Journal of Production Research*, 29(10):2097–2116, 1991.

Bouly H., Dang D. C. et Moukrim A. : A memetic algorithm for the team orienteering problem. *4OR : A Quarterly Journal of Operations Research*, 8(1):49–70, 2010.

Boussier S., Feillet D. et Gendreau M. : An exact algorithm for team orienteering problems. *4OR : A Quarterly Journal of Operations Research*, 5(3):211–230, 2007.

Brown E. C. et Sumichrast R. T. : Cf-gga : a grouping genetic algorithm for the cell formation problem. *International Journal of Production Research*, 39(16):3651–3669, 2001.

Butt S. E. et Cavalier T. M. : A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1):101–111, 1994.

Butt S. E. et Ryan D. M. : An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, 26(4):427–441, 1999.

Bychkov I. et Batsyn M. : An efficient exact model for the cell formation problem with a variable number of production cells. *arXiv preprint arXiv :1701.02472*, 2017.

Bychkov I., Batsyn M. et Pardalos P. M. : Exact model for the cell formation problem. *Optimization Letters*, 8(8):2203–2210, 2014.

Carrie A. S. : Numerical taxonomy applied to group technology and plant layout. *International Journal of Production Research*, 11(4):399–416, 1973.

Caux C., Bruniaux R. et Pierreval H. : Cell formation with alternative process plans and machine capacity constraints : A new combined approach. *International Journal of Production Economics*, 64(1):279–284, 2000.

Chaine S. et Tripathy M. : Design of an optimal smes for automatic generation control of two-area thermal power system using cuckoo search algorithm. *Journal of Electrical Systems and Information Technology*, 2(1):1–13, 2015.

Chan H. M. et Milner D. A. : Direct clustering algorithm for group formation in cellular manufacture. *Journal of Manufacturing Systems*, 1(1):65–75, 1982.

Chandrasekharan M. P. et Rajagopalan R. : An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. *International Journal of Production Research*, 24(2):451–463, 1986a.

Chandrasekharan M. P. et Rajagopalan R. : Modroc : an extension of rank order clustering for group technology. *International Journal of Production Research*, 24(5):1221–1233, 1986b.

Chandrasekharan M. P. et Rajagopalan R. : Zodiac—an algorithm for concurrent formation of part-families and machine-cells. *International Journal of Production Research*, 25(6):835–850, 1987.

Chandrasekharan M. P. et Rajagopalan R. : Groupability : an analysis of the properties of binary data matrices for group technology. *International Journal of Production Research*, 27(6):1035–1052, 1989.

Chao I. M., Golden B. L. et Wasil E. A. : The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996.

Chen W.-H. et Srivastava B. : Simulated annealing procedures for forming machine cells in group technology. *European Journal of Operational Research*, 75(1):100–111, 1994.

Chow W. S. et Hawaleshka O. : An efficient algorithm for solving the machine chaining problem in cellular manufacturing. *Computers & industrial engineering*, 22(1):95–100, 1992.

Chung S. H., Wu T. H. et Chang C. C. : An efficient tabu search algorithm to the cell formation problem with alternative routings and machine reliability considerations. *Computers & Industrial Engineering*, 60(1):7–15, 2011.

Chung T. P. et Liao C. J. : An immunoglobulin-based artificial immune system for solving the hybrid flow shop problem. *Applied Soft Computing*, 13(8):3729–3736, 2013.

Clover F. et Laguna M. : *Tabu search*. Kluwer Academic Publishers, 1997.

Dang D. C., El-Hajj R. et Moukrim A. : A branch-and-cut algorithm for solving the team orienteering problem. *In International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 332–339. Springer, 2013a.

Dang D. C., Guibadj R. N. et Moukrim A. : An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229(2):332–344, 2013b.

Daniel E. et Anitha J. : Optimum wavelet based masking for the contrast enhancement of medical images using enhanced cuckoo search algorithm. *Computers in biology and medicine*, 71:149–155, 2016.

Das K., Lashkari R. S. et Sengupta S. : Reliability considerations in the design of cellular manufacturing systems : A simulated annealing-based approach. *International Journal of Quality & Reliability Management*, 23(7):880–904, 2006.

De Castro L. N. et Von Zuben F. J. : The clonal selection algorithm with engineering applications. *In Proceedings of GECCO*, volume 2000, pages 36–39, 2000.

De Castro L. N. et Von Zuben F. J. : Learning and optimization using the clonal selection principle. *IEEE transactions on evolutionary computation*, 6(3):239–251, 2002.

Deutsch S. J., Freeman S. F. et Helander M. : Manufacturing cell formation using an improved p-median model. *Computers & industrial engineering*, 34(1):135–146, 1998.

Diabat A., Kannan D., Kaliyan M. et Svetinovic D. : An optimization model for product returns using genetic algorithms and artificial immune system. *Resources, conservation and recycling*, 74:156–169, 2013.

Diaby M. et Nsakanda A. L. : Large-scale capacitated part-routing in the presence of process and routing flexibilities and setup costs. *Journal of the Operational Research Society*, 57(9):1100–1112, 2006.

Díaz J. A., Luna D. et Luna R. : A grasp heuristic for the manufacturing cell formation problem. *Top*, 20(3):679–706, 2012.

Dimopoulos C. et Zalzala A. M. : Recent developments in evolutionary computation for manufacturing optimization : problems, solutions, and comparisons. *IEEE transactions on evolutionary computation*, 4(2):93–113, 2000.

Dinis-Carvalho J., Alves A. C. et Sousa R. M. : Moving from job-shop to production cells without losing flexibility : A case study from the wooden frames industry. *South African Journal of Industrial Engineering*, 25(3):212–225, 2014.

Durgun İ. et Yildiz A. R. : Structural design optimization of vehicle components using cuckoo search algorithm. *Materials Testing*, 54(3):185–188, 2012.

Elbenani B. et Ferland J. A. : *Cell formation problem solved exactly with the dinkelbach algorithm*, volume 7. CIRRELT, 2012.

Elbenani B., Ferland J. A. et Bellemare J. : Genetic algorithm and large neighbourhood search to solve the cell formation problem. *Expert Systems with Applications*, 39(3):2408–2414, 2012.

Engin O. et Döyen A. : A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future generation computer systems*, 20(6):1083–1095, 2004.

Expósito A., Brito J. et Moreno J. A. : A heuristic-biased grasp for the team orienteering problem. *In Conference of the Spanish Association for Artificial Intelligence*, pages 428–437. Springer, 2016.

Forrest S., Perelson A. S., Allen L. et Cherukuri R. : Self-nonself discrimination in a computer. *In IEEE Computer Society Symposium on Research in Security and Privacy, 1994.*, pages 202–212. IEEE, 1994.

Foulds L. R., French A. P. et Wilson J. M. : The sustainable cell formation problem : manufacturing cell creation with machine modification costs. *Computers & Operations Research*, 33(4):1010–1032, 2006.

Gao X. Z., Wang X. et Ovaska S. J. : Fusion of clonal selection algorithm and differential evolution method in training cascade–correlation neural network. *Neurocomputing*, 72(10):2483–2490, 2009.

Gherboudj A., Layeb A. et Chikhi S. : Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm. *International Journal of Bio-Inspired Computation*, 4(4):229–236, 2012.

Goldengorin B., Krushinsky D. et Pardalos P. M. : *Cell Formation in Industrial Engineering*. Springer, 2013.

Gonçalves J. F. et Resende M. G. C. : An evolutionary algorithm for manufacturing cell formation. *Computers & industrial engineering*, 47(2):247–273, 2004.

Guerrero F., Lozano S., Smith K. A., Canca D. et Kwok T. : Manufacturing cell formation using a new self-organizing neural network. *Computers & Industrial Engineering*, 42(2):377–382, 2002.

Gunawan A., Lau H. C. et Vansteenwegen P. : Orienteering problem : A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

Hachicha W. : *Nouvelles approches pour la formation des cellules de production dans le cadre d'une démarche de conception*. Thèse de doctorat, Ecole Nationale d'Ingénieurs de Sfax, 2009.

Hofmeyr S. A. et Forrest S. : Architecture for an artificial immune system. *Evolutionary computation*, 8(4):443–473, 2000.

Hu L. et Yasuda K. : Minimising material handling cost in cell formation with alternative processing routes by grouping genetic algorithm. *International Journal of Production Research*, 44(11):2133–2167, 2006.

Hung K. T. et Maleki H. : Applying group technology to the forging industry. *Production Planning & Control*, 25(2):134–148, 2014.

Islam K. M. S. et Sarker B. R. : A similarity coefficient measure and machine-parts grouping in cellular manufacturing systems. *International Journal of Production Research*, 38(3):699–720, 2000.

James T. L., Brown E. C. et Keeling K. B. : A hybrid grouping genetic algorithm for the cell formation problem. *Computers & Operations Research*, 34(7):2059–2079, 2007.

Jayaswal S. et Adil G. K. : Efficient algorithm for cell formation with sequence data, machine replications and alternative process routings. *International Journal of Production Research*, 42 (12):2419–2433, 2004.

Jerne N. K. : Towards a network theory of the immune system. *Ann. Immunol.*, 125:373–389, 1974.

Jia B., Yu B., Wu Q., Yang X., Wei C., Law R. et Fu S. : Hybrid local diffusion maps and improved cuckoo search algorithm for multiclass dataset analysis. *Neurocomputing*, 189:106–116, 2016.

Johnson D. J. et Wemmerlöv U. : Why does cell implementation stop? factors influencing cell penetration in manufacturing plants. *Production and Operations Management*, 13(3):272–289, 2004.

Joines J. A., Culbreth C. T. et King R. E. : Manufacturing cell design : an integer programming model employing genetic algorithms. *IIE transactions*, 28(1):69–85, 1996.

Jouzdani J., Barzinpour F., Shafia M. A. et Fathian M. : Applying simulated annealing to a generalized cell formation problem considering alternative routings and machine reliability. *Asia-Pacific Journal of Operational Research*, 31(04):1450021, 2014.

Kamalakannan R., Sudhakara Pandian R., Sornakumar T. et Mahapatra S. S. : An ant colony optimization algorithm for cellular manufacturing system. *In Applied Mechanics and Materials*, volume 854, pages 133–141. Trans Tech Publ, 2017.

Karoum B., El Khattabi N., Elbenani B. et El Imrani A. A. : An efficient artificial immune system algorithm for the cell formation problem. *Journal of Computational Methods in Sciences and Engineering*, 16(4):733–744, 2016a.

Karoum B. et Elbenani B. : Clonal selection algorithm for the team orienteering problem. *In 11th International Conference on Intelligent Systems : Theories and Applications (SITA), 2016*, pages 1–5. IEEE, 2016.

Karoum B. et Elbenani B. : A hybrid clonal algorithm for the cell formation problem with variant number of cells. *Production Engineering*, 11(1):19–28, 2017a.

Karoum B., Elbenani B. et El Imrani A. A. : Clonal selection algorithm for the cell formation problem. *In Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015*, pages 319–326. Springer, 2016b.

Karoum B., Elbenani B., El Khattabi N. et El Imrani A. A. : Manufacturing cell formation problem using hybrid cuckoo search algorithm. *In Recent Developments in Metaheuristics*, pages 151–162. Springer, 2018.

Karoum B. et Elbenani Y. B. : A clonal selection algorithm for the generalized cell formation problem considering machine reliability and alternative routings. *Production Engineering*, 11 (4-5):545–556, 2017b.

Karoum B. et Elbenani Y. B. : Discrete cuckoo search algorithm for solving the cell formation problem (submitted). *International Journal of Manufacturing Research*, 2017c.

Karoum B. et Elbenani Y. B. : Optimization of the material handling costs and the machine reliability in cellular manufacturing system using cuckoo search algorithm. *Neural Computing and Applications*, 2018.

Kashan A. H., Karimi B. et Noktehdan A. : A novel discrete particle swarm optimization algorithm for the manufacturing cell formation problem. *The International Journal of Advanced Manufacturing Technology*, 73(9-12):1543–1556, 2014.

Kashyap A., Agarwal M. et Gupta H. : Detection of copy-move image forgery using svd and cuckoo search algorithm. *arXiv preprint arXiv :1704.00631*, 2017.

Ke L., Archetti C. et Feng Z. : Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54(3):648–665, 2008.

Keeling K. B., Brown E. C. et James T. L. : Grouping efficiency measures and their impact on factory measures for the machine-part cell formation problem : A simulation study. *Engineering Applications of Artificial Intelligence*, 20(1):63–78, 2007.

Keshtkaran M., Ziarati K., Bettinelli A. et Vigo D. : Enhanced exact solution methods for the team orienteering problem. *International Journal of Production Research*, 54(2):591–601, 2016.

King J. R. : Machine-component grouping in production flow analysis : an approach using a rank order clustering algorithm. *International Journal of Production Research*, 18(2):213–232, 1980.

King J. R. et Nakornchai V. : Machine-component group formation in group technology : review and extension. *International Journal of Production Research*, 20(2):117–133, 1982.

Kirkpatrick S., Gelatt C. D. et Vecchi M. P. : Optimization by simulated annealing. *science*, 220 (4598):671–680, 1983.

Kumar K. R., Kusiak A. et Vannelli A. : Grouping of parts and components in flexible manufacturing systems. *European Journal of Operational Research*, 24(3):387–397, 1986.

Kumar K. R. et Vannelli A. : Strategic subcontracting for efficient disaggregated manufacturing. *BEBR faculty working paper ; no. 1252*, 1986.

Kumari A. et Shukla S. : Distributed generation allocation and voltage improvement in distribution system using cuckoo search algorithm. *International Journal of Engineering Science and Technology*, 7(9):298, 2015.

Kuo R. J., Chen C. M., Liao T. W. et Tien F. C. : Hybrid of artificial immune system and particle swarm optimization-based support vector machine for radio frequency identification-based positioning system. *Computers & Industrial Engineering*, 64(1):333–341, 2013.

Kusiak A. : The generalized group technology concept. *International journal of production research*, 25(4):561–569, 1987.

Kusiak A. : *Computational intelligence in design and manufacturing*. John Wiley & Sons, 2000.

Kusiak A. et Cho M. : Similarity coefficient algorithms for solving the group technology problem. *International Journal Of Production Research*, 30(11):2633–2646, 1992.

Kusiak A. et Chow W. S. : Efficient solving of the group technology problem. *Journal of manufacturing systems*, 6(2):117–124, 1987.

Landsbergis P. A., Cahill J. et Schnall P. : The impact of lean production and related new systems of work organization on worker health. *Journal of occupational health psychology*, 4(2):108, 1999.

Lee M. K., Luong H. S. et Abhary K. : A genetic algorithm based cell design considering alternative routing. *Computer Integrated Manufacturing Systems*, 10(2):93–108, 1997.

Levasseur G. A., Helms M. M. et Zink A. A. : A conversion from a functional to a cellular manufacturing layout at steward, inc. *Production and Inventory Management Journal*, 36(3):37, 1995.

Li T. : An immunity based network security risk estimation. *Science in China Series F : Information Sciences*, 48(5):557–578, 2005.

Li X., Baki M. F. et Aneja Y. P. : An ant colony optimization metaheuristic for machine–part cell formation problems. *Computers & Operations Research*, 37(12):2071–2081, 2010.

Lima F. P. A., Lotufo A. D. P. et Minussi C. R. : Wavelet-artificial immune system algorithm applied to voltage disturbance diagnosis in electrical distribution systems. *IET Generation, Transmission & Distribution*, 9(11):1104–1111, 2015.

Liu C. M. et Wu J. K. : Machine cell formation : using the simulated annealing algorithm. *International Journal of Computer Integrated Manufacturing*, 6(6):335–349, 1993.

Liu R., Zhang X., Yang N., Lei Q. et Jiao L. : Immunodomaince based clonal selection clustering algorithm. *Applied Soft Computing*, 12(1):302–312, 2012.

Logendran R. et Karim Y. : Design of manufacturing cells in the presence of alternative cell locations and material transporters. *Journal of the Operational Research Society*, 54(10):1059–1075, 2003.

Lozano S., Dobado D., Larrañeta J. et Onieva L. : Modified fuzzy c-means algorithm for cellular manufacturing. *Fuzzy Sets and Systems*, 126(1):23–32, 2002.

Mahdavi I., Paydar M. M., Solimanpur M. et Heidarzade A. : Genetic algorithm approach for solving a cell formation problem in cellular manufacturing. *Expert Systems with Applications*, 36 (3):6598–6604, 2009.

Majumder A. et Laha D. : A new cuckoo search algorithm for 2-machine robotic cell scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation*, 28:131–143, 2016.

Masnata A. et Settineri L. : An application of fuzzy clustering to cellular manufacturing. *International Journal of Production Research*, 35(4):1077–1094, 1997.

McAuley J. : Machine grouping for efficient production. *Production Engineer*, 51(2):53–57, 1972.

McCormick W. T., Schweitzer P. J. et White T. W. : Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972.

Ming L. C. et Ponnambalam S. G. : A hybrid ga/pso for the concurrent design of cellular manufacturing system. *In IEEE International Conference on Systems, Man and Cybernetics, 2008. SMC 2008.*, pages 1855–1860. IEEE, 2008.

Mitrofanov S. P. : *Scientific Principles of Group Technology :(Nauchnye Osnovy Gruppovoĭ Tekhnologii)*. National Lending Library for Science and Technology, 1966.

Montreuil B. : Fractal layout organization for job shop environments. *International Journal of Production Research*, 37(3):501–521, 1999.

Moon C. et Gen M. : A genetic algorithm-based approach for design of independent manufacturing cells. *International Journal of Production Economics*, 60:421–426, 1999.

Moon Y. B. et Chi S. C. : Generalized part family formation using neural network techniques. *Journal of Manufacturing Systems*, 11(3):149–159, 1992.

Mosier C. et Taube L. : The facets of group technology and their impacts on implementation - a state-of-the-art survey. *Omega*, 13(5):381–391, 1985a.

Mosier C. et Taube L. : Weighted similarity measure heuristics for the group technology machine clustering problem. *Omega*, 13(6):577–579, 1985b.

Muller K. E. et Fetterman B. A. : *Regression and ANOVA : an integrated approach using SAS software*. SAS Institute, 2002.

Muruganandam A., Prabhaharan G., Asokan P. et Baskaran V. : A memetic algorithm approach to the cell formation problem. *The International Journal of Advanced Manufacturing Technology*, 25(9):988–997, 2005.

Nagi R., Harhalakis G. et Proth J. M. : Multiple routeings and capacity considerations in group technology applications. *International Journal of Production Research*, 28(12):1243–1257, 1990.

Ng S. M. : Worst-case analysis of an algorithm for cellular manufacturing. *European Journal of Operational Research*, 69(3):384–398, 1993.

Nguyen T. T. et Truong A. V. : Distribution network reconfiguration for power loss minimization and voltage profile improvement using cuckoo search algorithm. *International Journal of Electrical Power & Energy Systems*, 68:233–242, 2015.

Noktehdan A., Karimi B. et Kashan A. H. : A differential evolution algorithm for the manufacturing cell formation problem using group based operators. *Expert Systems with Applications*, 37(7): 4822–4829, 2010.

Noktehdan A., Seyedhosseini S. et Saidi-Mehrabad M. : A metaheuristic algorithm for the manufacturing cell formation problem based on grouping efficacy. *The International Journal of Advanced Manufacturing Technology*, 82(1-4):25–37, 2016.

Onwubolu G. C. et Songore V. : A tabu search approach to cellular manufacturing systems. *Production Planning & Control*, 11(2):153–164, 2000.

Ouaarab A., Ahiod B., Yang X. S. et Abbad M. : Discrete cuckoo search algorithm for job shop scheduling problem. *In 2014 IEEE International Symposium on Intelligent Control (ISIC)*, pages 1872–1876. IEEE, 2014.

Ouk Kim C., Baek J. G. et Baek J. K. : A two-phase heuristic algorithm for cell formation problems considering alternative part routes and machine sequences. *International Journal of Production Research*, 42(18):3911–3927, 2004.

Pailla A., Trindade A. R., Parada V. et Ochi L. S. : A numerical comparison between simulated annealing and evolutionary approaches to the cell formation problem. *Expert Systems with Applications*, 37(7):5476–5483, 2010.

Papaioannou G. et Wilson J. M. : Fuzzy extensions to integer programming models of cell-formation problems in machine scheduling. *Annals of Operations Research*, 166(1):163–181, 2009.

Pare S., Kumar A., Bajaj V. et Singh G. K. : A multilevel color image segmentation technique based on cuckoo search algorithm and energy curve. *Applied Soft Computing*, 47:76–102, 2016.

Park S. et Suresh N. C. : Performance of fuzzy art neural network and hierarchical clustering for part-machine grouping based on operation sequences. *International Journal of Production Research*, 41(14):3185–3216, 2003.

Pattanaik L. N. et Sharma B. P. : Implementing lean manufacturing with cellular layout : a case study. *The International Journal of Advanced Manufacturing Technology*, 42(7):772–779, 2009.

Pavlyukevich I. : Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, 226(2):1830–1844, 2007.

Paydar M. M., Mahdavi I., Valipoor Khonakdari S. et Solimanpur M. : Developing a mathematical model for cell formation in cellular manufacturing systems. *International Journal of Operational Research*, 11(4):408–424, 2011.

Paydar M. M. et Saidi-Mehrabad M. : A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. *Computers & Operations Research*, 40 (4):980–990, 2013.

Pham D. T., Afify A. et Koc E. : Manufacturing cell formation using the bees algorithm. *In Innovative Production Machines and Systems Virtual Conference, Cardiff, UK*, 2007.

Prabhaharan G., Asokan P., Girish B. S. et Muruganandam A. : Machine cell formation for cellular manufacturing systems using an ant colony system approach. *The International Journal of Advanced Manufacturing Technology*, 25(9):1013–1019, 2005.

Puntura A., Theera-Umpon N. et Auephanwiriyakul S. : Optimizing support vector machine parameters using cuckoo search algorithm via cross validation. *In 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), 2016*, pages 102–107. IEEE, 2016.

Rajagopalan R. et Batra J. L. : Design of cellular production systems a graph-theoretic approach. *International Journal of Production Research*, 13(6):567–579, 1975.

Ramakrishnan B., Sreedivya S. R. et Selvi M. : Adaptive routing protocol based on cuckoo search algorithm (arp-cs) for secured vehicular ad hoc network (vanet). *International Journal of computer networks and applications (IJCNA)*, 2(4):173–178, 2015.

Raposo E. P., Buldyrev S. V., Da Luz M. G. E., Viswanathan G. M. et Stanley H. E. : Lévy flights and random searches. *Journal of Physics A : mathematical and theoretical*, 42(43):434003, 2009.

Ravichandran K. S. et Rao K. C. S. : A new approach to fuzzy part-family formation in cellular manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 18 (8):591–597, 2001.

Reynolds A. M. et Frye M. A. : Free-flight odor tracking in drosophila is consistent with an optimal intermittent scale-free search. *PloS one*, 2(4):e354, 2007.

Rezazadeh H. et Khiali-Miab A. : A two-layer genetic algorithm for the design of reliable cellular manufacturing systems. *International Journal of Industrial Engineering Computations*, 8(3):315–332, 2017.

Rodrigues L. C. A. et Weller T. R. : Cell formation with alternative routings and capacity considerations : A hybrid tabu search approach. *In Mexican International Conference on Artificial Intelligence*, pages 482–491. Springer, 2008.

Sankaran S. et Kasilingam R. G. : An integrated approach to cell formation and part routing in group technology manufacturing systems. *Engineering Optimization*, 16(3):235–245, 1990.

Sayadi M. K., Hafezalkotob A. et Naini S. G. J. : Firefly-inspired algorithm for discrete optimization problems : an application to manufacturing cell formation. *Journal of Manufacturing Systems*, 32(1):78–84, 2013.

Schaller J. : Tabu search procedures for the cell formation problem with intra-cell transfer costs as a function of cell size. *Computers & Industrial Engineering*, 49(3):449–462, 2005.

Schmidt B., Al-Fuqaha A., Gupta A. et Kountanis D. : Optimizing an artificial immune system algorithm in support of flow-based internet traffic classification. *Applied Soft Computing*, 54:1–22, 2017.

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*, 27 (7):1161–1165, 1989.

Seifoddini H. et Wolfe P. M. : Application of the similarity coefficient method in group technology. *IIE transactions*, 18(3):271–277, 1986.

Sekhar P. et Mohanty S. : An enhanced cuckoo search algorithm based contingency constrained economic load dispatch for security enhancement. *International Journal of Electrical Power & Energy Systems*, 75:303–310, 2016.

Shayan E. et Sobhanallahi A. : Productivity gains by cellular manufacturing. *Production Planning & Control*, 13(6):507–516, 2002.

Sofianopoulou S. : Manufacturing cells design with alternative process plans and/or replicate machines. *International Journal of Production Research*, 37(3):707–720, 1999.

Soleymanpour M., Vrat P. et Shankar R. : A transiently chaotic neural network approach to the design of cellular manufacturing. *International Journal of Production Research*, 40(10):2225–2244, 2002.

Solimanpur M. et Elmi A. : A tabu search approach for cell scheduling problem with makespan criterion. *International Journal of Production Economics*, 141(2):639–645, 2013.

Solimanpur M. et Foroughi A. : A new approach to the cell formation problem with alternative processing routes and operation sequence. *International Journal of Production Research*, 49(19): 5833–5849, 2011.

Solimanpur M., Saeedi S. et Mahdavi I. : Solving cell formation problem in cellular manufacturing using ant-colony-based optimization. *The International Journal of Advanced Manufacturing Technology*, 50(9):1135–1144, 2010.

Solimanpur M., Vrat P. et Shankar R. : Ant colony optimization algorithm to the inter-cell layout problem in cellular manufacturing. *European Journal of Operational Research*, 157(3):592–606, 2004a.

Solimanpur M., Vrat P. et Shankar R. : Feasibility and robustness of transiently chaotic neural networks applied to the cell formation problem. *International Journal of Production Research*, 42 (6):1065–1082, 2004b.

Spiliopoulos K. et Sofianopoulou S. : Designing manufacturing cells : a staged approach and a tabu search algorithm. *International Journal of Production Research*, 41(11):2531–2546, 2003.

Spiliopoulos K. et Sofianopoulou S. : An efficient ant colony optimization system for the manufacturing cells formation problem. *The International Journal of Advanced Manufacturing Technology*, 36(5):589–597, 2008.

Srinivasan G. et Narendran T. T. : Grafics—a nonhierarchical clustering algorithm for group technology. *International Journal of Production Research*, 29(3):463–478, 1991.

Srinlvasan G., Narendran T. T. et Mahadevan B. : An assignment model for the part-families problem in group technology. *International Journal of Production Research*, 28(1):145–152, 1990.

Stanfel L. E. : Machine clustering for economic production. *Engineering costs and production economics*, 9(1-3):73–81, 1985.

Sun D., Lin L. et Batta R. : Cell formation using tabu search. *Computers & Industrial Engineering*, 28(3):485–494, 1995.

Suresh S. et Lal S. : An efficient cuckoo search algorithm based multilevel thresholding for segmentation of satellite images using different objective functions. *Expert Systems with Applications*, 58:184–209, 2016.

Suresh Kumar C. et Chandrasekharan M. P. : Grouping efficacy : a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *International Journal of Production Research*, 28(2):233–243, 1990.

Susanto S., Kennedy R. D. et Price J. W. H. : A new fuzzy-c-means and assignment technique-based cell formation algorithm to perform part-type clusters and machine-type clusters separately. *Production planning & control*, 10(4):375–388, 1999.

Tang H. et Miller-Hooks E. : A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407, 2005.

Tavakkoli-Moghaddam R., Ranjbar-Bourani M., Amin G. R. et Siadat A. : A cell formation problem considering machine utilization and alternative process routes by scatter search. *Journal of Intelligent Manufacturing*, 23(4):1127–1139, 2012.

Torkul O., Cedimoglu I. H. et Geyik A. K. : An application of fuzzy clustering to manufacturing cell design. *Journal of Intelligent & Fuzzy Systems*, 17(2):173–181, 2006.

Tran C. D., Dao T. T., Vo V. S. et Nguyen T. T. : Economic load dispatch with multiple fuel options and valve point effect using cuckoo search algorithm with different distributions. *International Journal of Hybrid Information Technology*, 8(1):305–316, 2015.

Tunnukij T. et Hicks C. : An enhanced grouping genetic algorithm for solving the cell formation problem. *International Journal of Production Research*, 47(7):1989–2007, 2009.

Vakharia A. J. et Chang Y. L. : Cell formation in group technology : a combinatorial search approach. *International Journal of Production Research*, 35(7):2025–2044, 1997.

Valian E., Mohanna S. et Tavakoli S. : Improved cuckoo search algorithm for feedforward neural network training. *International Journal of Artificial Intelligence & Applications*, 2(3):36–43, 2011.

Vansteenwegen P., Souffriau W., Berghe G. V. et Van Oudheusden D. : Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290, 2009.

Venkatadri U., Rardin R. L. et Montreuil B. : A design methodology for fractal layout organization. *IIE transactions*, 29(10):911–924, 1997.

Venkumar P. et Haq A. N. : Manufacturing cell formation using modified art1 networks. *The International Journal of Advanced Manufacturing Technology*, 26(7-8):909–916, 2005.

Venkumar P. et Haq A. N. : Complete and fractional cell formation using kohonen self-organizing map networks in a cellular manufacturing system. *International Journal of Production Research*, 44(20):4257–4271, 2006.

Venugopal V. et Narendran T. T. : Cell formation in manufacturing systems through simulated annealing : an experimental evaluation. *European Journal of Operational Research*, 63(3):409–422, 1992.

Vin E., De Lit P. et Delchambre A. : A multiple-objective grouping genetic algorithm for the cell formation problem with alternative routings. *Journal of Intelligent Manufacturing*, 16(2):189–205, 2005.

Viswanathan G. M., Bartumeus F., Buldyrev S. V., Catalan J., Fulco U. L., Havlin S., Da Luz M. G. E., Lyra M. L., Raposo E. P. et Stanley H. E. : Lévy flight random searches in biological phenomena. *Physica A : Statistical Mechanics and Its Applications*, 314(1):208–213, 2002.

Waghodekar P. H. et Sahu S. : Machine-component cell formation in group technology : Mace. *International Journal of Production Research*, 22(6):937–948, 1984.

Wang J. et Roze C. : Formation of machine cells and part families : a modified p-median model and a comparative study. *International Journal of Production Research*, 35(5):1259–1286, 1997.

Watkins A. et Timmis J. : Exploiting parallelism inherent in airs, an artificial immune classifier. *In ICARIS*, pages 427–438. Springer, 2004.

Wemmerlöv U. et Hyer N. L. : Procedures for the part family/machine group identification problem in cellular manufacturing. *Journal of Operations Management*, 6(2):125–147, 1986.

Won Y. et Currie K. R. : Fuzzy art/rrr-rss : a two-phase neural network algorithm for part-machine grouping in cellular manufacturing. *International Journal of Production Research*, 45(9):2073–2104, 2007.

Won Y. et Kim S. : Multiple criteria clustering algorithm for solving the group technology problem with multiple process routings. *Computers & Industrial Engineering*, 32(1):207–220, 1997.

Won Y. et Lee K. C. : Modified p-median approach for efficient gt cell formation. *Computers & Industrial Engineering*, 46(3):495–510, 2004.

Wu T. H., Chang C. C. et Chung S. H. : A simulated annealing algorithm for manufacturing cell formation problems. *Expert Systems with Applications*, 34(3):1609–1617, 2008.

Wu T. H., Chung S. H. et Chang C. C. : Hybrid simulated annealing algorithm with mutation operator to the cell formation problem with alternative process routings. *Expert Systems with Applications*, 36(2):3652–3661, 2009.

Wu T. H., Chung S. H. et Chang C. C. : A water flow-like algorithm for manufacturing cell formation problems. *European Journal of Operational Research*, 205(2):346–360, 2010.

Wu T. H., Low C. et Wu W. T. : A tabu search approach to the cell formation problem. *The International Journal of Advanced Manufacturing Technology*, 23(11-12):916–924, 2004.

Wu T. H., Yeh J. Y. et Chang C. C. : A hybrid simulated annealing algorithm to the cell formation problem with alternative process plans. *In International Conference on Convergence Information Technology, 2007*, pages 199–204. IEEE, 2007.

Yang X. S. et Deb S. : Cuckoo search via lévy flights. *In World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009.*, pages 210–214. IEEE, 2009.

Yang X. S. et Deb S. : Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4):330–343, 2010.

Yasar M. : Optimization of reservoir operation using cuckoo search algorithm : Example of adiguzel dam, denizli, turkey. *Mathematical Problems in Engineering*, 2016:1–7, 2016.

Yin Y. et Yasuda K. : Manufacturing cells' design in consideration of various production factors. *International Journal of Production Research*, 40(4):885–906, 2002.

Ying K. C., Lin S. W. et Lu C. C. : Cell formation using a simulated annealing algorithm with variable neighbourhood. *European Journal of Industrial Engineering*, 5(1):22–42, 2011.

Yuan W., Yang M., Guo Q., Wang X. et Feng X. : Improved cuckoo search algorithm for spectrum sensing in sparse satellite cognitive systems. *In IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2016.

Zandieh M., Ghomi S. F. et Husseini S. M. M. : An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. *Applied Mathematics and Computation*, 180(1):111–127, 2006.

Zeb A., Khan M., Khan N., Tariq A., Ali L., Azam F. et Jaffery S. H. I. : Hybridization of simulated annealing with genetic algorithm for cell formation problem. *The International Journal of Advanced Manufacturing Technology*, 86(5-8):2243–2254, 2016.

Zettam M. et Elbenani B. : Gravitational search algorithm applied to the cell formation problem. *In Nature-Inspired Computation in Engineering*, pages 251–266. Springer, 2016.

Zhong Y., Zhang L., Huang B. et Li P. : An unsupervised artificial immune classifier for multi/hyperspectral remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 44(2):420–431, 2006.

Zolfaghari S. et Liang M. : An objective-guided ortho-synapse hopfield network approach to machine grouping problems. *International Journal of Production Research*, 35(10):2773–2792, 1997.

Zolfaghari S. et Liang M. : Machine cell/part family formation considering processing times and machine capacities : a simulated annealing approach. *Computers & industrial engineering*, 34 (4):813–823, 1998.

Résumé

Le travail de recherche présenté dans cette thèse porte sur la résolution du problème de formation de cellules, qui est considéré comme l'un des premiers problèmes rencontrés pour réaliser un système de production cellulaire. L'absence d'un algorithme déterministe capable de résoudre le problème de façon optimale et dans un temps de calcul raisonnable justifie souvent le recours aux algorithmes approximatifs. Dans ce contexte, nous proposons des approches basées sur deux métaheuristiques inspirées de la nature: l'algorithme de sélection clonale et l'algorithme de recherche du coucou. Nous commençons par traiter la première variante du problème en considérant sa matrice d'incidence (produits/machines) et en fixant le nombre de cellules de production. Nous visons à maximiser l'efficacité du système en réduisant les mouvements intercellulaires des produits. Puis, nous étudions le cas où le nombre de cellules est variable comme une deuxième variante du problème de formation de cellules. Ensuite, nous abordons la troisième variante du problème qui prend en compte la possibilité d'avoir plusieurs routes de traitement pour chaque produit et d'intégrer différents facteurs de production; ce qui rend le problème plus réaliste mais aussi plus complexe.

Nous avons développé plusieurs approches basées sur l'algorithme de sélection clonale et l'algorithme de recherche du coucou pour résoudre ces trois variantes. Chaque approche est adaptée différemment au problème pour satisfaire les contraintes de la variante étudiée. De plus, un bon réglage des paramètres est appliqué pour obtenir des meilleures solutions. Les performances des approches proposées sont évaluées sur un ensemble d'instances de référence et comparées à d'autres algorithmes collectés de la littérature. Les résultats obtenus indiquent l'efficacité de nos approches en termes de qualité des solutions et du temps de calcul. Il est à noter que nous sommes les premiers à proposer la résolution du problème et ses variantes avec ces deux approches inspirées de la nature.

Mots-clefs: Problème de formation des cellules; Métaheuristiques inspirées de la nature; Recherche du coucou; Sélection clonale; Fiabilité des machines; Routes alternatives; Problème de formation de cellules généralisé.

Abstract

The research work presented in this thesis focuses on solving the cell formation problem, which is considered one of the first problems faced in designing a cellular manufacturing system. The lack of a deterministic algorithm able to solve the problem optimally and in a reasonable computational time often justifies the use of approximation algorithms. In this context, we propose approaches based on two metaheuristics inspired by nature: the clonal selection algorithm and the cuckoo search algorithm. We start with the first variant of the problem by considering its incidence matrix (product / machine) and by setting the number of production cells. We aim to maximize the efficiency of the system by reducing the intercellular movements of products. Then, we study the case where the number of cells is variable as a second variant of the cell formation problem. Afterwards, we address the third variant of the problem that takes into account the possibility of having alternative process routings for each product and to integrate different production factors; which makes the problem more realistic but also more complex. We have developed several approaches based on the clonal selection algorithm and the cuckoo search algorithm to solve these three variants. Each approach is adapted differently to the problem to satisfy the constraints of the corresponding variant. In addition, a parameter setting is applied to obtain better solutions. The performances of the proposed approaches are evaluated on a set of benchmark instances and compared to other algorithms collected from the literature. The obtained results indicate the superiority of our approaches in terms of solution quality and computational time.

Keywords: Cell formation problem; Nature-inspired metaheuristics; Cuckoo search; Clonal selection; Machine reliability; Alternative routes; Generalized cell formation problem.