REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــــعـليـم العـــــــالي و البـــــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــــامعة أبي بكـــر بـلـقـايد– تـلمســـــــــــــــان –

Université Aboubakr Belkaïd– Tlemcen –

Faculté de TECHNOLOGIE

THESE

Présentée pour l'obtention du **grade** de **DOCTORAT**

En : Productique

Par : Mlle. HOUBAD Yamina

Sujet

Ordonnancement de système de production par métaheuristiques

Soutenue publiquement, le 26 /01/2023 , devant le jury composé de :

M. BELKAID Fayçal	MCA	Université de Tlemcen	Président
Mme. TRIQUI SARI Lamia	MCA	Université de Tlemcen	Directrice de thèse
M. HACHEMI Khalid	Professeur	Université Oran 2	Examinateur
M. BENNEKROUF Mohammed	MCA	ESSA - Tlemcen	Examinateur
M. HASSAM Ahmed	MCB	Université de Tlemcen	Invité

Remerciement

La rédaction de ce manuscrit est l'aboutissement de plusieurs années de travail. Ce travail lui-même n'aurait pu être mené sans l'aide de plusieurs personnes auxquelles je souhaite exprimer ma gratitude.

Ce n'est pas par tradition que cette page figure au préambule de cette thèse, mais c'est plutôt un devoir moral et une reconnaissance sincère qui me pousse à la faire.

Je tiens tout d'abord à remercier ma directrice de thèse Madame TRIQUI SARI Lamia, Maître de conférences à l'université de Tlemcen, pour m'avoir soutenu et prodigué de nombreux conseils tout au long de ma thèse. Je lui suis particulièrement reconnaissante de m'avoir laissé une grande liberté scientifique. En plus de ses qualités scientifiques et pédagogiques, j'ai pu apprécier sa grande sympathie qui rend le travail à ses côtés très agréable.

J'aimerai particulièrement adresser mes remerciements les plus vifs et ma reconnaissance à mon co-directeur de thèse Monsieur HASSAM Ahmed Maître de conférence à l'Université de Tlemcen, pour sa disponibilité, conseils, et ses orientations judicieuses et de m'avoir laissé une grande liberté scientifique, ses remarques et ses suggestions durant tout mon travail.

Toute ma gratitude va ensuite aux membres du Jury, qui ont accepté de consacrer une partie de leur temps à l'évaluation de mon travail :

Je tiens à remercier Monsieur BELKAID Fayçal, maître de conférences à l'Université de Tlemcen, pour m'avoir fait l'honneur de présider mon jury.

J'adresse des remerciements chaleureux à Monsieur HACHEMI Khalid, Professeur à l'Université d'Oran 2 pour avoir consacré de son temps pour examiner ce travail.

De même, j'adresse mes remerciements à Monsieur BENNEKROUF Mohammed, Maître de conférences à l'Ecole Supérieure de Management de Tlemcen d'avoir accepté d'examiner ce travail.

Je tiens aussi à remercier tous les membres de Laboratoire de Productique (MELT), pour leurs encouragements. Ce fut un plaisir de vous côtoyer quotidiennement, j'ai particulièrement apprécié l'ambiance chaleureuse qui règne au laboratoire. Je vous souhaite les meilleures des chances dans vos futurs projets. Que chacun trouve ici l'expression de mon amitié.

Mes remerciements les plus vifs s'adressent à ma spéciale Fatima Zohra BOUMEDIENE, Amina BENDJELLOUL, Latéfa GHOMRI, Hakim Nadhir BESSENOUCI et Khalid MEKAMCHA qui m'ont soutenu tout au long de ces longues années, pour leur amitié, et leur présence dans des moments difficiles.

Mes derniers mots seront à ma famille. Il m'est difficile d'exprimer en quelques mots tous mes remerciements, mon amour, et ma gratitude à la personne sans laquelle

je n’aurai jamais pu atteindre la réussite, ma mère. Elle m’a toujours fait confiance, soutenu, et conseillé.

Mes remerciements s’adressent aussi à mon père et mon frère Mekki pour leur aide, leurs conseils, et leurs encouragements qui étaient indispensables.

A tous, je dédie cette thèse qui n’aurait jamais vu le jour sans vous.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

Durant les dernières années, l'intérêt croissant apporté aux métaheuristiques s'est traduit par un nombre sans cesse de travaux scientifiques, vue même de revus scientifiques consacrées à ce sujet.

Les métaheuristiques sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Elles sont apparues dans le but de résoudre au mieux des problèmes d'optimisation. Malgré qu'elles ne fournissent aucune garantie sur l'optimalité de la solution obtenue elles ont montré leur efficacité en permettant de trouver une solution réalisable en un temps raisonnable pour de nombreux problèmes d'optimisation.

L'intérêt croissant de ces techniques est tout à fait justifié par le développement des machines avec des capacités calculatoires énormes, ce qui a permis de concevoir des métaheuristiques de plus en plus complexes qui ont fait preuve d'une certaine efficacité en termes de la qualité des solutions obtenues.

Lorsque l'on veut concevoir une nouvelle méthode de résolution de problème, il faut souvent une source d'inspiration. Celle-ci peut être issue de la modélisation des systèmes complexes naturels. Il s'agit d'imiter et d'adapter les concepts mis en œuvre par le monde des vivants pour la résolution de problèmes.

Les algorithmes génétiques sont parmi les métaheuristiques bio-inspiré les plus connues. Ils constituent une famille d'algorithmes par recherche probabiliste, basés sur l'évolution des espèces. L'évolution d'une espèce est simplement une suite successive d'amélioration afin qu'elle soit la mieux adaptée au milieu dans lequel elle évolue. Cette adaptation est réalisée grâce à la sélection naturelle et aux mécanismes de la reproduction. Les algorithmes génétiques sont conçus par analogie avec ce processus d'évolution biologique et tirent leur puissance des mêmes mécanismes.

En raison de leur simplicité à comprendre et leur facilité d'adaptation, les algorithmes génétiques, et depuis leur apparition, ont été largement utilisés pour la résolution d'un nombre important de problèmes d'optimisation combinatoire.

Notre contribution principale dans ce travail de thèse est la proposition d'une nouvelle métaheuristique. Nous nous sommes inspiré de mécanisme de survie de l'espèce de

singe Babouins chacma. Contrairement aux algorithmes génétiques où : d'un côté on ne distingue pas le genre de parents (père / mère) et d'un autre côté la création d'une nouvelle génération se fait en croisant les parents deux à deux ; dans l'algorithme des Babouins que nous proposons ici, la population est scindée en individus pères et individus mères et la création de la nouvelle génération se fait en choisissant aléatoirement deux enfants parmi tous les enfants obtenus en croisant une mère avec le nombre de pères candidats à la reproduction.

Ce qui nous a motivés pour le développement de cette nouvelle métaheuristique est la publication en 2014 d'un papier présentant une étude intéressante qui porte sur le phénomène d'infanticide animal et la stratégie développé par les femelles chez certaines espèces pour la protection de leurs petits. Sachant que le phénomène d'infanticide chez les animaux a été observé pour la première fois chez les Langurs d'Inde. Cette nouvelle a ébranlé les zoologues a été expliqué par certains en attribuant le phénomène au stress subi par ces singes au contact des hommes. C'est jusqu'à 1977 qu'une étude apporte une nouvelle explication à ce phénomène. En effet, il s'agit d'une stratégie liée à la théorie de l'évolution. Chez certaines espèces, le mâle dominant contrôle la provenance des petits qu'il aura à sa charge en tuant les petits des autres mâles afin de maximiser *ses* chances d'assurer une descendance.

Dans le papier publié en 2014, l'étude est basée sur l'observation détaillée de plus de 260 espèces de mammifères depuis plus d'un demi-siècle pour déterminer les espèces caractérisées par l'infanticide, analyser les causes et décrire les stratégies mises en place par les femelles pour préserver leurs progénitures. Sachant que dans cette étude, le comportement n'est considéré que lorsqu'il est stratégique, à savoir, lorsqu'il offre des avantages aux mâles. Les résultats accablants de l'étude ont montré que l'infanticide est présent chez 119 espèces. On compte ici gorilles, les chimpanzés, les babouins, les lions, les ours, les hippopotames, les souris ainsi que les écureuils. Le plus étonnant est que les mâles tuent les petits des autres, mais aucun meurtre de leurs propres petits n'ayant jamais été constaté chez les mammifères (à l'exception notable de l'homme).

L'étude était plus détaillée et a donné un intérêt remarquable à l'espèce des primates des babouins chacma de l'Okavango (Botswana), en vue qu'elle est très facile à observer et est très visibles en Afrique australe. Les femelles de cette espèce

multiplient les partenaires ce qui fait que les mâles, dans le doute, s'abstiennent de tuer un petit qui pourrait être le leur. En d'autres termes, les femelles s'accouplent avec autant de mâles que possible sur une courte période de temps, cela rend difficile de discerner la paternité des petits. Nous nous sommes inspiré de cette stratégie pour l'élaboration d'une nouvelle métaheuristique nommée l'algorithme des babouins.

Comme contribution secondaire de ce travail de thèse nous avons proposé une amélioration de l'algorithme génétique sous sa forme parallèle en modèle des îles, en introduisant un paramètre de la gestion de migration.

Nous avons validé les performances de l'ensemble des approches développées (à la fois l'algorithme des babouins ainsi que l'algorithme IMGAMM) en utilisant comme application l'ordonnancement des ateliers et plus particulièrement l'ordonnancement dans un atelier flow shop, plus communément connu sous le nom d'atelier à cheminement unique.

La résolution d'un problème d'ordonnancement consiste à programmer ou planifier, dans le temps ou dans l'espace, l'exécution de tâches en leur attribuant les ressources nécessaires matérielles ou humaines pour satisfaire un ou plusieurs critères, tout en respectant les contraintes de réalisation imposées. Les problèmes d'ordonnancement sont, dans le cas général, formulés sous forme de problème d'optimisation combinatoire. Dans certains cas leur résolution fait appel aux méthodes exactes qui garantissent en général l'optimalité de la solution fournie, mais dans la majorité des cas les problèmes d'ordonnancement sont classés NP-Difficiles, et les méthodes exactes se trouvent incapables de les résoudre en un temps polynomial et font appel aux méthodes approchées, notamment les métaheuristiques. La théorie de l'ordonnancement occupe un rôle important dans de nombreux secteurs de l'économie. C'est une branche de la recherche opérationnelle, visant à définir les dates de l'exécution d'un ensemble de travaux en tenant compte de la disponibilité des ressources, de façon à optimiser un ou plusieurs critères, tout en respectant les contraintes d'organisation et de fabrication. La résolution des problèmes d'ordonnancement fait appel à toutes les techniques de l'optimisation combinatoire (programmation mathématique, programmation dynamique, …). Ces méthodes, dites exactes, garantissent, l'optimalité de la solution fournie, mais certains problèmes demeurent hors de portée de ces approches, on parle alors des problèmes dits NP-Difficiles. L'appartenance à la classe

des problèmes de type NP-Difficiles signifie qu'on ne pourra probablement jamais résoudre ce problème d'une manière optimale et qu'il est nécessaire de construire des méthodes de résolution dédiée. Les métaheuristiques offrent alors une alternative intéressante pour la résolution de problèmes avec une telle complexité.

Le manuscrit est organisé de la manière suivante : Dans le premier chapitre, nous présentons les notions de base utiles pour la compréhension de cette thèse. Nous donnons les notions fondamentales concernant les systèmes de production. Nous consacrons par la suite une section dans laquelle nous détaillons quelques notions liées à l'ordonnancement, à savoir la définition d'un problème d'ordonnancement, les objectifs de l'ordonnancement, et les méthodes de résolution de ces problèmes, à savoir, les méthodes exactes et les méthodes approchées.

Dans le deuxième chapitre nous nous intéressons de manière approfondie aux métaheuristiques tout en décrivant certaines méthodes à solution unique et à base de population de solutions, leurs origines, leurs principes de base, et leurs algorithmes. La dernière section du chapitre est consacrée à un état de l'art exhaustif que nous avons réalisé et qui regroupe les travaux les plus importants proposant de nouvelles métaheuristiques afin de nous permettre de nous situer dans ce domaine. En effet le domaine qui nous a inspiré n'a fait l'objet d'aucune autre référence.

Le troisième chapitre est consacré à la première métaheuristique développée, à savoir l'algorithme génétique modèle des îles avec gestion de migration. Dans la première section du chapitre nous détaillons les notions de bases liées à l'algorithme génétique, ensuite nous présentons l'algorithme génétique sous sa forme parallèle en modèle des îles. La troisième section du chapitre présente notre contribution proprement dite au niveau de cet algorithme qui consiste à introduire un paramètre de gestion de migration entre les îles. Une étude de sensibilité a été menée et nous avons appliqué la technique développée pour la résolution d'un problème d'ordonnancement dans un atelier Flow Shop.

Le quatrième chapitre est consacré à la métaheuristique que nous avons développée en s'inspirant du comportement de reproduction et survie chez l'espèce des singes babouins. La première section présente l'espèce des singes babouins, leur comportement social et mis le point sur l'infanticide en étant un phénomène caractérisant pour certaines espèces y compris les babouins et la stratégie de protection

des petits adoptée par les femelles de cette espèce. Nous présentons par la suite l'algorithme développé, son principe et ses paramètres. La dernière section englobe les résultats de simulation d'une étude de sensibilité de l'algorithme ainsi qu'une comparaison des résultats obtenus par la méthodes développée et ceux de l'algorithme génétique classique, l'algorithme génétique modèles des îles, et l'algorithme génétique modèle des îles avec gestion de migration permettant l'évaluation de la performance des algorithmes développés.

Enfin, nous terminons par une conclusion générale qui fera la synthèse de nos différentes contributions, et donne quelques perspectives de recherche.

CHAPITRE 1

Généralités sur l'ordonnancement des systèmes de production

Ce chapitre présente des notions de base utiles pour la compréhension de ce document. La première partie du chapitre est consacrée aux notions fondamentales sur les systèmes de production. La seconde partie est consacrée aux notions liées à l'ordonnancement, à savoir la définition d'un problème d'ordonnancement, les objectifs de l'ordonnancement, leurs complexités et les méthodes de résolution.

I. Introduction

Un système de production rassemble tous les moyens permettant de donner une valeur ajoutée aux produits ou aux services. Les besoins industriels ont entraîné l'apparition et le développement de nouvelles technologies pour mieux gérer ces systèmes de production, d'où l'importance du rôle de la gestion de production. Cette dernière a pour objet la recherche d'une organisation efficace dans l'espace et dans le temps de toutes les activités relatives à la production afin d'atteindre les objectifs de l'entreprise. L'ordonnancement fait partie des fonctions de la gestion de production, dans ce chapitre nous nous intéressons à la fonction ordonnancement et leurs méthodes de résolution.

Les problèmes liés aux systèmes de production, tels que l'ordonnancement des tâches sont souvent formulés en problèmes d'optimisation combinatoires et leur difficulté augmente avec l'augmentation de la taille de problème. On ne connait pas d'algorithme capable de résoudre ces problèmes de manière exacte. Depuis la fin des années 80, les méthodes approchées, et en particulier, les métaheuristiques, présentent une alternative intéressante pour la résolution de tels problèmes, en raison de leur capacité à fournir des solutions de bonne qualité pour une consommation en temps de calcul réduit.

Dans la deuxième section de ce chapitre nous présentons quelques notions liés aux systèmes de production. La troisième section est consacrée à l'ordonnancement des systèmes de production, ses contraintes, ses objectifs, et la complexité de la résolution des problèmes d'ordonnancement. La dernière section présente une vue générale sur les méthodes de résolution des problèmes d'ordonnancement.

II. Les systèmes de production

« Produire c'est transformer » [1], donc le système de production rassemble tous les moyens permettant de donner une valeur ajoutée aux produits ou aux services. Les systèmes de production diffèrent par les objectifs tracés par les décideurs du système. Ces objectifs sont dynamiques et ils évoluent suivant l'évolution du marché et des technologies sur lesquelles s'appuie le système.

Les systèmes de production regroupent l'ensemble des moyens qui permettent de transformer les ressources en produits ou en services.

Les systèmes de production peuvent être classés en trois grandes catégories :

- **Les processus continus :** tels que la production électrique ;
- **Les processus discrets :** tels que l'usinage et toutes les activités d'assemblage ;
- **Les processus discontinus :** qui se situe à mi-chemin entre les processus continus et les processus discrets, les deux types de processus sont couplés : la production et continue mais il y a un conditionnement discret des produits.

III. L'ordonnancement

Les problèmes d'ordonnancement apparaissent dans tous les domaines de l'économie tels que la construction (suivi de projet), l'industrie (problèmes d'ateliers de production), les problèmes d'affectation et de transport, l'administration (emploi du temps et gestion des files d'attente) …

Ordonnancer la fabrication d'un ensemble de produits, c'est en répartir la réalisation dans le temps et l'espace.

Durant des décennies, l'ordonnancement de la production a fait l'objet de nombreuses recherches très approfondies. Il est considéré comme une branche de la recherche opérationnelle et de la gestion de la production, il vise à améliorer l'efficacité des entreprises en termes de coûts de production et des délais de livraison.

« Ordonnancer c'est affecter des ressources dans le temps pour achever un ensemble d'activités. C'est un processus de prise de décision utilisé régulièrement dans de nombreuses industries de production et de services. Il traite de l'allocation des ressources aux tâches sur des périodes de temps données et vise à optimiser un ou plusieurs objectifs » [2].

« Déterminer ce qui va être fait, quand, où et avec quels moyens, étant donné un ensemble de tâches à accomplir, le problème d'ordonnancement consiste à déterminer quelles tâches doivent être exécutées et à assigner des dates et des ressources à ces tâches de façon à ce que les tâches soient, dans la mesure du possible, accomplies en temps utile, au moindre coût et dans les meilleurs conditions » [3].

Les tâches, les ressources, et les contraintes sont les notions fondamentales en ordonnancement.

En d'autres termes, établir un ordonnancement revient à organiser dans le temps la réalisation d'un ensemble de tâches, compte tenu de contraintes temporelles et de contraintes liées à la disponibilité des ressources, afin de satisfaire un ou plusieurs critères d'optimisation.

D'après les définitions précédentes on remarque que dans un problème d'ordonnancement interviennent trois notions fondamentales : les tâches, les ressources, et les contraintes.

III.1. Les tâches

Les tâches sont le dénominateur commun des problèmes d'ordonnancement, leur définition n'est ni toujours immédiate, ni toujours triviale.

« Une tâche est le processus le plus élémentaire. Elle est constituée d'un ensemble d'actions à accomplir, dans des conditions fixées, pour obtenir un résultat attendu et identifié, en termes de performances, de coûts et de délais».

Une tâche est un travail élémentaire localisé dans le temps, et dont la réalisation nécessite un certain intervalle de temps appelé durée.

En effet, elles représentent toutes les opérations à effectuer pour la fabrication des produits. Une tâche, c'est à dire un ensemble d'opérations, requiert pour son exécution certaines ressources qu'il faut programmer de façon à optimiser un certain objectif.

Si l'ensemble des tâches à exécuter au cours du temps est donné a priori, c'est-à-dire leurs dates (date de début et date de fin) et leurs ordres d'exécution sont connus à l'avance, le problème d'ordonnancement est dit statique. Dans le cas contraire, si l'ensemble des tâches à exécuter évolue dans le temps, le problème est dit dynamique.

On distingue deux types de tâches :

- *les tâches préemptives* : elles peuvent être exécutées en plusieurs fois facilitant ainsi la résolution de certains problèmes ;
- *les tâches non morcelables* (indivisibles) qui sont exécutées en une seule fois et ne peuvent pas être interrompues avant qu'elles soient complètement terminées.

III.2. Les Ressources

Les ressources dans une organisation peuvent prendre de nombreuses formes. Elles peuvent être des ressources des machines dans un atelier, des équipes sur un chantier de construction, des pistes dans un aéroport, des unités de traitement dans un environnement informatique, etc [2].

Une ressource peut être définie comme étant un moyen (technique ou humain) nécessaire et indispensable utilisé dans la réalisation des tâches. Dans un système de production, plusieurs types de ressources sont distingués :

Une ressource est un moyen, technique ou humain, dont la disponibilité limitée ou non est connue a priori.

- *Les ressources renouvelables :* ce sont des ressources qui, après avoir été allouées à une tâche, redeviennent disponibles et qui peuvent être réutilisées (machines, personnel, etc.) ;
- *Les ressources consommables* : les ressources qui, après avoir été allouées à une tâche, ne sont plus disponibles, et sont donc épuisées (argent, matières premières, etc.) ;
- L*es ressources partageables* qui peuvent être partagées entre plusieurs tâches.

Une autre manière de classer les ressources est la suivante :

- L*es ressources disjonctif,* qui ne peuvent exécuter qu'une opération ou une tâche à la fois ;
- *Les ressources cumulatif,* qui peuvent exécuter plusieurs opérations simultanément.

Une machine est considérée comme un type de ressource qui n'est caractérisé que par son horaire de travail.

III.3. Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement les variables représentant les relations reliant les tâches et les ressources.

Une contrainte exprime des restrictions ou exigences sur les valeurs que peuvent prendre conjointement les variables représentant les relations reliant les tâches et les ressources. Ce sont les exigences qu'il faut respecter lors de la composition de l'ordonnancement pour qu'il soit réalisable. Elles rendent les problèmes d'ordonnancement plus difficiles, car il faut les respecter lors de la résolution de ces problèmes. Dans un problème ordonnancement d'atelier, on distingue quatre types de contraintes : contraintes de ressources, contraintes temporelles, contraintes d'enchaînement et contraintes technologiques.

- **Contraintes de ressources :** liées à la limitation de la quantité de ressources. En d'autres termes, ne pas dépasser un certain nombre de tâches à exécuter sur la ressource considérée. Elles peuvent être:
  - *Des contraintes disjonctives* : une seule ressource est partagée par deux ou plusieurs tâches ;
  - *Des contraintes cumulatives* : le nombre de tâches à exécuter en parallèle doit être inférieur à la capacité des ressources.
- **Contraintes temporelles :** elles constituent les valeurs que peuvent prendre certaines variables temporelles. On distingue :
  - *Des contraintes de dates butoirs* : certaines tâches doivent être achevées avant une date préalablement fixée ;
  - *Des contraintes de précédence* : une tâche i doit précéder la tâche j ;
  - *Des contraintes de dates au plus tôt* : liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.
- **Contraintes d'enchaînement :** Nous qualifions de contrainte d'enchaînement (dites aussi de succession) une contrainte qui lie le début ou la fin de deux activités par une relation linéaire. Ce sont des contraintes imposées, généralement, par la cohérence technologique (les gammes opératoires dans le cas d'ateliers) qui décrivent des positionnements relatifs qui doivent être respectés entre les tâches.

En outre, on peut distinguer les contraintes suivant qu'elles soient strictes ou pas. Les contraintes dites strictes sont des exigences à respecter alors que les contraintes dites de «préférences» peuvent éventuellement n'être pas satisfaites.

IV. Objectifs de l'ordonnancement

L'environnement manufacturier évolue rapidement et la concurrence devient de plus en plus acharnée et les entreprises doivent se fixer des objectifs diversifiés pour demeurer compétitif.

L'ordonnancement vise à coordonner les ressources de production et à synchroniser les opérations, de manière à satisfaire la demande en produits, et ce, au plus bas coût possible pour l'entreprise.

Les objectifs à satisfaire au niveau de l'ordonnancement sont issus des objectifs globaux de l'entreprise par décomposition conduisant à une structure d'objectifs qui permet de gérer les contradictions et les compromis [4]. La figure 1.1 représente un exemple de structure d'objectifs.

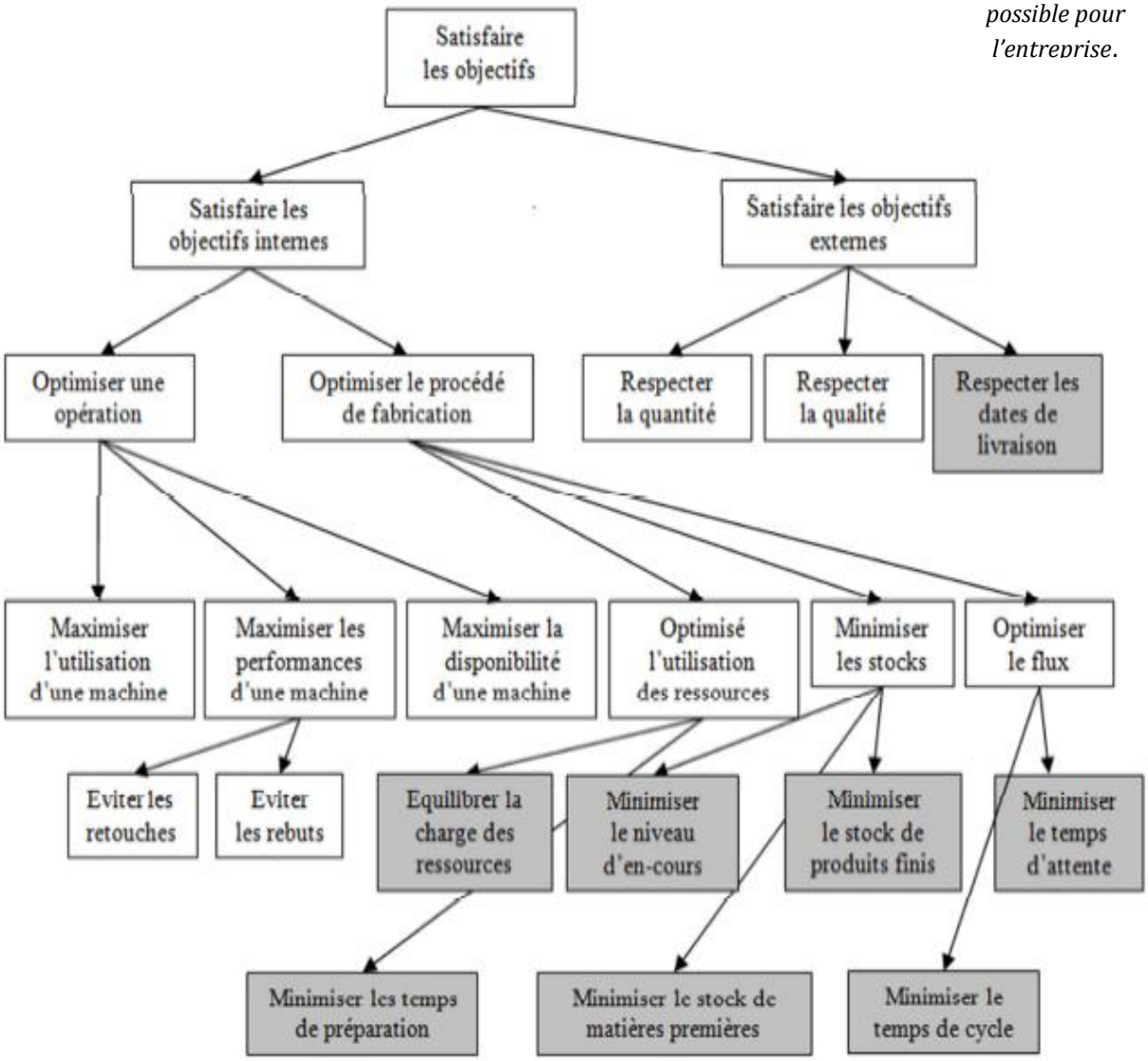


Figure 1.1. Exemple de structure d'objectifs [5]

Dans l'exemple illustré de la figure 1.1., les objectifs de haut niveau (stratégiques) sont décomposés en :

- **Objectifs externes :** qui quantifient la relation entre l'entreprise et son environnement.
- **Objectifs internes :** qui évaluent les performances des ressources du système.

On ne conserve ici que les objectifs concernant la fonction d'ordonnancement (les blocs en gris sur la Figure 1.1).

On distingue les objectifs externes reliant l'entreprise à son environnement et les objectifs internes qui génèrent un plan de travail détaillé pour chaque ressource impliquée dans la production.

On distingue plusieurs objectifs concernant un ordonnancement, nous citons ici les plus répondus :

- **Les objectifs liés au temps :** On trouve par exemple :

  - La minimisation du temps total d'exécution,
  - La minimisation du temps moyen d'achèvement,
  - La minimisation des durées totales de réglage ou des retards par rapport aux dates de livraison.

- **Les objectifs liés aux ressources :**

  - Maximiser la charge d'une ressource,
  - Minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches.

- **Les objectifs liés au coût :**

  - Minimiser les coûts de lancement,
  - Minimiser les coûts de production,
  - Minimiser les coûts de stockage,
  - Minimiser les coûts de transport.

V. Les problèmes d'ordonnancement

Les problèmes d'ordonnancement se rencontrent très souvent dans le milieu industriel, notamment dans l'optimisation de la gestion des systèmes de production.

Les problèmes d'ordonnancement sont présents dans tous les secteurs d'activités de l'économie depuis l'informatique jusqu'à l'industrie manufacturière et leur résolution continue à faire l'objet de nombreux travaux de recherche.

Résoudre un problème d'ordonnancement consiste à ordonnancer i.e. programmer dans le temps l'exécution des tâches en leurs attribuant les ressources matérielles ou humaines nécessaires de manière à satisfaire un ou plusieurs critères préalablement définis, tout en respectant les contraintes de réalisation [6].

Les problèmes d'ordonnancement d'ateliers présentent des difficultés importantes pour les entreprises. En effet, c'est à ce niveau que doivent être prises en compte les différentes caractéristiques des ateliers, ainsi que les perturbations internes telles que les pannes de machines et absence d'opérateurs ainsi que celles externes telles que la variation de la demande.

Les problèmes d'ordonnancement d'ateliers de production consistent à affecter, sur les machines, un ensemble de tâches (opérations ou travaux)pour que chaque opération doit être réalisée sur une machine de l'atelier de production.

Dans les problèmes d'ordonnancement d'ateliers, les ressources sont généralement les machines et souvent elles ne peuvent réaliser qu'une tâche ou opération à la fois (ressources disjonctives) et chaque job à ordonnancer concerne un produit ou un lot de produits à fabriquer en respectant une gamme de fabrication. Cette gamme précise un ordre strict entre les opérations qui la composent, on parlera alors de contraintes de précédence.

On décrit un problème d'ordonnancement souvent par un triplet α|β|γ, où :

α : décrit l'environnement manufacturier (les machines) et contient une seule entrée.

β : fournit des détails sur les caractéristiques du procédé et des contraintes (il peut ne pas contenir d'entrées comme il peut contenir une seule ou plusieurs entrées).

γ : contient l'objectif à optimiser et contient une seule entrée.

Nous citons dans cette section les configurations possibles de cellule spécifiées dans le champ « α ».

On peut distinguer les systèmes de production selon le nombre de machines dans le système ou la disposition des machines (qu'elles soient unique ou parallèle, identiques ou non, …, on parle alors de types d'ateliers.

- **Une seule machine :** C'est le cas plus simple et est particulier de tous les autres environnements plus complexes, la minimisation de la durée totale des opérations « γ » est entre autres triviale car la séquence des ordres de fabrication est sans importance.
- **Machines parallèles identiques :** dans ce cas on dispose de *m* machines identiques en parallèle ; les ordres de fabrication requièrent une unique opération et peuvent être traités en théorie par n'importe quelle machine. Celles-ci peuvent cependant avoir des vitesses différentes (à l'opposé des vitesses uniformes).

- **Atelier Flow shop :** dans ce type d'atelier est dit aussi à cheminement unique, chaque ordre de fabrication doit être traité par chacune des *m* machines disposées en série et ce, dans le même ordre. Il peut être défini comme un système de traitement dans lequel la séquence de tâches de chaque job est entièrement spécifiée, et tous les jobs visitent les machines dans le même ordre. La figure 1.2. représente un exemple illustratif d'un atelier Flow Shop.

Le flow shop est une production linéaire, caractérisée par une séquence d'opérations identiques pour tous les produits. Chaque produit passe successivement sur toutes les machines.

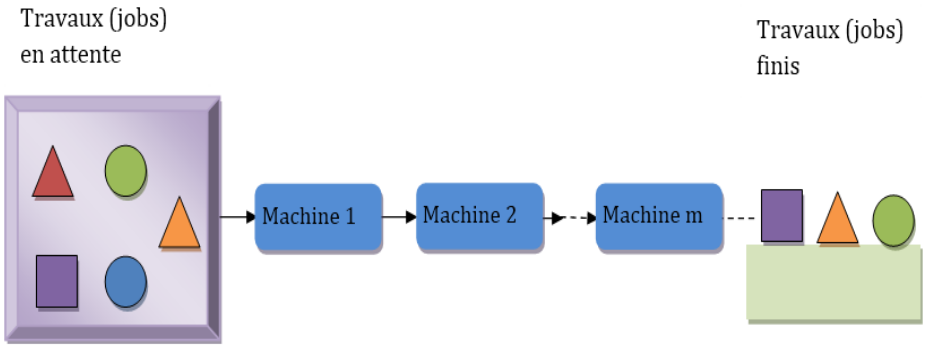


Figure 1.2. Exemple illustratif d'un atelier Flow Shop

- **Atelier Flow shop hybride :** il peut être vu comme une extension du Flow Shop, comme il peut aussi être vu comme une généralisation du Flow Shop et des machines parallèles au sens qu'au lieu de *m* machines en série, il y a *k* étages ou stations en série contenant des machines en parallèle**.** Chaque ordre de fabrication traverse les étages dans le même ordre, en utilisant une seule machine par étage. La figure 1.3. illustre un exemple d'un atelier Flow Shop hybride.

C'est un atelier Flow shop dans lequel chaque machine est remplacée par un étage composé de plusieurs machines qui ne sont pas forcément identiques et disposées en parallèle.

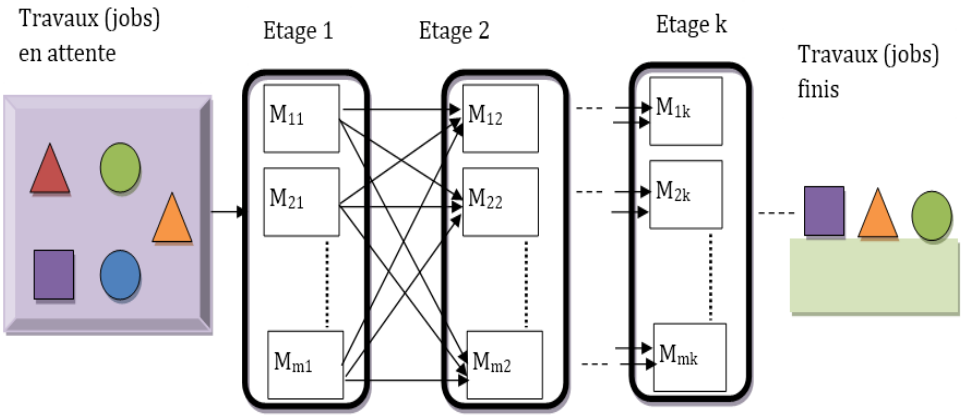
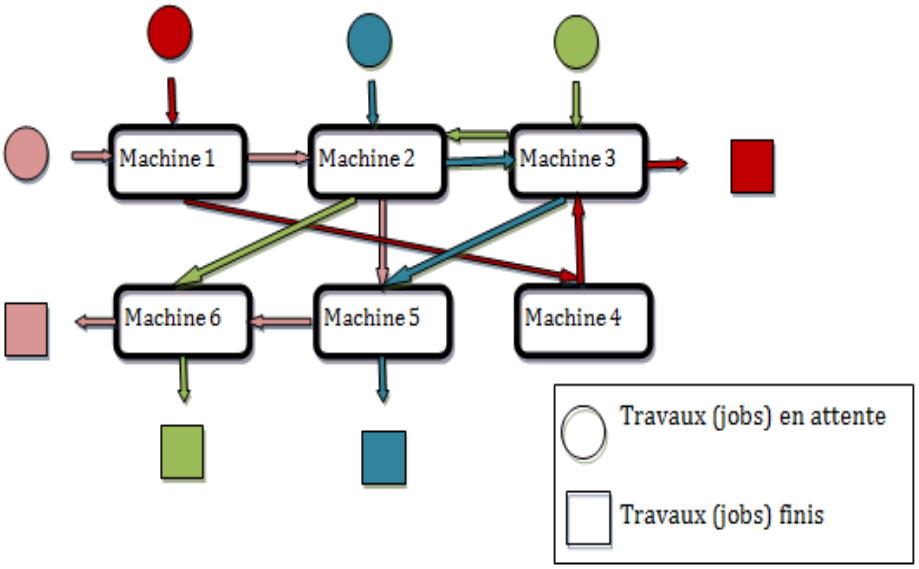


Figure 1.3. Exemple illustratif d'un atelier Flow Shop hybride

- **Atelier Job Shop :** dit aussi à cheminement multiple. Dans un atelier Job Shop chaque job suit une gamme qui lui est propre. La figure 1.4. décrit un exemple de ce type d'atelier.



Dans un problème de type Job shop les produits ne passent pas systématiquement sur toutes les machines selon un ordre commun : dans l'atelier, chaque produit emprunte un routage qui lui est propre.

Figure 1.4. Exemple illustratif d'un atelier Job Shop

Dans ce cas d'ateliers, on distingue aussi le job-shop flexible qui est une extension du modèle job-shop classique; sa particularité réside dans le fait que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations.

- **Atelier Open Shop :** Ce type d'ateliers est moins utilisé dans les entreprises comparé aux autres modèles d'ateliers, flow-shop ou job-shop. Le cheminement de toutes les opérations dans cet atelier est multiple et libre ; dans ce cas l'ordre des opérations n'est pas défini à l'avance et doit être déterminé lors de l'ordonnancement.

Un problème de type Openshop est un Jobshop dans lequel les contraintes de précédence sont relâchées. C'est-à dire, les opérations peuvent être effectuées dans n'importe quel ordre.

VI. Complexité des problèmes d'ordonnancement

Parmi les notions importantes dans la théorie d'ordonnancement, la complexité d'un problème. Cette dernière mesure la difficulté d'un problème. La complexité d'un problème est généralement liée à la complexité et l'efficacité de l'algorithme qui le résout. En d'autres termes, La complexité d'un problème est la complexité de son meilleur algorithme connu.

Déterminer la complexité d'un algorithme donné n'est pas toujours si simple. Il s'agit de déterminer quelle est la plus faible complexité d'un algorithme de résolution, parmi tous les algorithmes que nous pouvons imaginer.

La complexité d'un algorithme dépend de nombreux critères. Par exemple dans un problème d'ordonnancement, la complexité est représentée en fonction de la longueur des données d'entrée (le nombre de jobs, le nombre de ressources et le nombre d'opérations).

Les problèmes d'ordonnancement sont souvent formulés sous forme problèmes d'optimisation combinatoire et sont, dans le cas général, classés NP-Difficiles.

Les problèmes d'ordonnancement d'ateliers sont, généralement, des problèmes combinatoires difficiles, il n'existe pas de méthodes universelles permettant de résoudre tous les cas. On peut ainsi classer les problèmes suivant leur complexité. Cependant, on distingue deux familles de problèmes : les problèmes qui ont une complexité polynomiale, et les autres, ceux dont la complexité est exponentielle (problèmes NP-complets ou NP-difficiles)

- **Les problèmes de classe P (polynomial)** : autrement dits problèmes polynômiaux, leurs résolution nécessite des algorithmes qui les résolvent en un temps polynomial des tailles des problèmes.
- **La classe des problèmes NP (NP : Non deterministic Polynomial) :** dans cette classe les problèmes pouvant être résolus par un algorithme polynomial non déterministe. On distingue deux sous classe de cette classe :

L'appartenance à la classe des problèmes NP-difficiles signifie qu'on ne pourra probablement jamais résoudre les problèmes de grande taille de façon optimale.

- *La classe NP-complet*: s'il existe un algorithme polynomial pour la résolution d'un problème NP-complet, alors, on trouve pour tout le reste des problèmes de la classe, des algorithmes polynomiaux pour les résoudre.
- La classe NP-difficile : ces problèmes ne peuvent être résolus en un temps polynomial, mais on peut construire des algorithmes appelés algorithmes pseudo- polynomiaux ou des méthodes approchées; au cours de leur exécution, ces algorithmes font des choix dont l'optimalité n'est pas démontrable.

Les méthodes de résolution des problèmes d'ordonnancement représentent une grande partie des problèmes d'optimisation combinatoire. En effet pour de tels problèmes, les méthodes exactes requièrent un effort calculatoire qui croit exponentiellement avec la taille des instances du problème (explosion combinatoire) et, rapidement, les heuristiques ou métaheuristiques deviennent l'unique moyen d'obtenir une bonne solution en un temps raisonnable.

VII. Méthodes de résolution

La littérature cite plusieurs approches de résolution des problèmes d'ordonnancement qui font appel aux techniques de l'optimisation combinatoire. Elles sont classifiées en deux catégories: les méthodes exactes qui tentent de trouver des solutions optimales aux problèmes et les méthodes approchées qui sont utilisées comme une alternative aux méthodes exactes pour trouver de bonnes solutions ou des solutions acceptables en un temps raisonnable.

Deux grandes familles sont distinguées pour la résolution des problèmes d'ordonnancement : les méthodes exactes et les méthodes approchées.

VII.1. Les méthodes exactes

Ces techniques sont basées soit sur une résolution algorithmique ou analytique, soit par une énumération exhaustive de toutes les solutions possibles. Elles s'appliquent donc aux problèmes qui peuvent être résolus de façon optimale (les problèmes pour lesquels l'optimalité est garantie). L'optimalité de la solution est l'avantage majeur de ces techniques. Ces méthodes sont génériques et demandent une particularité vis-à-vis d'un problème spécifique. Parmi les méthodes exactes, on peut citer :

Ces méthodes ont l'avantage d'être capable de garantir l'optimalité de la solution obtenue, mais les problèmes classés NP-Difficiles demeurent hors de portée de ces méthodes.

VII.1.1. La programmation dynamique

La programmation dynamique est une méthode proposée par Bellman [7] dans le but d'une recherche du plus court chemin dans un graphique. Elle est utilisée pour la résolution des problèmes d'optimisation dont la fonction objectif possède la propriété de décomposabilité [8]. Son principe est basé essentiellement sur la décomposition du problème en un ensemble de sous problèmes liés entre eux par une relation de récurrence.

En effet, les informations obtenues lors de la résolution des sous-problèmes sont utilisées pour résoudre le problème de manière optimale. En conséquence, pour obtenir la solution optimale du problème, il suffit de revenir en arrière dans l'ensemble des décisions prises et stockées de la résolution de chaque sous-problème, ce qui peut coûter cher en temps et en mémoire.

VII.1.2. La programmation linéaire

La programmation linéaire est une approche générique basée sur la modélisation mathématique de problèmes d'optimisation combinatoire, où les contraintes, exprimées en inégalités et la fonction objectif en équation, sont linéaires par rapport aux variables de décision. De nombreux problèmes pratiques dans la recherche opérationnelle peuvent être exprimés sous forme de problèmes de programmation linéaire.

Le principal avantage des méthodes exactes c'est qu'elles garantissent l'optimalité de la solution obtenues. Leur inconvénient majeur est qu'elles demeurent incapables de résoudre les problèmes avec une grande difficulté.

VII.1.3. La Procédure de séparation et évaluation (Branch an Bound)

Cette méthode consiste à appliquer une technique de séparation et évaluation. La séparation permet de décomposer le problème en un ensemble de sous-problèmes de taille réduite. Les sous problèmes résultants sont ensuite évalués d'une relaxation (continue ou la grangienne principalement) jusqu'à ne plus avoir que des problèmes faciles à résoudre ou dont on sait avec certitude qu'ils ne peuvent pas contenir de solution optimale.

L'avantage des méthodes exactes c'est qu'elles fournissent une solution optimale au problème traité, mais en contrepartie elles sont généralement couteuses en terme du temps de calcul pour les problèmes de types NP-Difficiles. Au contraire, les méthodes approchées visent à générer des solutions de haute qualité en un temps de calcul raisonnable, mais il n'existe aucune garantie de trouver la solution optimale. Les heuristiques et les métaheuristiques font partie des méthodes approchées, qui présentent une alternative intéressantes pour la résolution des problèmes NP-Difficiles.

VIII. Les méthodes approchées

Depuis la fin des années 80, les méthodes approchées suscitent un intérêt croissant en raison de leur capacité de fournir des solutions d'excellente qualité pour une consommation en temps de calcul réduite. Une méthode approchée ou heuristique est un algorithme d'optimisation qui cherche à trouver une solution réalisable de la fonction objectif, mais sans garantir d'optimalité. L'avantage principal de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles, bien ou mal formulés, avec ou sans contraintes et de plus elles ne nécessitent pas une modélisation mathématique du problème.

Une méthode approchée (dite aussi heuristique) tente de trouver une solution dite approchée (ou sous optimale) en un temps de calcul raisonnable

Parmi les heuristiques, certaines sont adaptables à un grand nombre de problèmes différents sans changements majeurs dans l'algorithme, on parle alors de métaheuristiques.

- **Présentation générale des métaheuristiques**

Les métaheuristiques sont des méthodes d'optimisation stochastique proposées dans les années 80 pour la résolution des problèmes difficiles. Contrairement aux méthodes exactes, il n'est jamais garanti que l'utilisation des métaheuristiques aboutisse à la découverte d'une solution optimale. Le terme métaheuristique provient de la concaténation des mots méta (« au-delà » en grec) et heuristique ("trouver" en grec).

Lorsqu'une heuristique est adaptable à un grand nombre de problèmes, elle est alors dite métaheuristique.

L'interprétation de ces mots peut signifier que l'on effectue des recherches à un très haut niveau. Les métaheuristiques présentent un avantage majeur qui est leur possibilité de s'adapter à une grande classe de problèmes. Leur but est de minimiser (ou de maximiser) une (ou plusieurs) fonction(s) objectif(s). Les métaheuristiques travaillent de manière itérative. Ceci peut permettre l'arrêt de l'algorithme quand on le souhaite, et de récupérer la meilleure solution trouvée. En d'autres termes, l'utilisateur n'est pas obligé d'attendre la fin de l'exécution pour obtenir un résultat. Cependant, plus l'algorithme travaille longtemps, plus il y a de chances que la solution s'améliore.

Un point important des métaheuristiques est qu'elles font évoluer des solutions en les améliorant à chaque itération. En effet, il est toujours nécessaire de pouvoir comparer les solutions afin de retenir la meilleure.

Les métaheuristiques consistent à explorer intelligemment l'espace de recherche pour trouver la solution optimale ou une solution la plus proche possible.

Généralement, dans la construction d'une métaheuristique, une part d'aléatoire est présente afin d'empêcher l'algorithme de tomber dans un minimum local. La part d'aléatoire est différente selon les algorithmes. Il existe des métaheuristiques à méthode implicite, où la part d'aléatoire ne suit pas une loi donnée tels que les algorithmes génétiques par exemple, et des métaheuristiques à méthode explicite, où une distribution de probabilité suit une loi définie à chaque itération comme l'algorithme des colonies de fourmis. En général, la (ou les) solution(s) initiale(s) est choisie au hasard et évolue ensuite pour s'améliorer.

IX. Conclusion

Dans ce chapitre, nous avons présenté quelques notions sur les systèmes de production, l'ordonnancement de la production, et les méthodes dédiées à la résolution de ces types de problèmes.

Les problèmes d'ordonnancement d'ateliers sont des problèmes combinatoires classés extrêmement difficiles et il n'existe pas de méthodes universelles permettant de résoudre efficacement tous les cas. Les méthodes approchées, et plus particulièrement, les métaheuristiques offrent la possibilité de trouver une solution approchée, dite aussi sous optimale, en un temps raisonnable. L'optimalité de la solution trouvée n'étant pas garantie, on cherche à trouver une solution approchée mais de bonne qualité ou bien proche de la solution optimale tant que possible. Depuis l'apparition des premières métaheuristiques dans les années 80, plusieurs variantes ont été proposées et de nouvelles techniques ont été inspirées et développées de la biologie, de l'éthologie, …

CHAPITRE 2

Les métaheuristiques

Ce chapitre est consacré aux métaheuristiques, des techniques de résolution formant une famille d'algorithmes d'optimisation combinatoire, visant à résoudre les problèmes classés avec une énorme difficulté et pour lesquels on ne connait pas de méthodes classiques plus efficaces. Nous commençons par une présentation générale, suivie d'une section consacrée aux principes des métaheuristiques les plus répondues. Enfin nous présentons un état de l'art dans lequel nous citons diverses métaheuristiques qui ont été développées et présentées durant les dernières décennies.

I. Introduction

En présence d'un problème d'optimisation, la principale difficulté à laquelle est confronté un décideur est celle du choix d'une méthode efficace capable de produire une solution optimale en un temps de calcul raisonnable. La résolution des problèmes NP-Difficiles par les approches traditionnelles consiste à appliquer les méthodes exactes. Cependant, ces techniques ont montré leurs limites et certains problèmes demeurent hors de portée de ces approches. Même des formulations simplifiées peuvent s'avérer très difficiles à traiter. Pour trouver un optimum, le seul moyen est souvent de faire une recherche exhaustive sur l'ensemble des solutions réalisables. On distingue alors deux classes de méthodes : les méthodes exactes assurant la résolution des problèmes en un temps polynômial et les méthodes approchées permettant de trouver une solution proche de l'optimale en un temps tolérable. Les de méthodes approchées dites aussi d'approximation constituent une alternative intéressante pour traiter les problèmes d'optimisation de grande taille. Elles sont définies comme étant des procédures exploitant au mieux la structure du problème considéré afin de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible. Lorsque ces méthodes sont conçues de manière simple, rapide et ciblée sur un problème particulier, on les appelle ***heuristiques***. De ce fait, les heuristiques sont beaucoup plus adaptées au contexte industriel, où il ne s'agit pas tant de satisfaire complètement tel ou tel objectif que de fournir une solution réaliste. Lorsqu'une heuristique peut s'adapter à un grand nombre de problèmes elle est alors dite métaheuristique.

II Les métaheuristiques

Les métaheuristiques sont apparues au début des années 1980 avec une ambition commune : résoudre au mieux les problèmes dits d'optimisation difficile. Depuis leur apparition, elles constituent une partie importante des méthodes approchées et ouvrent des voies très intéressantes en matière de conception de méthodes heuristiques pour l'optimisation combinatoire. On appelle métaheuristiques (du grec, méta= au delà et heuristique du mot heuriskin = rechercher) des méthodes conçues pour échapper aux minima locaux. Le terme méta s'explique aussi par le fait que ces méthodes sont des structures générales dont il faut instancier les composants en fonction du problème traité par exemple tel que le voisinage, les solutions de départ ou les critères d'arrêt.

Les heuristiques qui peuvent être adaptées à des problèmes différents sans changements majeurs dans l'algorithme de base sont dites métaheuristiques.

En effet, les métaheuristiques s'appliquent à toutes sortes de problèmes discrets, et elles peuvent s'adapter aussi aux problèmes continus. L'un des intérêts majeurs des de ces techniques est leur facilité d'utilisation dans des problèmes concrets où l'utilisateur cherche généralement des méthodes efficaces permettant d'atteindre un optimum avec une précision acceptable dans un temps raisonnable. Faciliter le choix d'une méthode et simplifier son adaptation à un problème donné est l'un des enjeux dans la conception d'une métaheuristique.

La métaheuristique doit trouver les solutions de bonne qualité dans chaque zone de l'espace de recherche mais en s'échappant des minima locaux.

Les métaheuristiques les plus classiques sont celles fondées sur l'exploration d'un voisinage, et forment la première famille dites à solution unique ou méthodes de recherche locale. Ces méthodes sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage. Ces méthodes sont souvent les premières testées sur les nouveaux problèmes combinatoires émergeant des applications réelles et académiques. On trouve dans la littérature de nombreuses méthodes locales.

Les plus anciennes et les plus connues sont : la méthode de descente, le recuit simulé (Simulated Annealing, SA), la recherche tabou (Tabu Search, TS)…etc.

On trouve les métaheuristiques à solution unique, qui travaillent sur une seule solution et celles à population de solutions qui travaillent sur un ensemble de solutions.

L'autre famille est dite à population de solutions et comme leur nom l'indique ces approches travaillent sur un ensemble de solutions. A chaque itération, la métaheuristique fait évoluer un ensemble de solutions à la fois. Les algorithmes génétiques (Genetic Algorithms, GA), de recherche dispersée (Scatter Search, SS) ou les algorithmes de colonies de fourmis (Ant Colony, AC) sont des exemples courants dans cette famille.

III. Métaheuristiques à solution unique

Les métaheuristiques à solution unique reposent sur le principe de recherche de voisinage. Elles commencent par une solution initiale x0 (générée aléatoirement ou par une précédente méthode d'optimisation) considérée comme solution de départ, la recherche consiste à passer d'une solution à une solution voisine par déplacements successifs. L'ensemble des solutions que l'on peut atteindre à partir d'une solution x est appelé voisinage N(x) de cette solution. Déterminer une solution voisine de x dépend bien entendu du problème traité. Le processus s'arrête lorsqu'on ne peut plus améliorer la solution courante ou parce que le nombre maximal d'itérations (fixé au départ) est atteint.

Les métaheuristiques à solutions uniques sont celles qui partent d'une solution unique et tentent de l'améliorer progressivement.

Dans cette section, nous citons brièvement les méthodes les plus connues dans cette famille.

III.1. Le recuit simulé

Le recuit simulé est un algorithme inspiré de la métallurgie pour l'amélioration de la qualité du métal.

La méthode du recuit simulé a été introduite en 1983 par Kirkpatrick et al. [9] Cette méthode originale est basée sur les travaux bien antérieurs de Metropolis et al. [10] en 1953. Cette méthode peut être considérée comme la première métaheuristique 'grand public' et a reçu l'attention de nombreux travaux et principalement de nombreuses applications.

La méthode du recuit simulé s'inspire du recuit des métaux en métallurgie. Un métal refroidi trop vite présente de nombreux défauts microscopiques, qui par analogie c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire. Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalente à un optimum global.

La méthode du recuit simulé, appliquée aux problèmes d'optimisation, considère une solution initiale et recherche dans son voisinage une autre solution de façon aléatoire. L'originalité de cette méthode est qu'il est possible de se diriger vers une solution voisine de moins bonne qualité avec une probabilité non nulle. Ceci permet d'échapper aux minima locaux.

Le principal avantage du recuit simulé c'est que son mécanisme permet de s'échapper aux minima locaux.

L'évolution de l'algorithme est gérée par un paramètre appelé énergie, l'énergie désigne la valeur de la fonction objectif, dite aussi fonction coût de la solution courante. La température T est un paramètre de l'algorithme : elle détermine la probabilité avec laquelle une dégradation de la solution courante est acceptée. La température décroit graduellement au cours du temps.

Au début de la recherche, la température élevée permet au recuit d'explorer l'espace de recherche sans être piégé par les optima locaux. A mesure que la température décroît, le recuit est de moins au moins capable d'explorer l'espace de recherche, et la recherche se concentre sur une zone déterminée. Par conséquent la vitesse de décroissance de la température détermine la vitesse à laquelle le recuit « converge » vers une solution.

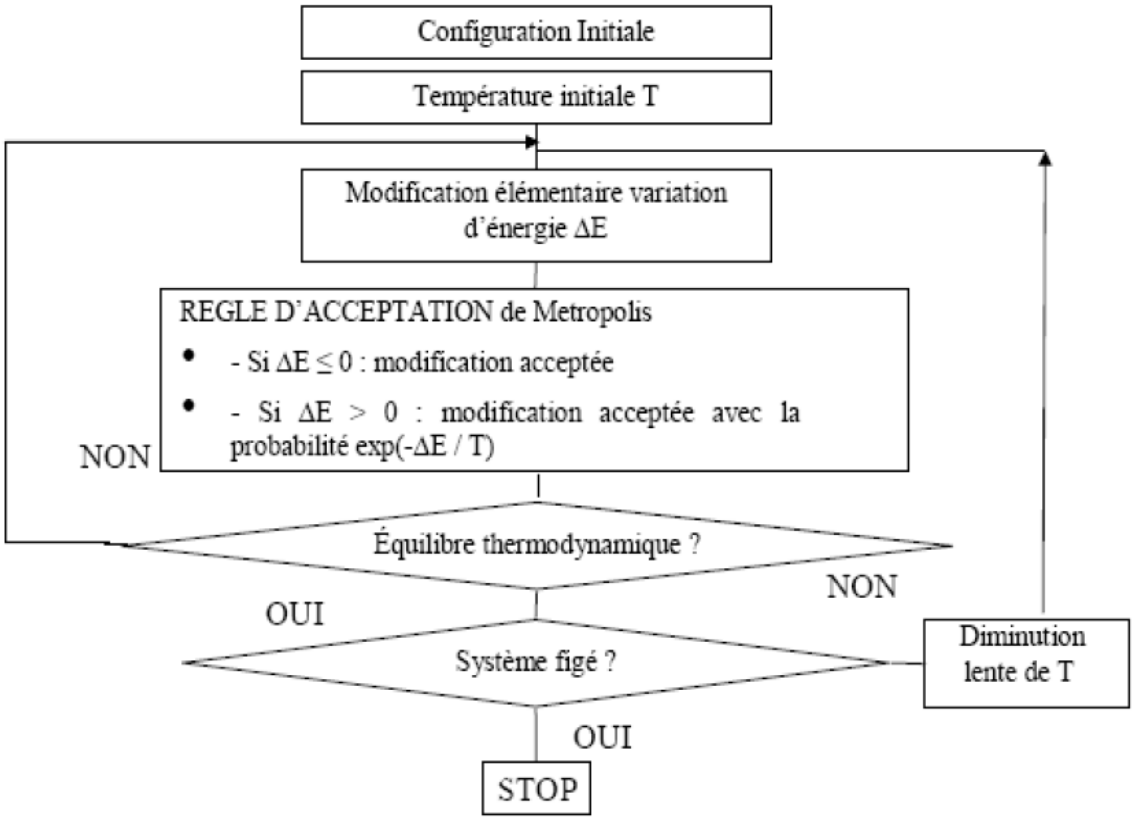


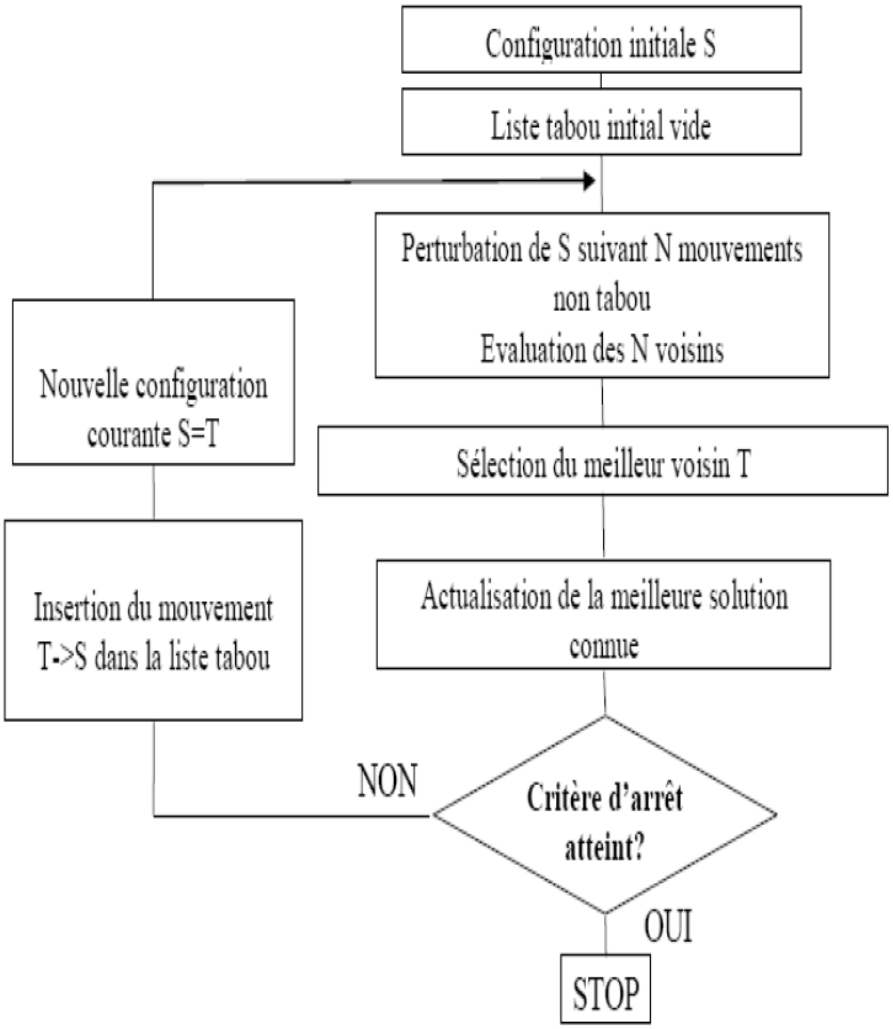
Figure 2.1: Organigramme du recuit simulé

III.2. La recherche taboue

Dans un article présenté par Glover [11] en 1986, on voit apparaître pour la première fois, le terme « recherche taboue » (tabu search). Contrairement au recuit simulé qui ne génère qu'une seule solution x_0 'aléatoirement' dans le voisinage N(x) de la solution courante *x,* la recherche tabou, dans sa forme la plus simple, examine le voisinage N(x) de la solution courante *x*. Cette méthode utilise la notion de mémoire pour éviter de tomber dans un optimum local.

La recherche tabou tente de trouver la meilleure solution dans un voisinage, en évitant d'examiner les mêmes solutions plusieurs fois.

Le principe de l'algorithme est qu'à chaque itération, le voisinage de la solution courante est examiné et la meilleure solution est sélectionnée. Ceci autorise de remonter vers des solutions qui semblent moins intéressantes mais qui ont peut-être un meilleur voisinage. Pour éviter le phénomène de se cycler entre deux solutions la méthode a l'interdiction de visiter une solution récemment visitée. Pour cela, une liste dite la liste taboue contenant les attributs des dernières solutions visitées et tenue à jour. Chaque nouvelle solution considérée enlève de cette liste la solution la plus anciennement visitée.



L'algorithme tabou est similaire à celui du recuit simulé mais dans lequel on interdit d'effectuer des cycles.

Figure 2.2 : Organigramme de la recherche tabou

IV. Métaheuristiques à population de solutions

Contrairement aux méthodes à solution unique qui font intervenir une seule solution, les méthodes de recherche à population, comme leur nom l'indique, travaillent sur un ensemble de solutions appelé population. Le principe général de toutes ces méthodes consiste à combiner des solutions entre elles pour en former de nouvelles en essayant d'hériter des 'bonnes' caractéristiques des solutions précédentes. Un tel processus est répété jusqu'à ce qu'un critère d'arrêt soit atteint (nombre de générations maximum, nombre de générations sans améliorations, temps maximum, borne atteinte,…). Parmi ces algorithmes à population, on trouve les algorithmes génétiques, les colonies de fourmis et les essaims de particules.

Les métaheuristiques à population de solutions travaillent sur tout un ensemble de solutions qui évoluent au cours des itérations.

IV.1. Les algorithmes évolutionnaires

Les algorithmes évolutionnaires sont des techniques d'optimisation itératives et stochastiques inspirées de la théorie de l'évolution.

Cette théorie a été élaborée par Charles Darwin (1859) [12] et fondée sur le modèle biologique. Un algorithme évolutionnaire simule le comportement d'évolution des êtres vivants. Dans ce cas, on considère que l'évolution tend à produire des organismes plus adéquats à leur environnement en ayant recourt à des phénomènes comme la sélection naturelle et l'héritage génétique. Au cours du cycle de simulation, trois opérateurs interviennent généralement : la recombinaison (ou croisement), la mutation et la sélection. La recombinaison et la mutation créent de nouvelles solutions candidates, tandis que la sélection élimine les candidats les moins prometteurs. Plusieurs types d'évolution ont été développés, parmi lesquelles nous citons : les stratégies d'évolution, la programmation évolutionnaire, et les algorithmes génétiques. Les plus connus et les plus utilisés en optimisation sont les algorithmes génétiques. La première description du processus des algorithmes génétiques a été donnée par Holland [13] en 1975, puis Goldberg [14] en 1989 les a utilisés pour résoudre des problèmes concrets d'optimisation.

Les algorithmes génétiques reposent sur le principe de la sélection naturelle et de l'évolution des espèce.

Les algorithmes génétiques sont faciles à utiliser, efficaces et facilement modifiables.

La figure 2.3 présente l'organigramme classique des algorithmes génétiques. Une présentation plus détaillée des algorithmes génétiques est donnée dans le chapitre III.

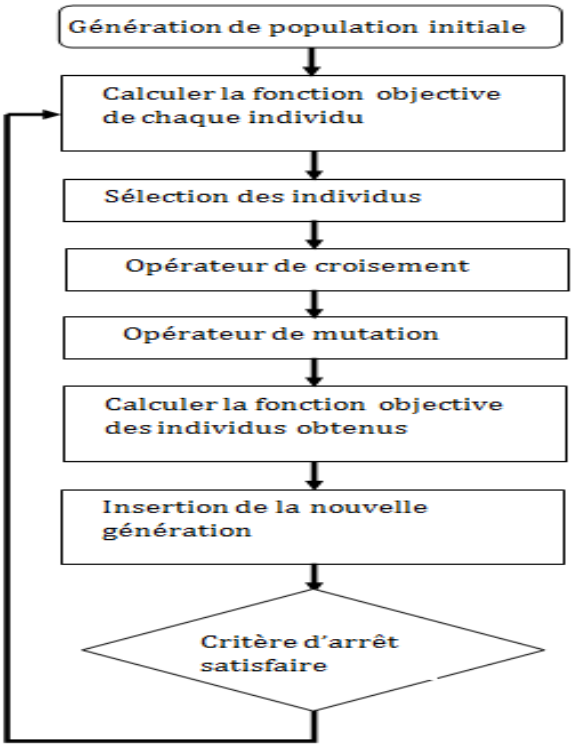


Figure 2.3 : Organigramme des algorithmes génétiques

IV.2. Les colonies de fourmis

Des constatations ont montré que les fourmis, bien qu'elles aient individuellement des capacités cognitives très limitées sont capables de sélectionner collectivement le plus court chemin entre une source de nourriture et leur nid. La coordination des activités collectives chez les fourmis est due à des mécanismes de communications indirectes via une substance volatile chimique appelée « phéromone », à laquelle les fourmis sont très sensibles. En se déplaçant du nid à la source de nourriture et vice-versa (ce qui, dans un premier temps, se fait essentiellement d'une façon aléatoire), les fourmis déposent au passage sur le sol une substance odorante appelée phéromone, ce qui a pour effet de créer une piste chimique. Les fourmis peuvent sentir ces phéromones qui ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent être utilisées par d'autres fourmis pour retrouver les sources de nourriture détectées par les autres fourmis.

Les messages chimiques utilisés par les fourmis sont un moyen de sélectionner le plus court chemin afin de transporter leur nourriture.

De plus, comme la phéromone s'évapore avec le temps, le chemin le plus long est de moins en moins emprunté et sa trace disparaît presque complètement. Par conséquent, les sentiers les plus courts seront les plus concentrés en phéromones.

Au début, les fourmis explorent différents chemins en effectuant des déplacements aléatoires. Une fois qu'un chemin menant à une source de nourriture est découvert, elles y déposent une quantité de phéromone renforçant ainsi son importance et la probabilité d'être choisi par d'autres fourmis de la colonie. D'un autre côté, les mauvais chemins auront tendance à disparaître avec l'évaporation de la phéromone. Ce procédé est basé sur le mécanisme de rétroaction positive. Il assure que les fourmis utilisent la voie d'accès la plus courte car elle sera la plus imprégnée par la phéromone.

La base du fonctionnement de cet algorithme repose sur les phéromones que déposent les fourmis lors de leurs trajets. La forte concentration des phéromones poussera les individus à utiliser majoritairement un chemin plutôt que l'autre.

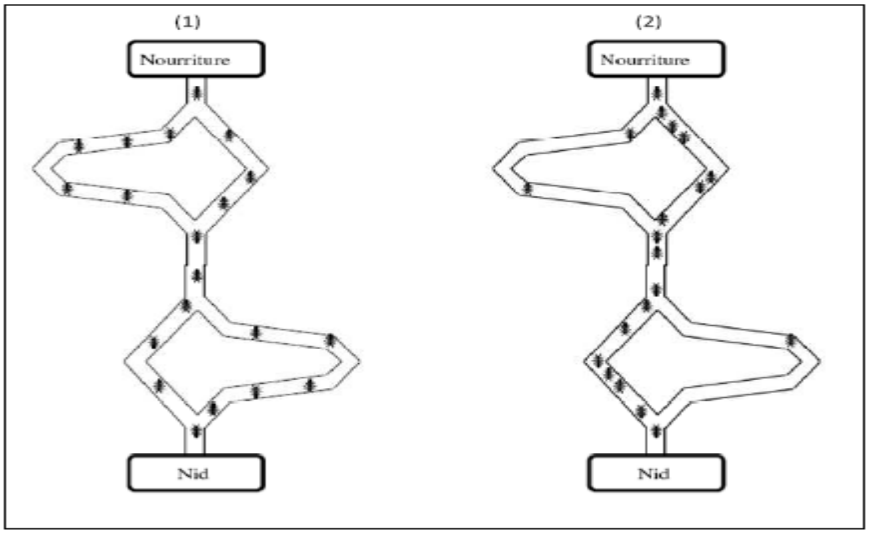


Figure 2.4 : Expérience de sélection du plus court chemin

La figure 2.4 montre l'expérience de sélection du plus court chemin. Au début de l'expérience les fourmis explorent indifféremment les deux planches du pont. A la fin de l'expérience, les fourmis ont tendance à emprunter le même chemin (le plus court).

Les algorithmes de colonies de fourmis (ACO) ont été introduits par Dorigo et *al.* [15] en 1996. Une des applications principales de la méthode originale était le problème du voyageur de commerce et depuis elle a considérablement évolué.

L'ACO consiste à travailler sur une population de solutions, chacune correspondant à une fourmi artificielle. Une structure de donnée commune, qui renferme des informations sur les quantités de phéromones artificielles accumulées dans l'espace de recherche, est utilisée pour coordonner le fonctionnement de la population. Ainsi, par analogie aux comportements des fourmis réelles, les algorithmes de colonie de fourmis sont basés sur la communication indirecte d'une colonie de fourmis artificielles. La quantité de phéromones représente une information utilisée par les fourmis pour former une solution de manière stochastique à un problème donné. Il s'agit donc d'une métaheuristique de construction aléatoire biaisée par un ensemble de données globales représentant une mémoire sur la qualité des solutions via les traces (ou quantités) de phéromones. Cet ensemble est régulièrement mis à jour par un mécanisme simulant l'évaporation de la phéromone. Il effectue des décisions probabilistes fonction des quantités de phéromones artificielles et d'une vision locale du problème via une heuristique d'information dite aussi visibilité.

La particularité des algorithmes des colonies des fourmis est qu'elles se basent sur la totalité de la population pour la recherche des solutions.

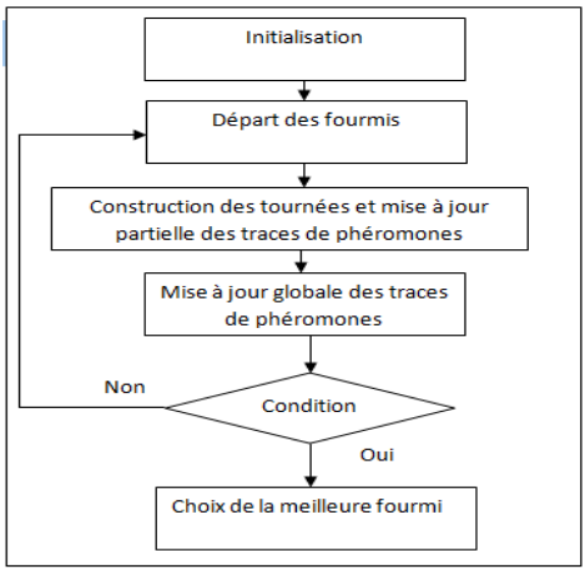


Figure 2.5 : Organigramme de Colonies de fourmis

IV.3. Les essaims de particules

L'optimisation par essaims de particules (OEP) est une méthode proposée en 1995 aux Etats Unis sous le nom de Particle Swarms Optimisation (PSO) par Eberhart et Kennedy [16] en 1995.

Cette métaheuristique s'inspire des interactions entre les individus au sein d'une population telle que les nuées d'oiseaux, les bancs de poissons, et les essaims d'abeilles. L'algorithme permet de faire converger les solutions vers des optimums locaux. Bien que cette approche ait été développée pour les milieux continus, elle a aussi été appliquée aux milieux discrets. Ces groupes d'animaux montrent des dynamiques de déplacements relativement complexes, alors qu'individuellement, ils n'ont accès qu'à des informations limitées, comme la position et la vitesse de leurs plus proches voisins. Ce type d'approches utilise une population de solutions potentielles nommé « essaim » et les individus sont appelés « particules ». Elle est basée sur la collaboration des particules entre elles en échangeant des informations permettant l'émergence de comportements complexes. Chaque particule est caractérisée par sa position courante et par le vecteur de changement de position dit la vélocité. Chaque particule dans un essaim n'évolue qu'en fonction de ses proches voisins, et non pas selon l'état global de la population à l'itération précédente. La figure 2.6 illustre la structure de base d'un algorithme d'OEP. A chaque itération de l'algorithme, la nouvelle vitesse et la nouvelle position de chaque particule sont calculées. Les particules sont ensuite évaluées et leurs meilleures positions connues sont mises à jour. La diversité au sein de la population permet à l'algorithme de ne pas rester bloqué dans un optimum local.

L'algorithmes des essaims de particules est une métaheuristique inspirée des systèmes naturels (tels que les nués d'oiseaux).

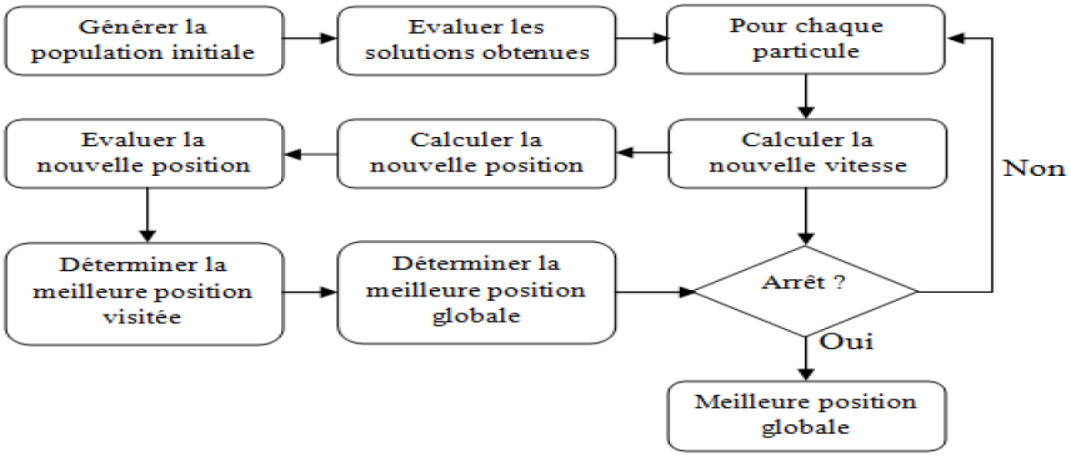


Figure 2.6 : Fonctionnement de l'algorithme des essaims de particules

V. Métaheuristiques récentes

Depuis l'apparition des métaheuristiques comme solution pour les problèmes NP-Difficiles, plusieurs études ont été menées pour élaborer d'autres. L'inspiration pour le développement de nouvelles métaheuristiques est diverse et touche de nombreux domaines tels que l'intelligence en essaims, l'éthologie, la biologie, …

Durant notre étude, nous avons trouvé de nombreuses nouvelles métaheuristiques proposées dans diverses applications. Le tableau 2.1 résume le principe de différentes approches durant plusieurs années.

Algorithme	**Auteur/ Année**	**Inspiration**
La recherche par dispersion (Scatter Search) [17]	F. Glover(1977)	Cet algorithme est différent des autres algorithmes évolutionnaires car il est basé sur le concept des méthodes systématiques pour créer des nouvelles solutions qui offrent plusieurs avantages par rapport à l'aspect aléatoire. Il utilise des méthodes systématiques de diversification et d'intensification du processus de recherche.
La recherche d'harmonie (Harmony Search) [18]	Zong Woo Geem, Joong Hoon Kim, and G. V. Loganathan (2001)	L'harmonie est une relation entre diverses ondes sonores de fréquences différentes. La meilleure harmonie est celle qui offre la meilleure expérience esthétique à l'auditeur. L'algorithme établit une similitude entre la recherche de l'harmonie dans une performance musicale et la recherche de la solution optimale à un problème d'optimisation.
Modèle d'hérédité des troupeaux de moutons (Sheep Flocks Heredity Model) [19]	Hyunchul Kim and Byungchul Ahn (2001)	Imite les troupeaux de moutons. Basé sur le concept que les moutons vivent et se reproduisent généralement au sein de leur troupeau permettant l'héritage biologique de l'intérieur d'un troupeau uniquement. Mais occasionnellement, des moutons de troupeaux différents se rencontrent pour se reproduire et propager de meilleures qualités.
(Algorithme de l'école de poissons artificiels (Artificial Fish School Algorithm) [20]	X.L. Li, Z.J. Shao and J.X. Qian (2002)	Méthode utilisée par les essaims de poissons pour rechercher de la nourriture et éviter les dangers pour la colonie. Trois comportements de base des poissons sont modélisés, à savoir la proie, l'essaim et le suivi.

Algorithme de l'évolution de la reine des abeilles (Queen-bee Evolution) [21]	S.H.Jung (2003)	L'algorithme est inspiré du concept que seule la reine des abeilles est le parent de la plupart des abeilles d'une colonie d'abeilles. Le principe de l'algorithme est que l'abeille la plus apte se croise avec d'autres abeilles sélectionnées par l'algorithme. Bien que cela augmente l'exploitation mais augmente en même temps les chances d'être piégé dans les optima locaux. Pour éviter cela, peu de membres de la population subissent une forte mutation.
Algorithme de saut de grenouille mélangé (Shuffled Frog Leaping Algorithm) [22]	Eusuff and K.E. Lansey (2003)	Imite le comportement coopératif des grenouilles lorsqu'elles recherchent de la nourriture dans un marais. Les grenouilles sont réparties en quelques groupes. Chaque grenouille se dirige soit vers la meilleure grenouille du groupe, soit vers la meilleure grenouille globale. A la fin de chaque itération, tous les groupes sont fusionnés et redistribués.
Optimisation par essaims de guêpes (Wasp Swarm Optimization) [23]	P Pinto, TA Runkler, JM Sousa (2005)	Imite le comportement social des guêpes qu'elles présentent lorsqu'elles se nourrissent et s'occupent de leurs couvées. Grâce à la coopération, les guêpes gèrent efficacement toutes les tâches sans aucune planification prédéfinie. Ils ont une hiérarchie de membres au sein d'une colonie. L'algorithme a été conçu spécifiquement en référence à l'optimisation du système logistique.
Algorithme des singes (Monkey Search) [24]	Antonio Mucherino and Onur Seref (2007)	Imite la façon dont les singes grimpent aux arbres à la recherche de nourriture. Les branches des arbres représentent des perturbations entre deux sources de nourriture probables. Les singes, en tant qu'agents de recherche, marquent et mettent à jour ultérieurement ces branches lorsqu'ils montent et descendent dans l'arbre.
Algorithme de reproduction de bourdons (Bumble Bees Mating Optimization) [25]	F. Comellas and J. Martinez Navarro (2009)	Imite la colonie d'insectes bourdons. Met en corrélation la forme physique de chaque bourdon avec sa durée de vie pour éliminer les pires solutions et retenir les meilleures.

Algorithme de coco (Cuckoo Search Algorithm) [26]	Xin-She Yang and Suash Deb (2009)	Inspiré du comportement parasitaire des couvées de certaines espèces de coucous et des vols de prélèvement de certains oiseaux.
Algorithme de lucioles (Firefly Algorithm) [27]	Xin-She Yang (2009)	Imite la propriété de lumière clignotante des lucioles. Modélise les membres de la population comme un ensemble de lucioles où chaque luciole est attirée par une autre dans la magnitude de son intensité lumineuse et de sa distance mutuelle.
Algorithme de chasse (Hunting Search) [28]	R. Oftadeh , M. J. Mahjoob (2009)	Imite les comportements de chasse des animaux comme les lions, les loups et les dauphins. L'algorithme imite la méthode de chasse dans laquelle les animaux de chasse recherchent une proie en groupe, l'encerclent et resserrent progressivement l'encerclement pour finir par attraper la proie.
Algorithme de chauvesouris (Bat Algorithm) [29]	Xin-She Yang (2010)	Inspiré du comportement d'écholocation des chauves-souris utilisé pour rechercher les directions et l'emplacement des insectes.
Optimisation par colonies des termites (Termite Colony Optimization) [30]	Ramin Hedayatzadeh, Foad Akhavan Salmassi, Reza Akbari, Koorush Ziarati (2010)	Inspiré des méthodes utilisées par les termites pour ajuster leurs chemins de recherche. Bien que les termites se déplacent au hasard dans l'espace de recherche, leurs trajectoires de recherche sont influencées par les directions de plus de phéromones.
Algorithme de la société anarchique (Anarchic Society Optimization) [31]	H. Shayeghi and J. Dadashpour (2012)	Modélise le problème sous forme d'un groupe d'individus anarchistes et aventureux et n'aimant pas la stabilité pour pouvoir s'échapper des minima locaux.
Algorithme de colonies de bactéries (Bacterial Colony Optimization) [32]	Ben Niu and Hong Wang (2012)	Basé sur les comportements de développement d'Escherichia Coli comme la chimiotaxie, la communication, l'élimination, la reproduction et la migration.
Algorithme des loups (Wolf Search Algorithm) [33]	Rui Tang, S. Fong, Xin-She Yang, and S. Deb (2012)	Cet algorithme imite le mécanisme dont les loups recherchent de la nourriture et survivent aux dangers autour de leur habitat. Chaque agent recherche indépendamment et conserve ses positions précédentes dans sa mémoire. Il

		ne fusionne avec n'importe quel autre agent que lorsqu'il le trouve meilleur que tous ses précédemment visités dans sa mémoire.
Algorithme du chien sauvage d'Afrique (African Wild Dog Algorithm) [34]	C. Subramanian , A.S.S. Sekar and K. Subramanian (2013)	Inspiré du comportement de chasse de l'espèce des chiens sauvages d'Afriques : les Lycaons. L'algorithme ne nécessite comme paramètre que la taille du groupe et le critère d'arrêt.
Algorithme des pinguoins (Penguins Search Optimization Algorithm) [35]	Y. Gheraibia, A. Moussaoui (2013)	L'algorithme est inspiré par le comportement coopératif des pingouins pendant qu'ils chassent. Chaque pingouin commence le processus de recherche à partir de sa propre localité et informe les autres membres du groupe de sa position et des poissons trouvés. Le meilleur résultat est sélectionné en identifiant le groupe qui a chassé le plus de poissons.
Optimisation de l'odeur des requins (Shark Smell Optimization) [36]	Oveis Abedinia, Nima Amjady, and Ali Ghasemi (2014)	Les requins sont l'un des chasseurs les plus intelligents. L'algorithme s'inspire des capacités olfactives du requin. Les étapes mathématiques de l'algorithme imitent le comportement intelligent du requin utilisé pour trouver la source de l'odeur dans l'environnement océanique.
Optimisation par le singe araigné (Spider Monkey Optimization) [37]	Jagdish Chand Bansal, Harish Sharma, Shimpi Singh Jadon, Maurice Clerc (2014)	L'algorithme est inspiré du le comportement de recherche de nourriture chez les singes araignées. Les singes araignées suivent une structure sociale basée sur la fission-fusion. Ils modifient la taille des groupes de plus petits à plus grands ou vice-versa en fonction de la quantité de nourriture disponible.
Algorithme des fourmilions (Ant Lion Algorithm) [38]	S. Mirjalili (2015)	Cette métaheuristique imite le comportement de capture de proie chez l'espèce des fourmis Myrmeleontidae connues sous le nom de fourmilions.
Algorithme d'éléphant (Elephant Search Algorithm) [39]	Suash Deb, Simon Fong, and Zhonghuan Tian (2015)	Dans cet algorithme, les membres de la population sont appelés éléphants d'un troupeau. Les males sont des agents de recherche pour explorer l'espace tandis que les membres femelles sont des agents de recherche locaux pour exploiter l'espace de recherche localement.

Algorithme de Jaguar avec comportement d'apprentissage (Jaguar Algorithm with Learning Behavior) [40]	Chin-Chi Chen, Yung-Che Tsai, I-I Liu, Chia-Chun Lai, Yi-Ting Yeh, Shu-Yu Kuo, Yao-Hsin Chou (2015)	Inspiré du comportement de chasse des jaguars. Une fois qu'un jaguar identifie sa proie, il se déplace rapidement et directement vers la cible. Les jaguars chassent également en équipe. L'algorithme imite les caractéristiques de chasse du jaguar pour équilibrer l'exploitation et l'exploration de la recherche.
Algorithme de Bœuf Africain (African Buffalo Optimization) [41]	J.B. Odili and M.N. Mohmad Kahar (2016)	Inspiré des compétences et de l'intelligence organisationnelle dont font preuve les buffles africain pour la recherche de nourriture.
Algorithme des chameaux (Camel Algorithm) [42]	M. K. Ibrahim, R. S. Ali, (2016)	Inspiré par le comportement de déplacement du chameau dans le désert. Utilise plusieurs facteurs et opérateurs tels que l'effet de la température, l'approvisionnement en eau, l'endurance, la visibilité, les conditions du terrain, etc.
Algorithme des lions (Lion Optimization Algorithm) [43]	Maziar Yazdani and Fariborz Jolai (2016)	Motivé par le comportement social et coopératif des lions. Les lions résident en deux catégories résidents et nomades. Cinq lions femelles et de leurs petits des deux sexes appelés résidents. Les lions mâles adultes quittent la troupe et se déplacent librement. Les résidents sont modélisés comme des agents de recherche intensive locale tandis que les nomades représentent des agents pour explorer l'espace de recherche.
Optimisation par évaporation d'eau (Water Evaporation Optimization) [44]	A. Kaveh and T. Bakhshpoori (2016)	L'algorithme est basé sur les propriétés d'évaporation des molécules d'eau de surfaces solides avec différentes mouillabilités. Les molécules d'eau sont considérées comme des membres de la population et la surface solide comme un espace de recherche. La mouillabilité de la surface ainsi que d'autres propriétés moléculaires sont codées comme différents paramètres de recherche.

Tableau 2.1: Liste des métaheuristiques

VI. L'équilibre entre l'intensification et la diversification dans une métaheuristique

Toutes les métaheuristiques s'appuient sur un équilibre entre deux stratégies : l'intensification de la recherche et la diversification de celle-ci afin d'améliorer l'efficacité de la technique. D'un côté, l'intensification consiste à explorer en détails une région de l'espace de recherche, jugée prometteuse. En d'autres termes, elle offre un élargissement temporaire du voisinage des solutions déjà trouvées dans le but de visiter un ensemble de solutions voisines.

La diversification met en place des stratégies qui permettent d'explorer un plus grand espace de solutions et d'échapper à des minima locaux. Ayant pour objectif de diriger la procédure de recherche vers des régions inexplorées de l'espace de recherche.

La stratégie de diversification la plus simple consiste à redémarrer périodiquement (ou sous certaines conditions) la recherche à partir d'une solution générée aléatoirement ou choisie judicieusement, dans une région qui n'a pas encore été visitée de l'espace de recherche.

Ne pas préserver cet équilibre conduit à une convergence trop rapide vers des minima locaux (manque de diversification) ou à une exploration trop longue (manque d'intensification).

VII. Conclusion

Ce chapitre a décrit les principales métaheuristiques, leurs origines, leurs principes, et leurs algorithmes de bases. De plus nous avons présenté une section qui regroupe de nombreuses métaheuristiques récentes développées récemment en décrivant brièvement le principe de chacune.

Le chapitre suivant est consacré à une présentation détaillée de l'algorithme génétique, son origine, son principe et en particulier à sa variante sous la forme parallèle dit algorithme génétique modèle des îles.

De plus nous présentons une nouvelle variante du ce dernier, que nous avons développé dite algorithme modèle des îles avec gestion de migration que nous appliquons pour la résolution d'un problème d'ordonnancement dans un atelier Flow Shop.

CHAPITRE 3

Algorithme Génétique Modèles des îles avec Gestion de Migration (IMGAMM)

Dans ce chapitre, nous commençons par présenter l'algorithme génétique, son origine, son principe et sa variante parallèle dite algorithme génétique modèle des îles. Dans ce dernier, nous avons introduit un mécanisme en vue d'améliorer la technique de recherche en contrôlant la procédure de diversification dans l'algorithme. En effet, un paramètre de gestion de migration est introduit pour la mesure de dissimilitude entre les individus lors de la migration. Le nouvel algorithme ainsi obtenu portera le nom de ***« Algorithme génétique modèle des îles avec gestion de migration » : Island Model Genetic Algorithm with Migration Management (IMGAMM)*** *et constituera une des contributions de cette thèse. La suite de ce chapitre sera consacrée à l'application des algorithmes génétique, modèle des îles et IMGAMM pour la résolution d'un problème d'ordonnancement dans un atelier Flow Shop. Une analyse de sensibilité pour étudier l'influence des paramètres des algorithmes sur la qualité des solutions obtenues sera ensuite récapitulée. Pour conclure ce chapitre nous présentons une comparaison des résultats des trois algorithmes cités ci-dessus.*

I. Introduction

Les métaheuristiques parallèles visent principalement à explorer tout l'espace de recherche en effectuant des recherches parallèles ciblant plusieurs zones de cet espace. Elles s'avèrent également beaucoup plus robustes que les versions séquentielles pour traiter les différents types de problèmes. Ils nécessitent également des efforts d'étalonnage des paramètres moins étendus et moins coûteux.

Le parallélisme est l'un des mécanismes les plus prometteurs pour améliorer la performance des algorithmes génétiques. En effet, un algorithme génétique parallèle doit travailler non seulement sur une population de solutions mais plutôt sur plusieurs populations en même temps. C'est le cas de l'algorithme génétique modèle des îles, dont le principe est de créer des sous-populations, nommées des îles, qui peuvent communiquer entre eux en faisant un échange d'individus périodiquement. Cet échange est nommé la migration.

La première contribution de cette thèse constitue en l'amélioration de l'algorithme génétique modèle des îles en introduisant un paramètre qui contrôle la diversité dans les différentes îles. Contrairement à l'algorithme modèle des îles basique,

où le choix de d'individus à échanger peut être : aléatoire, les plus mauvais ou les meilleurs, nous définissons ici un paramètre de dissimilitude qui va contrôler cet échange. Le principe est de faire migrer un individu d'une île initiale vers une nouvelle île où le taux de dissimilitude de cet individu avec tous les autres individus de la nouvelle île est fixé par le paramètre de gestion de migration Δ. Nous donnons le nom de l'algorithme génétique modèle des îles avec gestion de migration (Island Model Genetic Algorithm with Migration Management (IMGAMM)) à la variante de l'algorithme génétique modèle des îles que nous venons de définir.

II. Les algorithmes génétiques

La sélection naturelle, exprimée aussi par la survie de l'individu le plus fort, a été défini en 1929 par le chercheur et naturaliste Charles Darwin.

Dans son livre [45] Charles Darwin présente la sélection naturelle comme la préservation des traits qui permettent à un individu d'être plus apte à résister lorsqu'il se reproduit, en éliminant les traits qui le rendent plus faible ; ce processus est appelé « l'évolution » [46].

Les caractéristiques de chaque individu sont stockées dans un ensemble nommé « chromosome », celui-ci est composé par un ensemble d'unités appelées « gènes ». Les gènes les plus adaptés sont transmis de génération en génération au cours du processus de recombinaison appelé « croisement ».

Les algorithmes génétiques ont été introduits pour la première fois par John Holland à la fin des années 60, qui a développé une technique qui permettait aux programmes informatiques d'imiter le processus d'évolution. Après la publication de son livre [47], le terme « algorithme génétique » est devenu populaire. En 1989, Goldberg a publié un livre [48] fournissant une base solide pour la compréhension des algorithmes génétiques.

Le but des algorithmes génétiques est d'optimiser une fonction prédéfinie, appelée fonction objectif, ou fitness, ils examinent un ensemble de solutions candidates appelé « population » d'individus. Dans un algorithme génétique, un individu représente un chromosome.

Avant d'aborder les algorithmes génétiques, il est nécessaire d'introduire un certain nombre de termes. Par conséquent, nous définissons les termes suivants :

Définition 1 : (Gène)

On appellera gène la suite de symboles qui codent la valeur d'une variable.

Définition 2 : (Chromosome)

Un chromosome est constitué d'une séquence finie de gènes. Dans un problème d'optimisation, un chromosome représente une solution possible du problème étudié, en d'autres termes c'est un élément de l'espace de recherche. est généralement représenté sous forme de chaînes.

Définition 3 : (Individu)

Un individu est une solution potentielle. Dans la plupart des cas un individu est représenté par un seul chromosome.

Définition 4 : (Population)

Une population est un groupe d'individus sur lequel l'algorithme génétique effectue un certain nombre d'opérations.

Définition 5 : (Génération)

Une génération est l'ensemble des opérations qui permettent de passer d'une population à une autre. Ces opérations sont généralement la sélection des individus de la population courante, le croisement des individus, et la mutation de nouveaux individus. Par convention, le nombre d'individu est le même de génération à l'autre, en d'autres termes, il y a autant de morts que de nouveau-nés.

Définition 6 : (Fonction objectif)

Pour évaluer la pertinence d'une solution par rapport à une autre, on introduit ce que l'on nomme une fonction d'adaptation (ou bien fonction d'évaluation ou de fitness ou encore la fonction objectif) qui correspond à l'utilité de la solution par rapport au problème.

II.1. Les étapes d'un algorithme génétique classique

L'algorithme génétique commence par générer une population initiale d'individus, pour chacun d'entre eux on calcule la fitness (qui représente la valeur de la fonction objectif de la solution associée à l'individu considéré), l'opérateur de sélection permet de choisir les individus qui vont se reproduire, ensuite l'opérateur de croisement les combine (deux à deux) pour donner naissance à de nouveaux individus. Les individus résultants (les enfants) peuvent être mutés par un opérateur de mutation.

Les trois opérateurs de sélection, croisement, et mutation permettent de générer une nouvelle population d'individus (dite nouvelle génération). De génération en

génération, on obtient des individus plus performants que ceux des anciennes générations. Le processus s'arrête lorsqu'un critère d'arrêt est atteint. Avant de présenter l'algorithme génétique, nous détaillons dans cette section les quatre points essentiels pour la compréhension du fonctionnement des algorithmes génétiques.

II.1.1. Le codage

Lorsqu'on aborde les algorithmes génétiques, la première tâche consiste à trouver une modélisation adéquate de l'individu qui facilite la description du problème et respecte ses contraintes. Cette modélisation, appelée codage, permet de représenter les solutions sous forme de chromosomes. Le codage des solutions peut être binaire (c'est-à-dire les gènes contiennent seulement les valeurs de 1 ou 0) ou réel.

II.1.2. Sélection des individus

Cette étape consiste à choisir les individus qui vont se reproduire. Les individus sélectionnés sont nommés « parents ». en d'autres termes, les individus passent par un processus permettant la sélection des parents de la population suivante. Parmi les techniques de sélection, on cite les sélections par la roulette, par le tournoi et par le classement.

- **La sélection par roulette (RWS : Roulette Wheel Selection) :** Cette méthode exploite la métaphore d'une roulette de casino pour laquelle chaque individu de la population occupe une section de la roue proportionnelle à sa valeur d'évaluation. Ainsi, la position d'arrêt de la bille indique l'individu sélectionné.
- **La sélection par tournoi :** La sélection par tournoi consiste à choisir aléatoirement deux ou plusieurs chromosomes au hasard et permettre à celui qui a une meilleure évaluation, d'être sélectionné. L'avantage de cette sélection par rapport à la sélection par la roulette est d'éviter qu'un individu très fort soit sélectionné plusieurs fois. Par contre, en utilisant cette méthode, le meilleur individu peut ne pas être sélectionné et ainsi le champ d'exploration est réduit.
- **La sélection par classement :** La sélection par classement consiste à trier les individus de la population en fonction de leurs valeurs d'évaluation. Ainsi seuls les individus les plus forts sont conservés. L'inconvénient de cette méthode est la convergence prématurée de l'algorithme. En effet, il est parfois nécessaire de garder des individus jugés plus ou moins bons pour conserver la diversité au niveau de la population. Aussi certains

individus faibles, contenant de bonnes configurations, peuvent ne pas être sélectionnés, ainsi le champ d'exploration est réduit.

II.1.3. Opérateur de croisement

Les individus sélectionnés pour la reproduction sont soumis au croisement qui consiste à combiner deux individus sélectionnés pour donner naissance à deux nouveaux individus nommés « enfant ». En effet, le croisement permet de créer de nouvelles séquences de gènes pour les chromosomes enfants à partir d'une base de configurations des séquences héritées des chromosomes parents. On distingue le croisement à un point, le croisement à plusieurs points et le croisement uniforme.

II.1.3.1. Le croisement à un point

Dans ce type de croisement, Une position *k* est choisie aléatoirement le long des chromosomes parents. Les chromosomes enfants sont le fruit d'un échange de deux parties des parents. La figure (2) illustre un exemple de croisement à un point.

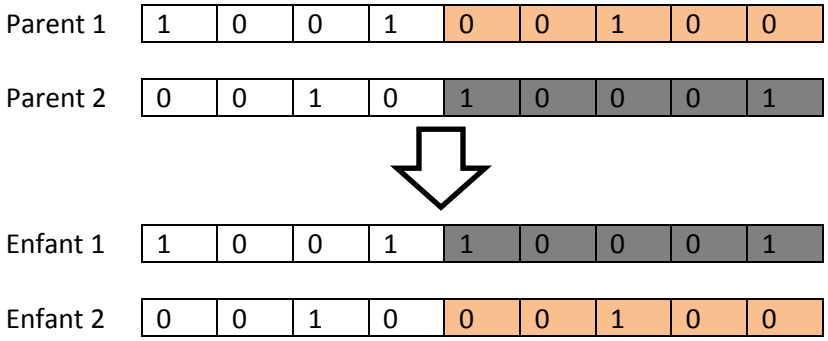


Figure 3.1 : Exemple de croisement à un point

II.1.3.2. Le croisement à plusieurs points (multi-points)

Ce croisement est similaire au précédent sauf que plusieurs points de croisement y sont impliqués. La figure II.6 illustre un cas du croisement multi-points qui est le croisement bi-points. Il consiste à choisir deux points de croisement *k1* et *k2* et à échanger les segments des deux parents.

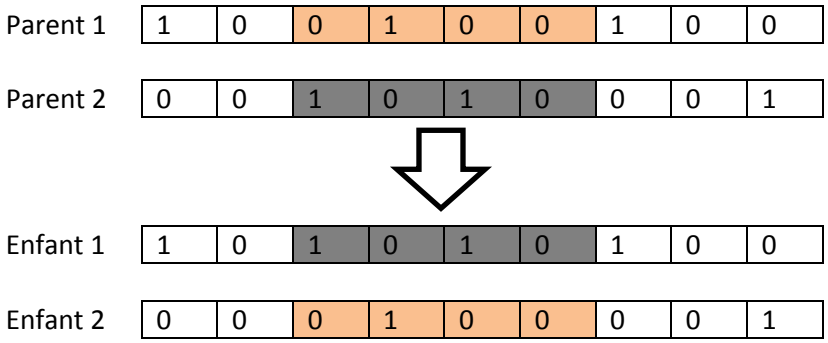


Figure 3.2 : Exemple de croisement multi-points

II.1.3.3. Le croisement uniforme (dans le cas de codage binaire)

Parmi les approches les plus connues du croisement uniforme, on cite l'utilisation du vecteur masque, le masque étant une suite aléatoire de 0 et 1 de la taille du chromosome. Il sert à déterminer de quel parent sont repris les gènes. La figure 3.2 présente un exemple du croisement uniforme utilisant le masque : en présence du 0, l'enfant hérite du parent 1 et en présence de 1, l'enfant hérite du parent 2.

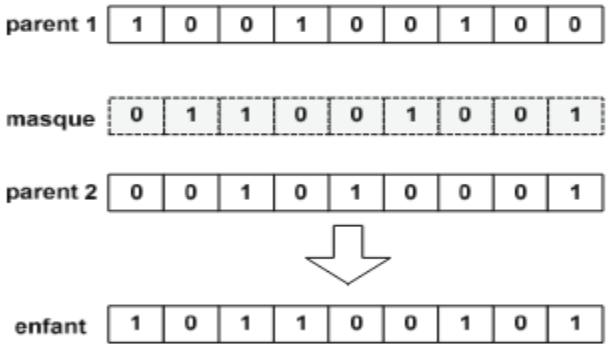


Figure 3.3 : Exemple de croisement uniforme

II.1.4 Opérateur de mutation

La mutation est une modification aléatoire légère au niveau du chromosome (individu), c'est-à-dire, une modification d'un ou plusieurs gènes aléatoirement dans le chromosome. Elle est appliquée sur les enfants obtenus par le croisement et elle permet d'avoir une certaine diversité dans la population. L'algorithme 3.1 représente les différentes étapes d'un algorithme génétique classique.

Algorithme 3.1 : Algorithme génétique classique

(1) **Initialiser** la taille de population
(2) **Générer** une population initiale de taille *Taille pop*
(3) **Tant que** le critère d'arrêt n'est pas atteint
(4) **Evaluer** chaque individu de la population (calcul de la fitness)
(5) **Sélectionner** deux individus (parents)
(6) **Effectuer** l'opérateur croisement
(7) **Effectuer** l'opérateur de mutation
(8) **Remplacer** les parents par des descendants
Fin pour
Fin tant que
(9) La solution finale est la solution minimale de toute les solutions trouvées

III. L'algorithme génétique parallèle modèle des îles

Pour rendre un algorithme génétique plus performant et plus robuste lors de la résolution des problèmes, plusieurs mécanismes ont été proposés. Certaines techniques proposent l'hybridation, qui consiste à combiner l'algorithme génétique avec une autre (ou plus) approche de recherche (méthode exacte, heuristique ou encore métaheuristique) en vue d'avoir compensation des faiblesses des techniques utilisées.
L'une des premières techniques qui ont proposé l'hybridation de l'algorithme génétique on trouve l'algorithme mémétique [49] qui combine l'algorithme génétique avec une procédure de recherche locale.

Allant plus loin pour résoudre le problème de la convergence prématurée de l'algorithme génétique, d'autres techniques comme l'algorithme mémétique avec gestion de population (MA/PM) [50] proposent le contrôle de diversité de la population. Dans un algorithme mémétique avec gestion de population, l'idée consiste à utiliser un paramètre qui permet de mesurer la ressemblance (ou même la dissemblance) entre les individus de chaque génération.

Un autre mécanisme proposé pour l'amélioration de la performance de l'algorithme génétique est le parallélisme [51] vu que le nombre de solutions examinées augmente en utilisant une procédure parallèle.

Plusieurs travaux utilisant les algorithmes génétiques parallèles ont été développés dans la littérature [52-55]. Ils ont montré leur efficacité et leur facilité d'implémentation [56-59].

Il existe principalement deux modèles d'algorithmes génétiques parallèles: le modèle maître-esclave et le modèle en îles. Dans le premier modèle, un processeur maître est dédié aux étapes de sélection, de croisement, de mutation et de remplacement et les processeurs esclaves s'occupent de calculer la fitness des individus.

Dans le deuxième modèle (modèle des îles), la population est divisée en plusieurs sous-populations réparties sur des processeurs. Chaque processeur utilise les algorithmes génétiques conventionnels.

Dans le modèle des îles, les sous-populations (nommées les îles) peuvent communiquer entre elles sous forme de migration d'individus (Figure3.4). En effet, certains individus sélectionnés parmi une sous-population peuvent être déplacés vers une autre.

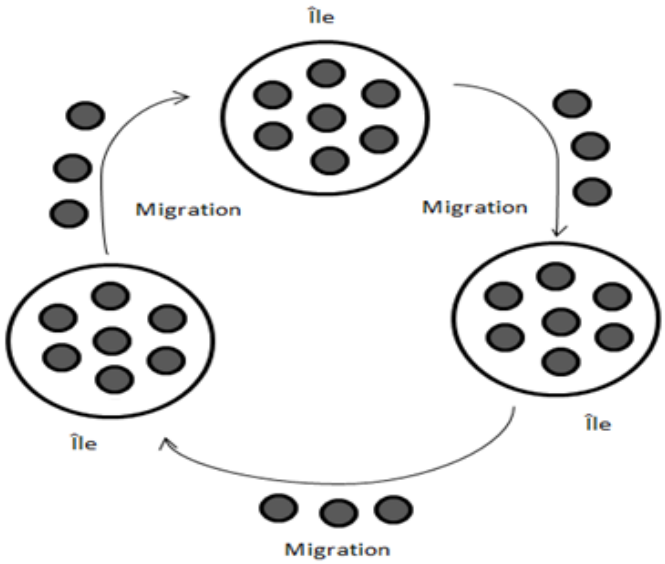


Figure 3.4 : Exemple illustratif d'un Algorithme génétique modèle des îles

Le choix des individus candidats à la migration peut être conditionné selon l'utilisateur mais, très souvent, il s'agit de faire écouler une période de temps pendant laquelle aucune migration n'est permise pour ensuite choisir un individu (le meilleur, le mauvais ou aléatoirement) et le substituer avec un autre (le meilleur, le mauvais ou l'aléatoire) d'une autre sous-population voisine. La migration peut être régie par deux paramètres: l'intervalle de migration et le taux de migration. En d'autres termes, le choix de l'individu candidat à la migration est soit inconditionné, soit conditionné par rapport à l'île à laquelle il appartient, et aucune contrainte d'insertion dans la nouvelle île n'est imposée. L'algorithme 3.2 présente les différentes étapes de l'algorithme génétique parallèle modèle des îles.

Algorithme 3.2 : Algorithme génétique parallèle modèle des îles

(1) **Initialiser** la taille de population
(2) **Générer** une population initiale de taille *Taille pop*
(3) **Subdiviser** la population créée en k sous-populations
(4) **Tant que** le critère d'arrêt n'est pas atteint
(5) **Pour chaque île**
Sélectionner deux personnes (parents)
Effectuer l'opérateur croisement
Effectuer l'opérateur de mutation
Remplacer les parents par des descendants
Fin pour
Fin tant que
(6) La solution finale est la solution minimale atteinte parmi toutes les solutions minimales obtenues dans toutes les îles

IV. L'algorithme génétique modèle des îles avec gestion de migration (IMGAMM)

Dans un algorithme génétique parallèle en modèle des îles, la migration joue un rôle primordial sur la façon dont la solution converge. En effet, le choix des solutions candidates pour la migration en termes de qualité de la solution (le meilleur, le plus mauvais ou aléatoire) influe énormément sur la qualité des solutions obtenues. Sachant que le problème majeur des algorithmes génétiques est la convergence prématurée, un mauvais choix ou un choix inadéquat peut créer un faible équilibre entre l'intensification et la diversification dans les sous-populations de l'espace de recherche. Une technique de recherche efficace ne doit pas se concentrer sur seulement une région de l'espace de recherche. Elle doit maintenir un équilibre entre l'intensification et la diversification de la recherche.

Nous avons constaté que dans l'algorithme génétique modèle des îles la migration se fait soit : sans contrainte sur les individus (le choix se fait d'une manière aléatoire), ou en classant l'individu dans son île (le meilleur ou le plus mauvais). En d'autres termes, si une contrainte doit être imposée, alors elle l'est dans l'île qui va émettre l'individu et non l'île réceptive. Ce qui ne garantit pas une diversification de l'espace de recherche dans l'île réceptive. Pour combler cette lacune, nous avons défini une nouvelle démarche pour la gestion de processus de la migration.

Le principe de la démarche de migration que nous proposons se base sur le maintien de l'équilibre entre l'intensification et la diversification au niveau des sous-populations (îles) de l'algorithme de génération en génération.

Si tous les individus (les solutions) dans une sous-population sont suffisamment différents (des solutions distinctes), ceci dit avoir une bonne diversification dans ces sous-populations (les îles). En plus, si chaque individu qui va être inséré dans une sous-population est aussi suffisamment différent des individus existants dans cette sous-population, ceci maintient cette diversification. Nous proposons donc d'utiliser un paramètre de gestion de migration des individus d'une île vers une autre noté Δ.

Le paramètre de gestion de migration Δ permet de mesurer la distance d'une solution à une autre, le mot distance ici ne mesure pas une longueur mais une distinction (dissimilitude). En effet, le paramètre Δ mesure la ressemblance (ou la dissemblance) entre chaque deux solutions en comparant le nombre de gènes en commun et l'ordre dont ils sont placés.

Par exemple, si Δ est fixé à 3 et étant donné deux solutions à examiner, notre idée consiste à comparer les 3 premiers gènes des deux solutions un par un. Si les 3 gènes se ressemblent alors ces deux solutions ne doivent pas appartenir à la même île. De ce fait, la migration des individus d'une île vers une autre n'est autorisée que si l'individu candidat à la migration vérifie la contrainte de dissemblance avec tous les individus du l'île vers laquelle il est supposé migrer. En d'autres termes, il ne doit ressembler à aucun des individus de cette île (Figure 3.5). Par conséquence, l'algorithme que nous proposons est nommé « Algorithme Génétique Modèle des îles avec Gestion de Migration : The Island Model Genetic Algorithm with Migration Management (IMGAMM) ».

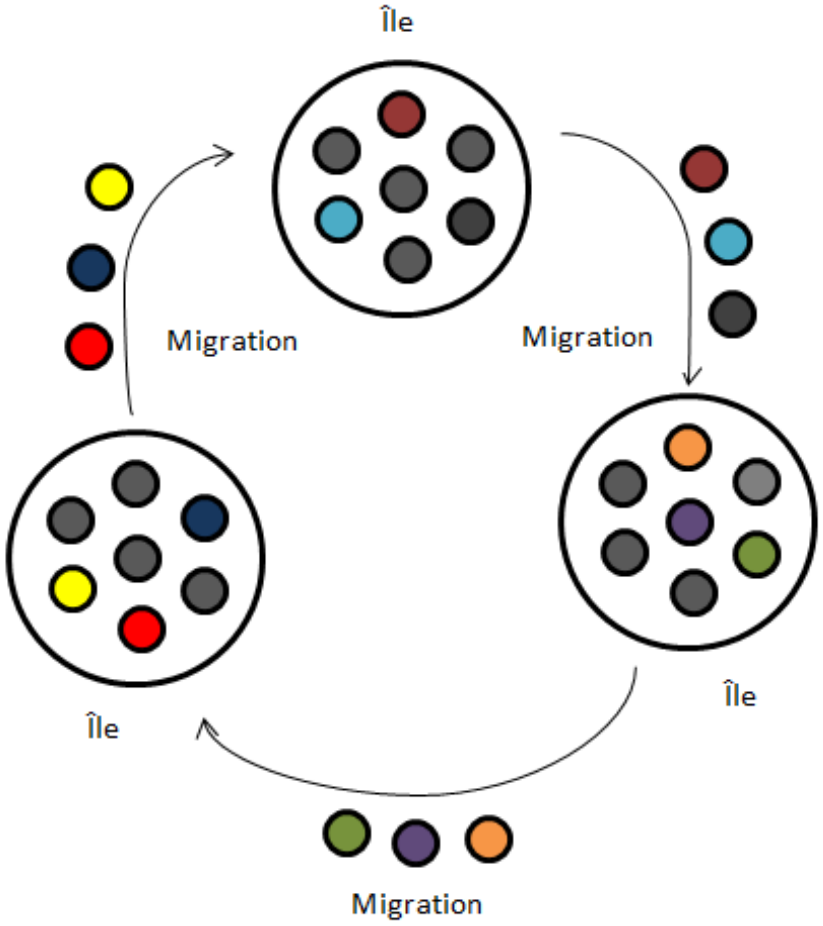


Figure 3.5 : Exemple illustratif de l'IMGAMM

L'algorithme 3.3 présente les différentes étapes de l'algorithme IMGAMM.

Algorithme 3.3 : Algorithme IMGAMM

(1) **Initialiser** la taille de population, le taux de migration et le paramètre de gestion de migration δ
(2) **Générer** une population initiale de taille *Taille pop*
(3) **Subdiviser** la population créée en k sous-populations
(4) **Tant que** le critère d'arrêt n'est pas atteint
(5) **Pour chaque île**
Sélectionnez au hasard *n* individus à migrer de l'île
Sélectionnez au hasard deux îles *Li* et *Lj*
Pour chaque *xk* individuel sélectionné pour être migré de *Li* vers *Lj*
Si *d(xk,xl)≥δ* Alors
Insérer *xk* dans *Lj* Autre Choisissez au hasard un autre xk de Ii et Fin pour
(6) **Sélectionner** deux personnes (parents)
(7) **Effectuer** l'opérateur croisement
(8) **Effectuer** l'opérateur de mutation
(9) **Remplacer** les parents par des descendants
Fin pour
Fin tant que
(10) La solution finale est la solution minimale atteinte parmi toutes les solutions minimales obtenues dans toutes les îles

V. Description du problème

Un atelier Flow Shop peut être définit par un ensemble de jobs (produits) n (J1,…, Jn)qui doivent être traités sur un ensemble de machines m (m1,…, mm). Chaque job Ji (i=1, …, n) doit être traité sur toutes les machines mj (j=1,…,m)Durant un temps de traitement noté *Pij*.

La résolution d'un problème d'ordonnancement dans un atelier Flow Shop avec minimisation de makespan consiste à trouver le meilleur séquencement des jobs sur les différentes machines de sorte à minimiser la fonction makespan. Rappelons que le makespan peut être défini comme étant le temps calculé entre le début de traitement du premier job sur la première machine et le temps de traitement du dernier produit sur la dernière machine du système.

Certaines contraintes sont à considérer :

- Il n'y a aucune contrainte de précédence entre les jobs,
- Chaque machine mj ne peut traiter qu'un seul job Ji { la fois,
- Chaque job Ji ne peut être traité que par une seule machine mj à la fois,

- Le temps de déplacement des jobs entre les machines est négligeable,

La formulation mathématique du problème est la suivante :

$f=min(Cmax)$ (**1**)

$Cmax=max\ \{ Cij \}$ for all $i=1,...,n;\ j=1,...,m$ (**2**)

$Cij = Sij + Pij$ for all $i=1,...,n;\ j=1,...,m$ (**3**)

n: le nombre de jobs

m: le nombre de machines

Sij: le temps de début de traitement du job *Ji* sur la machine *mj*

Pij: le temps de traitement du job *Ji* sur la machine *mj*

Cij: la date de fin de traitement du job *Ji* sur la machine *mj*

Le but étant de déterminer la meilleure séquence qui permet de trouver le Cmax minimum du système.

Comme première étape, nous avons adapté l'algorithme génétique classique, ensuite nous avons adapté l'algorithme génétique sous sa forme modèle des îles, et finalement nous avons développé et adapté l'algorithme génétique modèle des îles avec gestion de migration IMGAMM.

VI. Adaptation de l'algorithme génétique

Dans cette section, nous présentons l'adaptation de l'algorithme génétique pour la résolution du problème décrit dans la section précédente.

La première étape lors de l'adaptation de métaheuristiques est la représentation ou la construction de solutions. Dans un algorithme génétique, cette représentation est dite « le codage des solutions ».

- **Codage des solutions :** dans un problème d'ordonnancement Flow Shop une solution réalisable du problème est une séquence de jobs. Par exemple, s'il y a 10 jobs à traiter (j_1,j_2, ..., j_{10}), un exemple de solution réalisable peut être donné par la Figure 3.6.

J_8	J_4	J_9	J_6	J_2	J_3	J_7	J_{10}	J_1	J_5

Figure 3.6 : Exemple de codage de solution

- **Opérateur de croisement :** nous avons choisi le croisement à un point. En conséquence, l'opérateur de croisement se fait en choisissant le point de croisement le long du chromosome sélectionné, puis en gardant la partie gauche du premier chromosome (parent 1) et en cherchant dans le deuxième

chromosome (parent 2) les gènes restants pour compléter la nouvelle solution construite, et faire de même stratégie avec la deuxième partie du chromosome. Un exemple illustratif est donné dans la Figure 3.8.

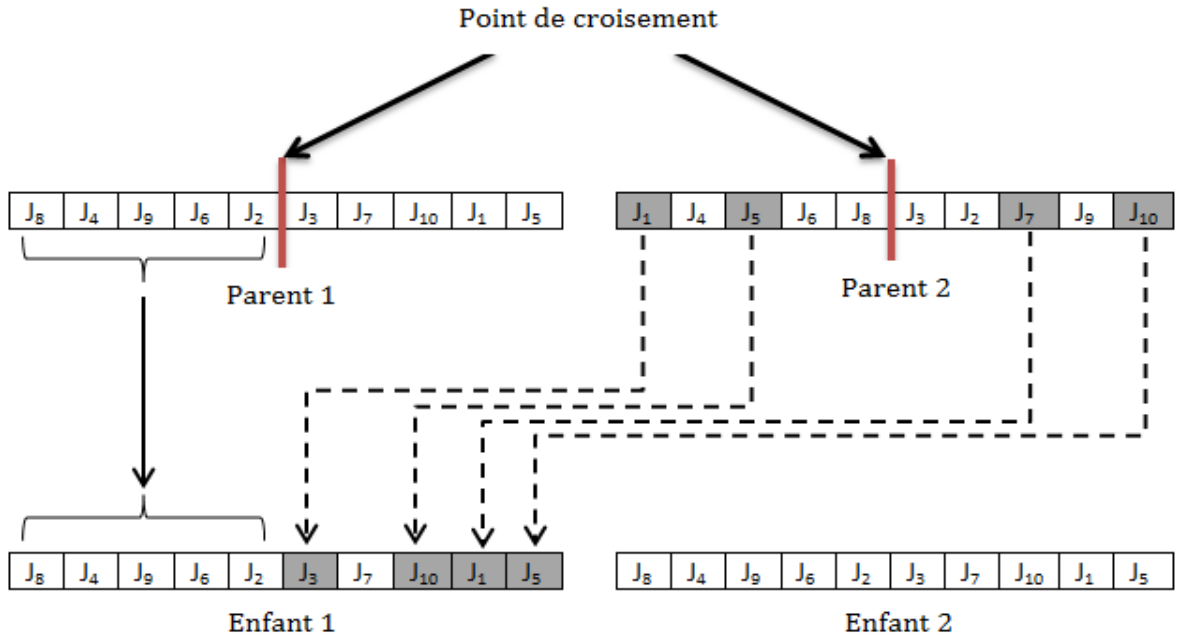


Figure 3.7 : Exemple d'un opérateur de croisement appliqué sur deux individus

L'opérateur de croisement utilisé dans cette étude complète les gènes restants chez chaque parent à partir du l'autre pour former deux nouveaux chromosomes (nommés descendants ou enfants).

- **L'opérateur de mutation :** Chaque chromosome sera muté en sélectionnant aléatoirement une paire de jobs (gènes) afin de les permuter. La figure 3.9 présente un exemple d'application de l'opérateur de mutation sur deux séquences de jobs.

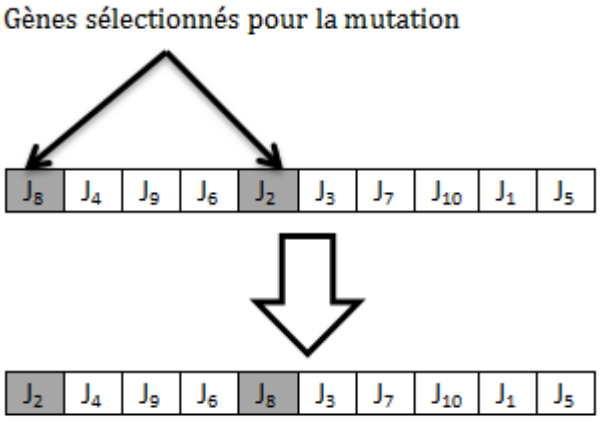


Figure 3.8. Exemple de l'opérateur de mutation utilisé

VI.1. Analyse de sensibilité de l'algorithme génétique

Le choix des paramètres dans une métaheuristique joue un rôle très important sur les résultats de la recherche et la qualité des solutions obtenues. En d'autres termes, un choix judicieux peut mener à des solutions de bonne qualité. Cependant, il n'existe pas une règle générale permettant de faire ce choix. En conséquence, une analyse de sensibilité peut aider à déterminer les meilleurs paramètres d'une métaheuristique pour un problème donné. Dans cette section nous présentons une analyse de sensibilité de l'algorithme génétique par rapport à la taille de population.

Pour pouvoir mener les simulations, un ensemble d'instances a été généré sur la base d'un système de production caractérisé par des classes de problèmes de tailles différentes. Nous considérons dans cette étude 3 classes de problèmes (10jobs/20 machines, 20 jobs/20 machines et 50 jobs/20 machines). Chaque classe de problème contient 5 exemples selon le temps de traitement qui est généré dans un intervalle [0,30] unités de temps. Les tableaux 3.1, 3.2 et 3.4 présentent les résultats de de l'analyse de sensibilité de l'algorithme génétique classique pour différentes tailles de population.

10x20	M=10%		M=20%	
	Taille POP	Cmax	Taille POP	Cmax
Exemple N°1	10	561	10	538
	20	544	20	546
	30	539	30	537
	40	**521**	40	529
	50	529	**50**	**524**
Exemple N°2	10	580	10	577
	20	574	20	574
	30	576	30	573
	40	**560**	40	566
	50	561	**50**	**553**
Exemple N°3	10	572	10	575
	20	547	20	547
	30	553	30	551
	40	**540**	**40**	**540**
	50	544	50	551
Exemple N°4	10	579	10	560
	20	550	20	538
	30	536	30	539
	40	**533**	**40**	**528**
	50	541	50	534
Exemple N°5	10	600	10	589

	20	592	20	586
	30	581	30	588
	40	**574**	40	579
	50	584	**50**	**579**

Tableau 3.1. Analyse de sensibilité de l'algorithme génétique en fonction de la taille de population pour 10 jobs / 20 machines

20x20	M=10%		M=20%	
	Taille POP	Cmax	Taille POP	Cmax
Exemple N°1	10	855	10	879
	20	840	20	849
	30	847	30	850
	40	**831**	40	844
	50	831	**50**	**836**
Exemple N°2	10	805	10	812
	20	813	20	819
	30	818	30	803
	40	**782**	40	795
	50	801	**50**	**794**
Exemple N°3	10	811	10	819
	20	806	20	788
	30	791	30	787
	40	**786**	40	802
	50	796	**50**	**779**
Exemple N°4	10	850	10	834
	20	855	20	838
	30	824	30	823
	40	**810**	**40**	**803**
	50	836	50	813
Exemple N°5	10	834	10	838
	20	818	20	805
	30	805	30	811
	40	**795**	40	815
	50	812	**50**	**799**

Tableau 3.2. Analyse de sensibilité de l'algorithme génétique en fonction de la taille de population pour 20 jobs / 20 machines

50x20	M=10%		M=20%	
	Taille POP	Cmax	Taille POP	Cmax
Exemple N°1	10	1444	10	1409
	20	1436	20	1435
	30	1426	30	1423
	40	**1404**	40	1419
	50	1420	**50**	**1406**
Exemple N°2	10	1485	10	1454
	20	1439	20	1454
	30	1454	30	1426

	40	**1426**	40	1423
	50	1443	**50**	**1417**
Exemple N°3	10	1447	10	1426
	20	1423	20	1429
	30	1439	30	1423
	40	**1404**	**40**	**1392**
	50	1410	50	1408
Exemple N°4	10	1499	10	1507
	20	1463	20	1494
	30	1452	30	1492
	40	**1442**	**40**	**1426**
	50	1475	50	1463
Exemple N°5	10	1431	10	1431
	20	1412	20	1419
	30	1416	30	1416
	40	**1400**	40	1407
	50	1414	**50**	**1400**

Tableau 3.3. Analyse de sensibilité de l'algorithme génétique en fonction de la taille de population pour 50 jobs/20 machines

Les résultats de simulation donnés dans les tableaux 3.1, 3.2 et 3.3 représentent les valeurs de C_{max} pour différents exemples de classes de problème considérés pour des taux de mutation de 10% et 20%.

On voit clairement que plus la taille de population augmente les résultats sont meilleurs. Ceci peut être expliqué par le fait que plus le nombre de solutions examinées augmente on a tendance à retrouver le plus meilleures solutions, vu qu'on aura une possibilité d'examiner plus de solutions et donc de couvrir plus de zones dans l'espace de recherche. Cependant, pour une taille de population supérieure à 50 on risque d'avoir une saturation et donc on n'aura pas toujours des solutions de bonne qualité. Dans la majorité des cas (les différents exemples traités) les meilleures solutions sont obtenues pour une taille de population égale à 40.

VII. Adaptation de l'algorithme génétique modèle des îles

L'idée principale de l'algorithme génétique modèle des îles est de créer un certain nombre de sous-populations différentes avec la même taille nommées « îles ». Chaque île contient sa population locale générée initialement de manière aléatoire (en décomposant la population initiale en sous-populations).

Le fait d'avoir plusieurs sous-populations préserve une certaine diversité dans la technique de recherche, étant donné que chaque île peut mener sa propre recherche.

Bien que chaque procédure de recherche (les opérateurs d'algorithmes génétiques) pour chaque île s'exécute principalement indépendamment des autres procédures des autres îles, l'île peut occasionnellement envoyer des individus aux autres îles (la migration).

VII.1. Analyse de sensibilité de l'algorithme génétique parallèle modèle des îles

Dans cette section nous présentons l'analyse de sensibilité par rapport au nombre de recherches parallèles (RP) de l'algorithme. En se basant sur les résultats de l'étude de sensibilité de l'algorithme génétique présentée dans la section précédente, la taille de population est fixée à 40 ensuite à 50. Les tableaux 3.4, 3.5 et 3.6 montrent les résultats de simulation pour 3 classes de problèmes de (10 jobs/ 20 machines, 20 jobs/ 20 machines et 50 jobs/ 20 machines) contenant 5 exemples chacune.

10x20	Taille POP=40		Taille POP=50	
	RP	Cmax	RP	Cmax
Exemple N°1	2	516	2	524
	4	510	4	522
	6	505	6	519
	8	**501**	**8**	**513**
	10	506	10	517
Exemple N°2	2	569	2	561
	4	551	4	549
	6	554	6	555
	8	**549**	**8**	**546**
	10	550	10	550
Exemple N°3	2	534	2	541
	4	535	4	542
	6	539	6	530
	8	**528**	**8**	**526**
	10	529	10	531
Exemple N°4	2	540	2	542
	4	528	4	531
	6	534	6	529
	8	**523**	**8**	**526**
	10	539	10	529
Exemple N°5	2	576	2	572
	4	569	4	569
	6	568	6	566
	8	**562**	**8**	**560**
	10	576	10	569

Tableau 3.4. Analyse de sensibilité de l'algorithme génétique modèle des îles par rapport à la taille de population pour 10 jobs/20 machines

20x20	Taille POP=40		Taille POP=50	
	RP	Cmax	RP	Cmax
Exemple N°1	2	828	2	841
	4	839	4	825
	6	819	6	829
	8	**816**	**8**	**823**
	10	833	10	827
Exemple N°2	2	793	2	798
	4	781	4	785
	6	777	6	776
	8	**774**	**8**	**770**
	10	788	10	775
Exemple N°3	2	784	2	784
	4	780	4	782
	6	782	6	784
	8	**773**	**8**	**776**
	10	775	10	777
Exemple N°4	2	819	2	805
	4	810	4	810
	6	806	6	806
	8	**801**	**8**	**793**
	10	808	10	802
Exemple N°5	2	805	2	801
	4	789	4	801
	6	788	6	802
	8	**787**	**8**	**794**
	10	797	10	796

Tableau 3.5. Analyse de sensibilité de l'algorithme génétique modèle des îles en fonction de la taille de population pour 20 jobs/20 machines

50x20	Taille POP=40		Taille POP=50	
	RP	Cmax	RP	Cmax
Exemple N°1	2	1386	2	1426
	4	1411	4	1409
	6	1413	6	1402
	8	**1385**	**8**	**1392**
	10	1400	10	1396
Exemple N°2	2	1421	2	1426
	4	1422	4	1397
	6	1413	6	1407
	8	**1383**	**8**	**1405**
	10	1404	10	1411
Exemple N°3	2	1408	2	1406
	4	1397	4	1395
	6	1387	6	1374
	8	**1371**	**8**	**1373**
	10	1375	10	1386

Exemple N°4	2	1453	2	1450
	4	1451	4	1465
	6	1448	6	1451
	8	**1436**	**8**	**1442**
	10	1451	10	1448
Exemple N°5	2	1420	2	1413
	4	1398	4	1414
	6	1391	6	1400
	8	**1383**	8	1371
	10	1388	**10**	**1368**

Tableau 3.6. Analyse de sensibilité de l'algorithme génétique modèle des îles en fonction de la taille de population pour 50 jobs/20 machines

Les résultats des simulations donnés par les tableaux 3.4, 3.5 et 3.6 montrent que l'augmentation de nombre de nombre de recherches parallèles (RP) entraine une amélioration dans la qualité des solutions obtenues et que les meilleurs résultats sont obtenues pour un nombre de recherches parallèles égale à 8. Dans le cas d'un nombre de recherche parallèle plus élevé la qualité des solutions diminue.

VIII. Résultats de simulation de l'algorithme IMGAMM

Prenons l'exemple suivant : étant donné que le nombre d'individus dans chaque île est 6 et qu'on fixe le paramètre de gestion de migration à 3. Un individu sélectionné d'une autre île ne peut être inséré dans cette île que si les trois premiers gènes de son chromosome sont complètement différents des trois premiers gènes de chaque individu de l'île dans laquelle il est supposé inséré.

La figure 3.11 illustre un exemple dans lequel l'individu ne vérifie pas la contrainte de migration qui n'est autre que la dissemblance aux différents individus de l'île.

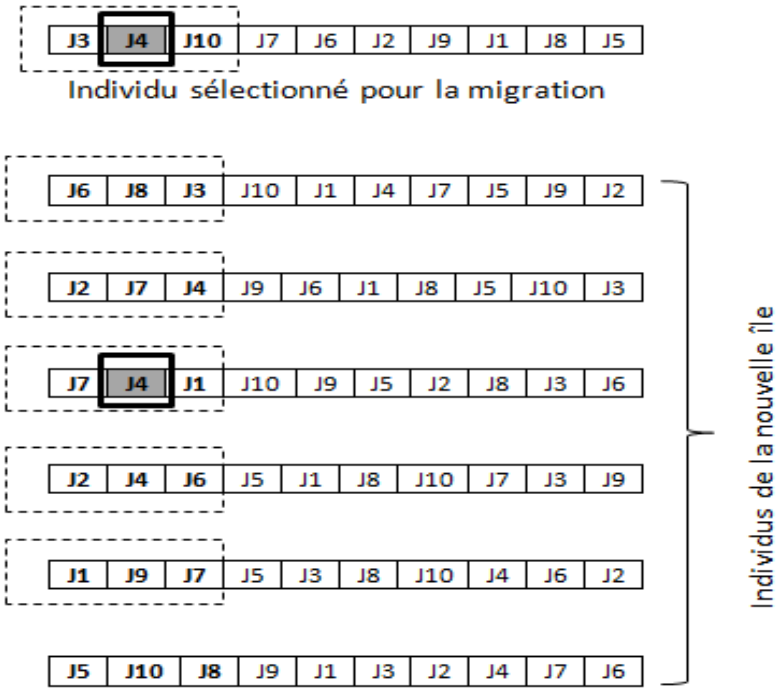


Figure 3.10 : Exemple de vérification de la condition de migration (individu non accepté)

VIII.1. Analyse de sensibilité de l'algorithme génétique parallèle modèle des îles avec gestion de migration en fonction de Δ

Les résultats de calcul donnés dans le tableau 4.7 montrent les valeurs obtenues du Makespan pour les différents exemples considérés. Les simulations sont faites pour une taille de population égale à 160 individus subdivisés en 8 îles (Sous-populations), chaque île comprend alors 20 individus. Le taux de migration est fixé à 0,5 ; cela signifie que la moitié de la population des îles est censée avoir migré. Nous avons fait l'analyse de sensibilité pour un système caractérisé par 20jobs et 20 machines. Un nombre de jobs égale à 20 signifie que la taille de chromosome est 20, c'est-à-dire chaque individu est représenté par un chromosome qui contient 20 gènes (un individu représente ici une séquence de 20 jobs). Le paramètre de gestion de migration Δ peut donc varier de 0 à 20. Nous avons incrémenté la valeur de Δ d'un pas de 2. Les résultats de simulation sont donnés par le tableau 3.7.

	Cmax				
Paramètre de gestion de migration Δ	**Example 1**	**Example 2**	**Example 3**	**Example 4**	**Example 5**
2	767	794	789	790	782
4	760	789	787	790	783
6	760	785	784	782	778
8	856	779	781	782	777
10	**729**	**776**	**759**	**779**	**746**
12	760	780	770	780	772
14	764	781	780	788	775
16	770	782	786	788	776
18	775	793	793	790	808

Tableau 3.7. Analyse de sensibilité de l'algorithme génétique modèle des îles avec gestion de migration en fonction du paramètre de gestion de migration Δ

Les résultats de simulation montrent que les meilleures valeurs de makespan obtenues par l'IMGAMM sont données pour un paramètre de gestion de la migration égal à 10. C'est-à-dire la moitié du chromosome de l'individu candidat pour la migration doit être totalement différente de tous les individus dans l'île réceptive. Cela peut s'expliquer par le fait que plus les solutions ne sont pas similaires plus la diversité dans l'espace de recherche est garantie. Cependant, si la dissimilitude est très grande (Δ supérieur à 10), contrairement aux attentes, les solutions ne sont pas de meilleures qualité (forte diversification). De ce fait, une dissimilitude moyenne entre les nouveaux individus (à migrer) et les individus de l'île réceptive entraine un bon équilibre entre l'intensification et la diversification.

VIII.2. Comparaison des résultats entre l'algorithme génétique, l'algorithme génétique parallèle modèle des îles, et l'IMGAMM

Les résultats de simulation sont donnés pour différentes classes d'instances du problème d'ordonnancement de Flow Shop. Pour les deux algorithmes, le nombre d'îles est fixé à 8, le nombre d'individus dans chaque île est de 20 et le taux de migration est égal à 0,5.

Pour l'IMGAMM, le paramètre de gestion de la migration Δ est fixé à 10 lorsque le nombre de jobs est égal à 20 et est fixé à 25 lorsque le nombre de jobs égal à 50. Cela signifie que l'individu à insérer dans une île donnée doit ont une similitude maximale

qui est de moitié par rapport à tout individu dans cette île. En d'autres termes, la distance maximale, dans ce cas, est égale à 50 %. Les résultats de calcul sont donnés dans le tableau 4.8 et 4.9.

	Solution optimale		Algorithme génétique		Modèle des îles		IMGAMM	
	10 jobs 10 machines	10 jobs 20 machines	10 jobs 10 machines	10 jobs 20 machines	10 jobs 10 machines	10 jobs 20 machines	10 jobs 10 machines	10 jobs 20 machines
Example N°1	338	539	360	565	354	551	342	540
Example N°2	336	549	366	578	350	560	338	552
Example N°3	360	529	381	555	373	546	361	532
Example N°4	312	528	343	555	333	544	315	532
Example N°5	342	541	382	579	364	557	345	543

Tableau 3.8. Comparaison de résultats entre l'algorithme génétique, le modèle des îles et le modèle des îles avec gestion de migration pour des faibles instances

	Algorithme génétique				Modèle des îles				IMGAMM			
	20 jobs 20 machines	20 jobs 30 machines	50 jobs 20 machines	50 jobs 30 machines	20 jobs 20 machines	20 jobs 30 machines	50 jobs 20 machines	50 jobs 30 machines	20 jobs 20 machines	20 jobs 30 machines	50 jobs 20 machines	50 jobs 30 machines
Example N°1	806	1055	1453	1728	744	1016	1397	1700	729	1001	1379	1673
Example N°2	813	1052	1482	1700	790	1030	1435	1671	776	1004	1421	1660
Example N°3	818	1006	1465	1723	783	979	1427	1674	759	958	1413	1654
Example N°4	810	1030	1423	1682	782	999	1398	1647	779	980	1383	1640
Example N°5	779	1066	1411	1731	759	1030	1363	1698	746	1017	1349	1680

Tableau 3.9. Comparaison de résultats entre l'algorithme génétique, le modèle des îles et le modèle des îles avec gestion de migration pour des grandes instances

Les tableaux 4.8 et 4.9 montrent une comparaison entre les meilleurs résultats obtenus par l'algorithme génétique du modèle d'île classique et ceux obtenus par l'IMGAMM pour les mêmes instances de problème et les mêmes paramètres d'algorithmes.

On voit bien que le makespan minimum est obtenu par l'IMGAMM proposé dans les différents exemples. Par conséquent, nous pouvons dire que, pour le problème

d'ordonnancement de Flow Shop considéré, l'algorithme IMGAMM est plus efficace et donne de meilleurs résultats.

V. Conclusion

Dans ce chapitre, nous avons présenté une revue sur l'algorithme génétique et sa variante algorithme génétique modèle des îles. Nous avons aussi présenté une nouvelle variante de l'algorithme génétique modèle des îles que nous avons développé. Cette variante que nous avons appelé « Algorithme génétique modèle des îles avec gestion de migration (IMGAMM) » a la particularité de gérer le processus de migration en se basant sur un paramètre qui mesure la dissimilitude. Le but est de maximiser la dispersion des individus dans l'espace de recherche de chaque île. Les résultats de la nouvelle variante proposée sont très prometteurs. En effet, une comparaison entre les trois algorithmes : génétique, modèle des îles et modèle des îles avec gestion de migration a montré une supériorité dans les performances de la variante développée. Le chapitre suivant sera consacré à l'algorithme des babouins qui constitue une deuxième contribution de cette thèse.

CHAPITRE 4

Algorithme des babouins

La reproduction chez les mammifères a inspiré l'introduction de la métaheuristique algorithme génétique ainsi que toutes ses variantes. Dans ce chapitre, nous présentons la contribution principale de ce travail, qui est aussi inspirée du processus de reproduction et de survie chez une espèce particulière des mammifères, il s'agit de l'espèce des singes « Babouins chacma». Nous allons dans ce qui suit d'abord présenter le phénomène naturel qui nous a inspiré pour le développement de la métaheuristique que nous avons baptisé « algorithme des Babouins ». La suite du chapitre sera consacrée à la définition : du principe, de l'adaptation, et des résultats de cette métaheuristique.

I. Introduction

Les métaheuristiques bio-inspirées ont connu et continuent à connaître un intérêt croissant pour la résolution des problèmes d'optimisation NP-Difficiles. Les algorithmes génétiques sont parmi les métaheuristiques bio-inspirées qui ont fait l'objet de plusieurs recherches visant à résoudre de nombreux problèmes. Cependant, nous avons remarqué que le mécanisme de reproduction et de survie n'est pas commun entre toutes les espèces. Certaines espèces présentent des mécanismes de reproduction qui peuvent se révéler intéressantes pour l'inspiration de nouveaux algorithmes.

Une étude récente [60] a discuté le phénomène d'infanticide causé par les mâles de la même espèce chez les mammifères et quelles sont les différentes techniques adoptées par les femelles pour protéger les enfants contre ce phénomène. L'étude a porté essentiellement sur l'espèce des singes Babouins chacma. Elle a montré que parmi 260 espèces des mammifères examinées 119 ont montré le phénomène d'infanticide causé par les mâles de la même communauté. Ce qui pousse les femelles à développer des stratégies pour la protection de leurs enfants. Une de ces stratégies est celle remarquée chez les babouins chacma qui consiste en l'accouplement d'une même femelle avec plusieurs males donnant ainsi l'impression à chacun d'entre eux qu'il soit le géniteur de l'enfant. C'est de cette publication que nous nous sommes inspiré pour le développement d'une nouvelle métaheuristique à qui nous donnons le nom de métaheuristique des babouins. L'idée de base est que, contrairement aux algorithmes

génétiques, il y'a deux genres d'individus (individu mâle et individu femelle) et que le croisement se fait entre un individu femelle et plusieurs individus mâles donnant ainsi lieu à un nombre d'enfants qui est le double de nombre des pères, deux des individus enfants seront sélectionnés pour être insérés dans la nouvelle génération. Ces individus sont : le meilleur qui va remplacer la mère (si sa fitness est meilleure que celle de la mère) et le deuxième meilleur qui va remplacer le plus mauvais des pères (aussi si sa fitness est meilleure que ce dernier).

Le principe de la métaheuristique des babouins sera présenté beaucoup plus en détails dans ce qui suit.

II. Les babouins

Les babouins sont de gros singes au gros museau nu et allongé, semblable à celui du chien, d'où leur ancien nom de *cynocéphales (*Le mot « cynocéphale » est emprunté au grec ancien qui signifie « qui a une tête de chien »). Ce nom « cynocéphale » a été remplacé par la suite par « babouin ».

Comme beaucoup de primates, les babouins vivent en bandes organisées. Ils ne se sentent en sécurité qu'à l'intérieur de leur bande, dominée par quelques mâles puissants qui en assurent la défense.

Les formes de communication sont variées : transmission de signaux au moyen de la queue, la posture, les cris et les jappements. Selon les espèces, les babouins pèsent de 14 à 40 kg et mesurent entre 50 et 115 cm. Les femelles sont deux fois plus petites que les mâles. L'espérance de vie avoisine les 20 ans, mais le double peut être atteint en captivité.

VII.1. Les babouins Chacma (Papio ursinus)

Le babouin chacma (Papio *Ursinus*), connu aussi sous le nom de **babouin** du **Cap**, est l'un des plus grands de tous les singes. Situé principalement en Afrique australe. Les babouins chacmas vivent une vingtaine d'années. La taille d'un babouin chacma adulte peut aller de 0,7 m et 1,1m et son poids allant de 15 jusqu'à 31kg. Cette espèce a une grande variété de comportements sociaux, notamment une hiérarchie de dominance, une recherche de nourriture collective, l'adoption de jeunes par des femelles et des couples d'amitié. En général, l'espèce n'est pas menacée, mais les contacts entre les humains et les babouins sont devenu de plus en plus présent en raison de la pression démographique. Par conséquent, la chasse et les accidents tuent ou retirent de

nombreux babouins de la nature, réduisant ainsi le nombre de babouins et perturbant leur structure sociale. Les babouins vivent en groupes hiérarchisés de 5 à 250 animaux (le plus souvent autour de 50 individus) dont la taille varie en fonction des circonstances, de l'espèce ou de l'époque de l'année.

Figure 4.1. Le babouin chacma

III. L'infanticide chez les animaux

Durant les années 70, l'infanticide animal a été observé pour la première fois, chez les langurs d'Inde, cette nouvelle a ébranlé les zoologues. Certains ont voulu attribuer le phénomène au « stress » subi par ces singes au contact des hommes. Jusqu'à ce que l'Américaine Sarah Hrdy [61] démontre qu'il s'agissait d'une stratégie liée à la théorie de l'évolution. Le mâle dominant contrôle la provenance des petits qu'il aura à sa charge en tuant les petits des autres mâles . En d'autres termes *« il maximise ainsi ses chances d'assurer une descendance ».*

En 2014, dans [60] une étude intéressante a porté sur l'observation détaillée de plus de 260 espèces de mammifères depuis plus d'un demi-siècle. L'objectif étant de déterminer les espèces caractérisées par l'infanticide, et ainsi d'analyser les causes et décrire les stratégies mises en place par les femelles pour préserver leurs progénitures.

Sachant que dans cette étude, le comportement n'est considéré que lorsqu'il est stratégique, à savoir, lorsqu'il offre des avantages aux mâles. Les résultats accablants de

l'étude ont montré que le meurtre infantile est présent chez 119 espèces (presque la moitié) y compris les gorilles, les chimpanzés, les babouins, les lions, les ours, les hippopotames, les souris ainsi que les écureuils. En effet, les mâles tuent les petits des autres, mais aucun meurtre de ses propres petits n'ayant jamais été constaté chez les mammifères (à l'exception notable de l'homme).

L'étude était plus détaillée et a donné un intérêt remarquable à l'espèce des primates des babouins chacma de l'Okavango (Botswana), car elle est très facile à observer et est très visibles en Afrique australe.

Les babouins chacma vivent en groupes sous la domination d'un mâle qui doit souvent remettre son titre en jeu. Le vainqueur de chaque combat peut être battu quelques jours ou quelques semaines plus tard, de ce fait il doit s'accoupler au plus vite. Par conséquent il doit tuer les petits qui ne sont pas les siens pour rendre leurs mères de nouveaux fécondes.

Chez cette espèce les prédateurs tels que les léopards et les guépards ne représentent pas la menace la plus mortelle pour les petits babouins Chacma. En effet, ce sont les mâles adultes de leur propre espèce qui constituent la plus grande menace. Ce comportement (l'infanticide chez cette espèce) est la cause de 70% de la mortalité infantile.

Dans la même étude, une stratégie très intéressante adaptées par les femelles, qu'il s'agisse d'une femelle souris ou d'un hippopotame « *Elles multiplient les partenaires ce qui fait que les mâles, dans le doute, s'abstiennent de tuer un petit qui pourrait* ***être le leur »***. En d'autres termes, les femelles utilisent une stratégique pour empêcher les mâles de tuer leurs bébés. En s'accouplant avec autant de mâles que possible sur une courte période de temps, cela rend difficile de discerner la paternité des petits.

IV. Algorithme des babouins

Nous nous sommes inspirés de la stratégie adaptée par les femelles pour l'élaboration d'une nouvelle métaheuristique. Une des métaheuristiques les plus classiques est l'algorithme génétique. Dans un algorithme génétique classique un enfant est obtenu en croisant les chromosomes de deux parents. Dans la nouvelle métaheuristique proposée que nous appelons « Algorithme des babouins », un enfant est obtenu en croisant le chromosome d'une mère avec plusieurs pères (ici, par rapport aux

algorithmes génétiques, on distingue le genre de la solution). En effet, le croisement consiste à prendre des gènes du chromosome de la mère ainsi que des gènes de différents parents, on obtient ainsi plusieurs enfants (le nombre d'enfants est deux fois le nombre de pères). A l'issu de cette opération, on n'en garde que deux enfants, les deux qui ont les meilleurs fitness. L'enfant ayant la meilleure fitness va remplacer la mère si jamais sa fitness est meilleure que celle de la mère. Le deuxième va remplacer le plus mauvais des pères aussi si sa fitness est meilleure à celle de ce dernier.

L'algorithme 4.1 ci-après présente les détails de la métaheuristique des babouins.

Algorithme 4.1 : Algorithme des babouins

(1) **Initialisation** fixer la taille de population $Taille_{pop}$
fixer le nombre de père candidats à la reproduction (nbr_{pere})
(2) **Générer** une population initiale de taille $Taille_{popmere}$
(3) **Evaluer** chaque individu dans la population mère en calculant la valeur de fonction objectif (la fitness) (C_{max})
(4) **Générer** une population initiale de taille $Taille_{poppere}$
(5) **Evaluer** chaque individu dans la population père en calculant la valeur de fonction objectif (la fitness) (C_{max})
(6) **Tant que** le critère d'arrêt n'est pas atteint
Pour chaque mère
Sélectionner les pères candidats à la reproduction
Appliquer l'opérateur de croisement
Appliquer l'opérateur de mutation
Evaluer tous les enfants
Classer les enfants
Classer les pères
Choisir le meilleur enfant
Si f(enfant)<f(mère)
Remplacer mère par enfant
Fin si
Choisir le deuxième meilleur enfant
Si f(enfant)<f(père)
Remplacer père par enfant
Fin si
Fin pour
Fin tant que
(7) **Faire** sortir meilleure solution

IIV.1. Analyse de sensibilité de l'algorithme

La taille de la population et le nombre de pères candidats pour la reproduction sont les paramètres qui jouent un rôle primordial dans l'efficacité de l'algorithme en termes d'équilibre entre l'intensification et la diversification de l'algorithme. En effet, le nombre de solutions générées initialement et examinées au cours des itérations a un impact sur la qualité des solutions obtenues.

De ce fait, nous avons fait une analyse de sensibilité relative à la population initiale de l'algorithme des babouins, ainsi que le nombre de pères candidats pour la reproduction.

IIV.1.1. Analyse de sensibilité de l'algorithme par rapport à la taille de population initiale

Rappelons que l'ensemble de nombre de pères et de mères générés initialement représente la taille de la population de l'algorithme, nous présentons dans cette section une étude de sensibilité en variant le nombre de solutions générées initialement (la taille de population).

Nous avons considéré un système de 20 jobs/ 20 machines, contenant 5 exemples avec des temps de traitement généré aléatoirement dans un intervalle de temps de [1 30] unités de temps. Cette étude a été menée pour un nombre de pères candidats pour la reproduction égale à 4. La taille de population des mères et la taille de population des pères est calculée à l'aide des équations 4.1 et 4.2 respectivement.

Taille pop mère=4*Taille de pop /5 4.1

Taille de pop père= Taille de pop /5 4.2

Les résultats de simulations sont donnés par le tableau 4.1.

	C_{max}				
Taille pop	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5
30	720	743	740	752	720
40	719	739	739	740	715
50	717	736	735	738	719
60	709	737	732	733	710
70	**706**	**733**	**729**	**728**	**704**
80	711	739	731	731	707
90	716	741	735	733	708
100	717	741	736	733	711

Tableau 4.1. Analyse de sensibilité de l'algorithme des babouins par rapport à la taille de population

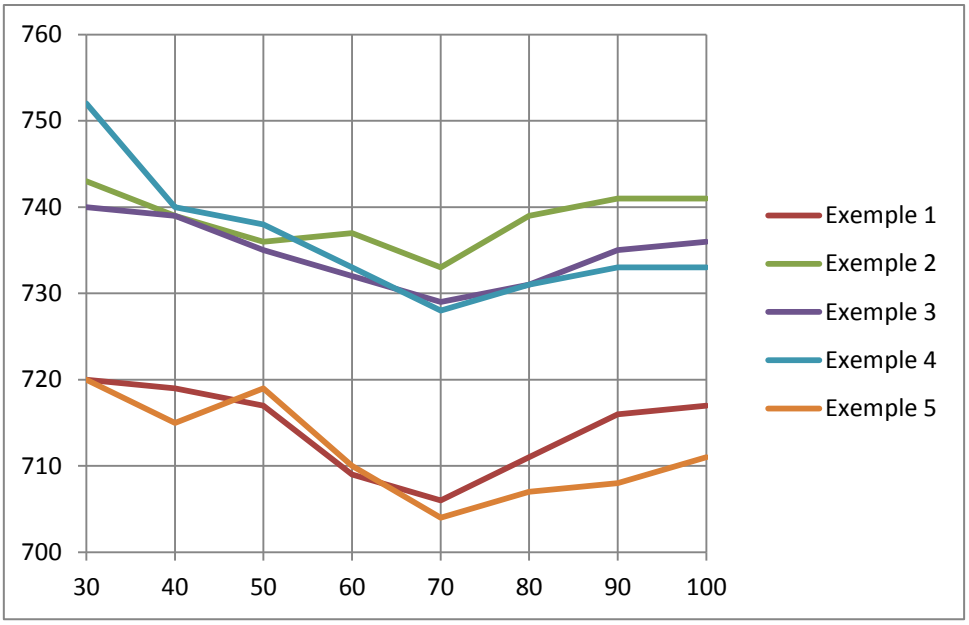


Figure 4.2. Analyse de sensibilité de l'algorithme des babouins par rapport à la taille de population

Le tableau 4.1 et la figure 4.2 montrent l'analyse de sensibilité de la taille de population sur les résultats obtenus. Nous remarquons que plus la taille de la population augmente le C_{max} diminue mais au-delà d'une taille de population de 70 le C_{max} augmente. Il est clairement vu que les meilleurs résultats sont donnés pour une taille de population égale à 70.

IIV.1.2. Analyse de sensibilité de l'algorithme par rapport au nombre de pères

Nous définissons ici le nombre de pères NPR comme le nombre de père candidats à la reproduction. Il représente le nombre de possibilité de combinaisons possibles entre chaque mère et les individus de la population des pères. En d'autres termes, un nombre de pères candidats à la reproduction égale à 4 signifie que le croisement a lieu entre chaque individu de la population des mères et 4 individus choisis aléatoirement de la population des pères.

Nous avons effectué une étude de sensibilité dans laquelle nous avons fixé la taille de population à 70 (relativement aux résultats du tableau 4.1) et en incrémentant le nombre de pères candidats à la reproduction (noté NPR). Les résultats de simulation sont donné dans le tableau 4 .2.

	C_{max}				
NPR	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5
1	726	760	759	759	739
2	724	740	743	741	718
3	717	736	735	735	711
4	706	733	729	728	704
5	704	729	724	726	702
6	**703**	**725**	**722**	**722**	**700**
7	707	728	726	730	703
8	711	734	730	731	707
9	711	736	736	737	711
10	712	738	740	742	718

Tableau 4.2. Analyse de sensibilité de l'algorithme des babouins par rapport au nombre de pères candidats à la reproduction

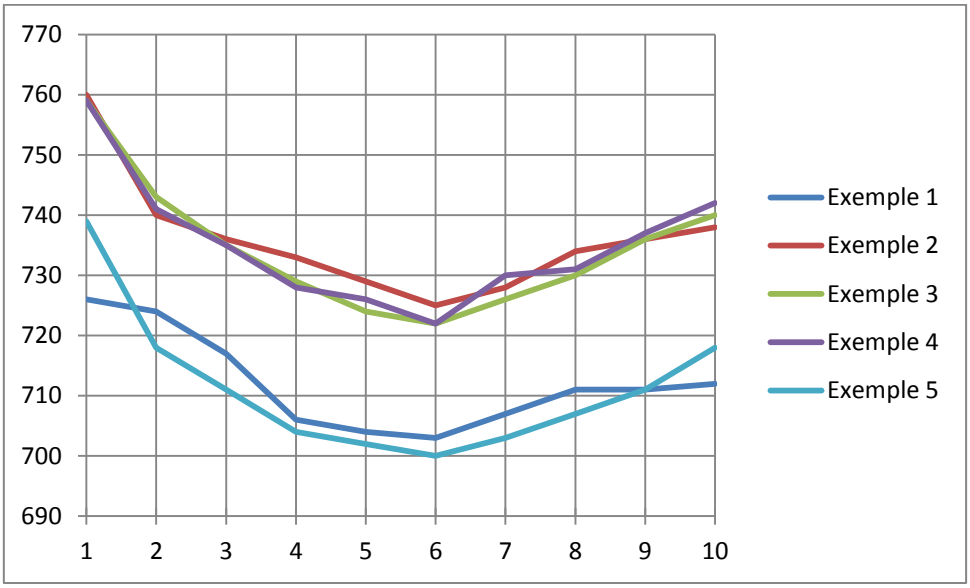


Figure 4.3. Analyse de sensibilité de l'algorithme des babouins par rapport au nombre de pères candidats à la reproduction

Les résultats de simulation du tableau 4.2 et de la figure 4.3 montrent que plus le nombre de pères candidats à la reproduction augmente le C_{max} diminue. Mais pour des valeurs de nombre de pères supérieur à 6 les valeurs de C_{max} augmentent. Les meilleurs résultats sont obtenus pour un nombre de pères égale à 6.

IIV.1.3. Résultats de simulation pour les différentes instances

Dans cette section nous présentons les résultats de simulations de l'algorithme des Babouins en de base tout en respectant les résultats de l'étude de sensibilité (taille de population égale à 70 et nombre de pères candidats à la reproduction égale à 6)

comparés à ceux obtenus par l'algorithme IMGAMM présenté dans le chapitre 3. Nous avons considéré les mêmes instances et les mêmes exemples que le chapitre 3.

Le tableau 4.3 montre les résultats de simulation de l'algorithme des babouins pour différentes classes de problèmes comparés à ceux de l'algorithme IMGAMM.

	20 jobs 20 machines		20 jobs 30 machines		50 jobs 20 machines		50 jobs 30 machines	
	Algorithme des Babouins	IMGAMM	Algorithme des Babouins	IMGAMM	Algorithme des Babouins	IMGAMM	Algorithme des Babouins	IMGAMM
Exemple N°1	703	729	943	1001	1249	1379	1564	1673
Exemple N°2	725	776	979	1004	1295	1421	1539	1660
Exemple N°3	722	759	910	958	1276	1413	1526	1654
Exemple N°4	722	779	928	980	1262	1383	1505	1640
Exemple N°5	700	746	957	1017	1254	1349	1583	1680

Tableau 4.3. Comparaison de résultats entre l'algorithme des babouins et l'algorithme génétique modèle des îles avec gestion de migration.

Nous constatons à partir des résultats de simulation du tableau 4.3 que l'algorithme des babouins montre son efficacité par rapport à l'algorithme génétique modèle des îles avec gestion de migration pour les différentes instances.

Pour bien examiner l'efficacité de l'algorithme des Babouins, nous l'avons utilisé pour la résolution du problème considéré pour des faibles instances auxquels nous avons déjà calculé la valeur optimale de C_{max} (par énumération). Les résultats obtenus sont donnés par le tableau 4.4.

	Solution optimale		Algorithme des Babouins	
	10 jobs 10 machines	10 jobs 20 machines	10 jobs 10 machines	10 jobs 20 machines
Exemple N°1	**338**	**539**	**338**	**539**
Exemple N°2	336	549	336	549
Exemple N°3	**360**	**529**	**360**	**529**
Exemple N°4	312	528	318	**528**
Exemple N°5	**342**	541	**342**	542

Tableau 4.4. Comparaison de résultats entre l'algorithme des babouins et l'algorithme génétique modèle des îles avec gestion de migration et la solution optimale pour les faibles instances

Le tableau 4.4 montre les résultats de simulation pour des systèmes composés de 10 jobs / 10 machines et 10 jobs / 20 machines. Les résultats montrent que les solutions obtenues sont proches de la solution optimale, et donne même la solution optimale dans la majorité des cas. Ces résultats montrent l'efficacité de l'algorithme des Babouins pour la résolution du problème d'ordonnancement dans un atelier Flow Shop.

V. Conclusion

Dans ce chapitre nous avons présenté une nouvelle métaheuristique que nous avons développée en se basant sur le processus de reproduction chez les babouins.

L'analyse de sensibilité de cette métaheuristique a été étudiée et présentée dans ce chapitre, en étudiant l'influence de variation de ses paramètres internes sur ses performances en termes de qualité des solutions obtenues.

Comme nous l'avons montré dans le chapitre précédent, les résultats de l'algorithme IMGAMM présente les meilleurs résultats par rapport aux deux algorithmes génétique et génétique modèle des îles. Dans ce chapitre, la nouvelle métaheuristique proposée à savoir, métaheuristique des Babouins, a été appliquée aux mêmes instances que ceux utilisées en chapitre 3 et comparée aux résultats de l'IMGAMM. Les résultats obtenus ont démontré que l'algorithme des Babouins donne nettement des résultats meilleurs. En effet, à plusieurs reprises cet algorithme a donné un résultat optimal.

Conclusion générale

Les métaheuristiques bio-inspirées ont connu et continuent à connaître un intérêt croissant pour la résolution des problèmes d'optimisation combinatoire classés NP-Difficiles. Les algorithmes génétiques sont parmi les métaheuristiques bio-inspirées qui ont fait l'objet de plusieurs recherches visant à résoudre de nombreux problèmes d'optimisation combinatoires. Ce sont des algorithmes basés sur la reproduction chez les espèces. Depuis leur apparition plusieurs variantes ont été proposées ayant pour objectif l'amélioration de leur performance. Les algorithmes génétiques parallèles sont parmi les variantes les plus prometteuses des algorithmes génétiques en termes de qualité des solutions obtenues. D'autre part, on remarque que le mécanisme de reproduction et de survie n'est pas commun entre toutes les espèces. Certaines espèces présentent des mécanismes de reproduction qui peuvent révéler intéressantes pour l'inspiration de nouvelles algorithmes bio-inspirés.

Dans cette thèse, nous présentons deux nouvelles métaheuristiques basées sur la reproduction chez les espèces. La premières est une amélioration de l'algorithme génétique parallèle sous sa forme modèle des îles, en introduisant un paramètre de gestion de migration. L'idée étant basée sur le contrôle de diversité dans l'espace de recherche. Durant notre étude sur les algorithmes génétiques nous nous sommes intéressés aux mécanismes de reproduction et de survie de plusieurs espèces. Nous avons constaté que l'infanticide est un phénomène présent dans plusieurs communautés d'animaux pour diverses raisons, et que les mères dans ces communautés adoptent certaines stratégies pour la protection de sa progéniture. Nous nous sommes inspirés de la stratégie utilisée par les mères chez l'espèce des singes babouins pour le développement d'une nouvelle métaheuristique dite « l'algorithme des babouins ».

Les algorithmes développés ont été utilisés pour la résolution du problème d'ordonnancement dans un atelier Flow Shop ayant pour objectif la minimisation du makespan. Des études de sensibilité ont été menées pour pouvoir établir un choix judicieux des paramètres des deux algorithmes. Les résultats de ces simulations ont mené aux conclusions suivantes :

L'algorithme génétique parallèle modèle des îles avec gestion de migration a donné de meilleurs résultats en termes de qualité des solutions obtenues en les

comparants avec l'algorithme génétique de classique et avec l'algorithme génétique modèle des îles.

L'algorithme des babouins développé a donné des meilleurs résultats par rapport à l'algorithme génétique classique, à celui du modèle des îles et à celui du modèle des îles avec gestion de migration.

Les perspectives :

Plusieurs perspectives sont ouvertes pour ce travail :

Au niveau de l'algorithme génétique modèle des îles avec gestion de migration : hybrider l'algorithme avec une procédure de recherche locale au niveau des îles qui permet de couvrir un grand ensemble de l'espace de recherche.

Au niveau de l'algorithme des babouins : vue la nouveauté de l'algorithme, il peut être appliquer pour la résolution des problèmes d'optimisation combinatoires connus tels que le problème de voyageur de commerce et le problème de tournée de véhicules.

Références bibliographiques

[1] Sassine C., "Intégration des politiques de maintenance dans les systèmes de production manufacturiers", Thèse de doctorat soutenue à l'INP de Grenoble (France), 1998.

[2] Pinedo. M. L., "Scheduling: Theory, Algorithms and systems", London, Springer, Scheduling, 2010.

[3] Parunak, H.V.D., "Manufacturing experience with the contract net. In Proceedings of the Fith Workshop on Distributed Artificial Intelligence", 1985.

[4] Farhoodi F., "A knowledge-based approach to dynamic job-shop scheduling", International Journal of Computer Integrated Manufacturing, 3 (2), 1990, pp. 84-95.

[5] Grabot B., "Objective satisfaction assessment using neural nets for balancing multiple objectives", International Journal of Production Research, 36, (9), 1998, pp. 2377-2395.

[6] Groupe d'Ordonnancement Théorique et Appliqué, "Les problèmes d'ordonnancement", R.A.I.R.O. Recherche opérationnelle/Operational Research, 27(1), 1993, pp. 77-150.

[7] Bellman R., "The theory of dynamic Programming", Bulletin of the American Mathematical Society, 60, 1954, pp. 503-515.

[8] Gondran M., Minoux M., "Graphs and Algorithms", NewYork, John Wiley, Series in discrete Mathematics Optimization, 1984.

[9] Kirkpatrick, S., Gellat, C.D., et Vecchi, M.P., "Optimisation by Simulated Annealing. Science", 220, 1983, pp. 671-680.

[10] Metropolis N., Rosenbluth A., Rosenbluth M. & Teller, E., "Equation of state calculations by fast computing machines", Journal of chemical physics, 21, 1953, pp. 1087-1092.

[11] Glover, F., "Future paths of integer programming and links to artificial intelligence", Computers and Operations Research, 13, 1986, pp. 533-549.

[12] Darwin C., "On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, Londres, John Murray, 1859.

[13] Holland. J., "Adaptive in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, 1975.

[14] Goldberg, D.E., "Genetic algorithms in search, Optimization and machine learning", Addison-Wesley publishing company, 1989.

[15] Dorigo, M., Maniezzo, V., & Colorni, A., "The Ant System: Optimization by a colony of cooperating agents". IEEE Transactions on systems, Man, and Cybernetics-Part B, 26(1), 1996, pp.1-13.

[16] Eberhart, R.C. & Kennedy, J., "New optimizer using particle swarm theory", Proceedings of the 6th International Symposiumon Micro Machine and Human Science, 1995, pp.39-43, Nagoya, Japan.

[17] Glover F., "Heuristics for Integer Programming Using Surrogate Constraints", Decision Sciences, 8, 1977, pp. 156-166.

[18] Geem Z. W., Kim J. H., Loganathan G. V., "A new heuristic optimization algorithm: harmony search", Simulation, 76(2), 2001, pp. 60-68.

[19] Kim H., Ahn B., "A new evolutionary algorithm based on sheep flocks heredity model", In Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and signal Processing, PACRIM, 2, 2001, pp. 514-517.

[20] Li X. L., Shao Z. J., Qian J. X., "An optimizing method based on autonomous animals: Fish-swarm Algorithm", System Engineering Theory and Practice, vol. 22(11), 2002, pp.32-38.

[21] Jung S.H., "Queen-bee evolution for genetic algorithms", Electronics letters, 39(6), 2003, pp. 575-576.

[22] Eusuff M. & Lansey K.E., "Optimization of water distribution network design using the shuffled frog leaping algorithm", Journal of Water Resources Planning and Management, 129(3), 2003, pp. 210–225.

[23] Pinto P., Runkler T. A., Sousa J. M., "Wasp swarm optimization of logistic systems", Adaptive and Natural Computing Algorithms, 2005, pp. 264-267.

[24] Mucherino A. & Seref O., "Monkey search: a novel metaheuristic search for global optimization", Data Mining, Systems Analysis and Optimization in Biomedicine, 953(1), 2007.

[25] Comellas F. & Martinez N. J., "Bumblebees: a multiagent combinatorial optimization algorithm inspired by social insect behaviour", In Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, Shanghai, China, 2009, pp. 811-814.

[26] Yang X. S. & Deb S., "Cuckoo Search via Lévy flights", In proceedings of World Congress on Nature & Biologically Inspired Computing, Coimbatore, India, 2009, pp. 210-214.

[27] Yang X., "Firefly algorithms for multimodal optimization." Stochastic algorithms: foundations and applications. Springer Berlin Heidelberg, 2009, pp. 169-178.

[28] Oftadeh R. & Mahjoob M. J., "A new meta-heuristic optimization algorithm: Hunting Search", In proceeding of the Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009.

[29] Xin-She Yang, "A new metaheuristic bat-inspired algorithm", In Proceedings of the Fourth International Workshop on Nature inspired cooperative strategies for optimization (NICSO 2010), Berlin, Heidelberg, 2010, pp.65-74.

[30] Hedayatzadeh R., Salmassi F. A., Akbari R., Ziarati K., "Termite colony optimization: A novel approach for optimizing continuous problems", In the proceedings of 18th IEEE Iranian Conference on Electrical Engineering, 2010, pp. 553-558.

[31] Shayeghi H. & Dadashpour J., "Anarchic Society Optimization Based PID Control of an Automatic Voltage Regulator (AVR) System", Electrical and Electronic Engineering, 2(4), 2012, pp. 199-207.

[32] Niu B., Wang H., "Bacterial Colony Optimization", Discrete Dynamics in Nature and Society, 2012.

[33] Tang R., Fong S., Yang X. S. & Deb S., "Wolf search algorithm with ephemeral memory". In proceedings of Seventh International Conference on Digital Information Management, 2012, pp. 165–172.

[34] Subramanian C., Sekar A. S. S. & Subramanian K., "A New Engineering Optimization Method: African Wild Dog Algorithm", International Journal of Soft Computing, 8(3), 2013, pp. 163-170.

[35] Gheraibia Y., Moussaoui A., "Penguins search optimization algorithm (PeSOA)", In proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2013, pp. 222-231.

[36] Abedinia O., Amjady N. & Ghasemi A., "A new metaheuristic algorithm based on shark smell optimization", Complexity, 2014.

[37] Bansal J. C., Sharma H., Jadon S. S., Clerc M., "Spider monkey optimization algorithm for numerical optimization", Memetic Computing, 6(1), 2014, pp. 31-47.

[38] Mirjalili S., "The ant lion optimizer", Advances in Engineering Software, 83, 2015, pp. 80-98.

[39] Deb S., Fong S. & Tian Z., "Elephant Search Algorithm for optimization problems", In Proc. of the 10th IEEE International Conference on Digital Information Management (ICDIM), 2015, pp. 249-255.

[40] Chen C., Tsai Y., Liu I., Lai C., Yeh Y., Kuo S. & Chou Y., "A Novel Metaheuristic: Jaguar Algorithm with Learning Behavior." In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2015, pp. 1595-1600.

[41] Odili J. B. & Kahar M. N. M., "Solving the Traveling Salesman's Problem Using the African Buffalo Optimization",. Computational intelligence and neuroscience, 2016.

[42] Ibrahim M. K. & Ali R. S., "Novel Optimization Algorithm Inspired by Camel Traveling Behavior", Iraq J. Electrical and Electronic Engineering, 12(2), 2016, 167-178.

[43] Yazdani M. & Jolai F., "Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm", Journal of Computational Design and Engineering, 3(1), 2016, pp. 24-36.

[44] Kaveh A. & Bakhshpoori T., "Water Evaporation Optimization: A novel physically inspired optimization algorithm", Computers & Structures, 167, 2016, pp. 69-85.

[45] Darwin, D., "The Origin of Species by Means of Natural Selection or the Preservation of Favored Races in the Struggle for Life", The Book League of America, Originally published in 1859, (1929).

[46] Coello C., "An empirical studyof evolutionary techniques for multiobjective optimization in engineering design", (Doctoral dissertation, PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA), 1996.

[47] Holland J. H., "Adaptation in Natural and Artificial Systems", Ann Harbor: University of Michigan Press, 1992.

[48] Goldberg, D. E, “Genetic Algorithm in Search, Optimization and Machine Learning”, Reading, Mass: Addison-Wesley Publishing Co, 1989.

[49] Moscato, P., "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms", Technical Report C3P 826, Caltech Concurrent Computation Program., 1989.

[50] Sorensen, K., “A framework for robust and flexible optimisation using metaheuristics with application in supply chain design”, PhD thesis, University of Antwerp, 2003.

[51] Gao J., He G. & Wang, Y., "A new parallel genetic algorithm for solving multiobjective scheduling problems subjected to special process constraint", International Journal Advanced Manufacturing Technology, 43, 2009, pp. 151-160.

[52] Cantú-Paz E., "A survey of parallel genetic algorithms", Calculateurs Parallèles, Réseaux et Systems Repartis, 10(2), 1998, pp. 141–171, Hermès, Paris.

[53] Cantú-Paz E., Goldberg D.E., "Parallel genetic algorithms: theory and practice. Computing Methods Applications on Mechanics and Engneering, 186, 2000, pp. 221–238.

[54] Chipperfield A. & Fleming P., "Parallel genetic algorithms",. Parallel and Distributed Computing Handbook, McGraw-Hill, New York, 1996, pp. 1118–1143.

[55] Dorigo M., Maaniezzo V., "Parallel Genetic Algorithms, Introduction and Overview of Current Research", Journal Stender (Ed.), IOS Press, 1993.

[56] Alba E., Nebro A. J., Troya J. M., "Heterogeneous computing and parallel genetic algorithms", Journal of Parallel Distributed Computing, 65, 2002, pp. 1362–1385.

[57] Alba E., Tomassini M., "Parallelism and evolutionary algorithms", IEEE Transactions on Evolutionary Computing, 6(5), 2002, pp. 443–462.

[58] Goldberg D. E., "The design of innovation: lessons from genetic algorithms, lessons for the real world. Tech", Forecasting and Social Change, 64 (1), 2000, pp. 7–120.

[59] Gong Y., Nakamura M., Tamaki S., "Parallel genetic algorithms on line topology of heterogeneous computing resources", In: H.-G. Bauer(Ed.), Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, , New York, 2005, pp. 1447–1454.

[60] Dieter L., Elise H., "The evolution of infanticide by males in ammalian societies", Science, American Association for the Advancement of Science (AAAS), 2014, 346 (6211), pp.841-844.

[61] Sarah B. H., "Infanticide as a Primate Reproductive Strategy: Conflict is basic to all creatures that reproduce sexually, because the genotypes, and hence self-interests, of consorts are necessarily nonidentical. Infanticide among langurs illustrates an extreme form of this conflict", American Scientist, 65 (1), 1977, pp. 40-49, Published By: Sigma Xi, The Scientific Research Honor Society.

Résumé

Les métaheuristiques bio-inspirées ont connu et continuent à connaître un intérêt croissant pour la résolution des problèmes d'optimisation NP-Difficiles. Les algorithmes génétiques sont parmi les métaheuristiques bio-inspirées basés sur le mécanisme de reproduction chez les espèces. Ils ont fait l'objet de plusieurs recherches visant à résoudre de nombreux problèmes. Cependant, nous avons remarqué que le mécanisme de reproduction et de survie peut différer d'une espèce à une autre, et celui de certaines espèces peut se révéler intéressant pour l'inspiration de nouveaux algorithmes.

Chez certaines espèces, pour des raisons de domination ou autres, les mâles tuent les petits des autres mâles pour maximiser ainsi leurs chances d'assurer une descendance. Pour cela, les mères développent des stratégies pour la protection de leurs petits.

Dans ce travail nous présentons deux nouvelles métaheuristiques, la première est une amélioration de l'algorithme génétique modèle des îles en introduisant un paramètre qui permet la gestion de migration. La deuxième est basée sur la stratégie remarquée chez les mères de l'espèce des singes babouins chacma pour la protection de leurs petits contre l'infanticide. Les deux algorithmes ont été utilisés pour la résolution de problème d'ordonnancement dans un atelier Flow Shop. Une étude de sensibilité a été effectuée. Les résultats obtenus par les deux algorithmes ont montré leur efficacité.

Mots clés : Ordonnancement, Flow shop, Métaheuristiques bio-inspirées, Algorithme génétique parallèle, Les Babouins.

Abstract

Bio-inspired metaheuristics have seen and continue to see growing interest in solving NP-Hard optimization problems. Genetic algorithms are among the bio-inspired metaheuristics based on species reproduction mechanism. They have been the subject of several researches applied to solve many problems. However, we noticed that the mechanism of reproduction and survival can differ from one species to another, some of that could be used to develop new algorithms. In some species, for dominance or other reasons, males kill other male's offspring. Consequently, mothers develop strategies for the protection of their offspring. In this work, we present two new metaheuristics, the first one is an improvement of the island model genetic algorithm by introducing a migration management parameter. The second is based on the strategy observed in the baboon chacma females for the protection of their offspring against infanticide. A sensitivity study was carried out for both algorithms. Computational results have shown the effectiveness of the developed algorithms.

Key words: Scheduling, Flow shop, metaheuristics, Island Model Genetic algorithm, Baboon Monkey.

الملخص

شهدت فوقيات الاستدلال المستوحاة من الطبيعة ولا تزال اهتمامًا متزايدًا في حل اشكاليات كثيرات الحدود الصعبة. تعد الخوارزميات الجينية من بين فوقيات الاستدلال البيولوجية المستوحاة من آلية التكاثر لدى الكائنات التي شكلت موضوع العديد من الأبحاث التي تهدف إلى حل العديد من الاشكاليات. ومع ذلك، لاحظنا أن آلية التكاثر والبقاء لدى الكائنات يمكن أن تختلف من نوع إلى آخر، ويمكن أن تكون آلية بعض الأنواع مثيرة للاهتمام لإلهام خوارزميات جديدة.

في بعض الأنواع، سواء بداعي الهيمنة أو لأسباب أخرى، يقتل الذكور صغار الذكور الأخرى لزيادة فرصهم في ضمان النسل. لهذا ، تضع الأمهات استراتيجيات لحماية صغارهن. في هذا العمل ، نقدم فوقيتي استدلال جديدتين، الأولى تهدف الى تحسين الخوارزمية الجينية النموذجية للجزر من خلال إدخال وسيط تسمح بإدارة الهجرة. أما الثاني فيقوم على الاستراتيجية التي لوحظت لدى الأمهات في قردة البابون لحماية صغارهم من القتل. تم استخدام كلا الخوارزميتين لحل مشكلة الجدولة في ورشة عمل ذات مسار واحد. تم إجراء دراسة الحساسية لكلا الخوارزميتين. النتائج التي حصلت عليها الخوارزميتان أظهرت فعاليتها.

الكلمات المفتاحية الترتيبات، وحيدة المسار، الخوارزميات المستوحاة من الطبيعة، خوارزمية الوراثة المتوازية، قردة البابون.