REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA
FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Informatique Décisionnelle et Optimisation

Par :Bara Houria

SUJET

Développement et implémentation d'une méthode hybride pour la résolution du problème d'assignation quadratique

Soutenu publiquement le : 13 / 07 / 2019
devant le jury composé de :

Dr DEBBI Aimad-Eddine	**Université de M'sila**	**Président**
Dr. Lamiche Chaabane	**Université de M'sila**	**Rapporteur**
Dr . LAKEHAL Ayat Raouf Ouanis	**Université de M'sila**	**Examinateur**

Promotion : 2018 /2019

TABLE DES MATIERES

Chapitre 3 : Méthode proposée

Liste des figures

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

LISTE DES ABREVIATIONS

QAP : Problème d'affectation Quadratique (de l'anglais quadratic assignment problem)

NP : Complexité polynomiale non déterministe (de l'anglais : Non déterministe polynomial time).

P : Complexité polynomiale (de l'anglais : polynomial time).

TSP: Le problème du voyageur de commerce (de l'anglais: Travelling Salesman Problem)

BB : La méthode par séparation et évaluation (de l'anglais: Branch and Bound)

DM : La méthode de descente (de l'anglais : Descent method)

ILS : la recherche locale réitérée (de l'anglais: Iterated Local Search)

TS : La recherche tabou (de l'anglais: Tabu Search)

POC : problèmes d'optimisation combinatoire (de l'anglais: Combinatorial Optimization Problem)

SA: Le recuit simulé (de l'anglais: Simulated Annealing)

ABC : L'optimisation par colonies d'abeilles artificielles (de l'anglais: Artificial Bee Colony)

GA: L'algorithme génétique (de l'anglais: Genetic Algorithm)

ACO: L'optimisation par colonies de fourmis (de l'anglais: Ant Colony Optimization)

QAPLIB : Bibliothèque, reconnue par la communauté œuvrant sur le QAP (de l'anglais: Bibliothèque d'instances du QAP)

MCP : Le problème de la clique maximale (de l'anglais : maximum clique problem)

GASA :L'algorithme hybride de génétique et de recuit simulé (de l'anglais : Genetic Algorithm with Simulated Annealing) .

GATS :L'algorithme hybride de génétique et de tabous ((de l'anglais : Genetic Algorithm mixed with Tabu Search) .

BBO : Optimisation basé sur la biogéographie (Biogeography - based optimization)

BBOTS : L'algorithme hybride de biogéographie et de tabou (de l'anglais : Biogeography - based optimization mixed Tabu Search) .

INTRODUCTION GÉNÉRALE

Le succès d'un projet ou le développement d'une entreprise est dans de nombreux cas liés à la capacité des ingénieurs et des décideurs de résoudre des problèmes de type minimiser les coûts et maximiser la production avec l'inclusion d'un certain nombre de paramètres. Ce type de problème appelé problème d'optimisation, où nous voulons minimiser une fonction de coût ou maximiser une fonction objective.

Un problème d'optimisation combinatoire *(POC)* comprend un ensemble fini de solutions, où chaque solution doit respecter un ensemble de contraintes relatives à la nature du problème. On associe à chaque solution une valeur, nommée valeur de l'objectif, qui est évaluée à l'aide d'une fonction, la fonction objectif.

Il existe de nombreux exemples de problèmes d'optimisation combinatoire . Dans notre étude nous intéressons au problème classique mais il suscite encore aujourd'hui beaucoup d'intérêts, soit dans le monde pratique ou académiques, c'est le problème d'affectation quadratique ou bien "Quadratic Assignment Problem (QAP)" qui consiste à trouver le placement optimal de n objets "usines, matériels" sur n locations "sites", tout en minimisant le coût qui dépend à la fois des distances inter locations et des flux inter objets . Il s'agit de trouver l'affectation qui permettra de minimiser le coût total. Le QAP est un problème important en optimisation combinatoire car il possède de nombreuses applications (placement, ordonnancement, synthèse d'images, etc).

Les problèmes d'optimisation sont souvent faciles à définir, mais généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace et valable pour toutes les données. Il y’a généralement deux approches pour résoudre les problèmes d’optimisation : une approche de résolution exacte et approche de résolution approximative. La première cherche à trouver la solution exacte d'un problème donné, cette approche est limitée à des petits instances de problèmes difficiles. La deuxième tendance est la résolution approximative, où nous essayons de trouver des bonnes solutions en utilisant des méthodes stochastiques appelées heuristiques, parmi lesquelles nous pouvons trouver des méthodes générales qui peuvent être appliqués sur un large éventail de problèmes, on appele ces méthodes les métaheuristiques.

il existe une autre approche pour résoudre les POC: il s'agit des algorithmes hybrides. Ceux-ci combinent les forces de deux ou plusieurs algorithmes pour en faire un seul. On peut combiner les algorithmes de différentes façons: en les exécutant soit en série, soit en parallèle ou en incorporant un algorithme à un autre, donnant lieu à une méthode maître et esclave.

L'objectif de ce travail est d'aborder le problème d'affectation quadratique (QAP) en utilisant des métaheuristiques comme approche de résolution, en particulier l'algorithme algorithme génétique couple à l'algorithme de recuit simule. L'idée c'est de faire une coopération entre les deux algorithmes de recherche, tel que les résultats fournis par le premier algorithme sont les solutions initiales du second , ce qui signifie que nous parlons alors d'un algorithme hybride.

Notre travail est méthodologiquement organisé comme suit :

Le Chapitre 1 est consacré essentiellement au problème de l'optimisation combinatoire. Il traite aussi des questions de la théorie de la complexité et de la complexité en elle même. Il va sans dire qu'il traite des différentes classes de la complexité. Nous y avons inclus des exemples de problèmes d'optimisation combinatoire.

Le Chapitre 2 quant à lui, intitulé revue sur les métaheuristiques d'optimisation, traite des méthodes de résolutions exactes et approchées. Il est essentiellement axé sur les métaheuristiques d'une part à solution unique pour lesquelles différentes méthodes sont énoncées, et d'autre part pour les métaheuristiques a base de population de solutions. Comme de bien entendu, nous parlons de la méthode hybride.

Le Chapitre 3 traite de manière exhaustive du problème quadratique. Après un aperçu historique, il s'intéresse aux instances du problème d'affectation. Ensuite nous avons présenté le problème d'affectation quadratique par le truchement de sa formalisation mathématique appuyée d'un exemple concret. De plus nous avons traité de la complexité des problèmes de cette nature. Enfin, arrive le tour de la méthode hybride proposée entre 'l'AG' algorithme génétique et le 'RS' recuit simulé. Alors vint le tour de la présentation des résultats obtenus et la comparaison.

CHAPITRE 1

OPTIMISATION COMBINATOIRE : CONCEPTS FONDAMENTAUX

1. Introduction

L'importance de l'optimisation combinatoire réside dans la nature des problèmes qu'elle participe à résoudre. Ses domaines sont les mathématiques et l'informatique. Les recherches actuelles, de par leur complexité ne peuvent s dispenser des outils incontournables aux fins d'objectifs : particulièrement la minimisation (maximisation) et comme de bien entendu l'optimisation.

La problématique de l'optimisation ne peut être approchée sans la création d'algorithmes efficaces et sans faire intervenir le concept de la théorie de la complexité.

La littérature de spécialité recense des classes de problèmes. Nous citons les suivants : Les classes P, NP et NP-complet , NP-difficiles.

Nous étudierons à titre d'exemple le problème du sac à dos, celui de la coloration des graphes, du voyageur de commerce, de l'affectation linéaire et en fin, l'objet de ma recherche le problème de l'affectation quadratique.

2. L'optimisation

2.1 Définition

L'optimisation c'est l'art de comprendre un problème réel, de pouvoir le transformer en un modèle mathématique que l'on peut étudier afin d'en extraire les propriétés structurelles et de caractériser les solutions du problème. Enfin, c'est l'art d'exploiter cette caractérisation afin de déterminer des algorithmes qui les calculent mais aussi de mettre en évidence les limites sur l'efficience et l'efficacité de ces algorithmes [9] .

2.2 Optimisation combinatoire

L'optimisation combinatoire est une technique mathématique, qui consiste à minimiser ou maximiser une fonction objectif (Coût, temps, distance, etc.). Dont le but est de rechercher et définir une solution optimale la plus appropriée pour l'optimisation à partir d'un ensemble de solutions possible .

2.3 Problème d'optimisation combinatoire

Un problème d'optimisation combinatoire est défini par un ensemble d'instances où à chaque instance on associe un ensemble discret de solutions S composé par un sous ensemble des solutions admissibles (ou réalisables) X et une fonction coût f (fonction objectif) qui associe à chaque solution s ∈ X un nombre réel f(s) représentant son coût. La résolution d'un problème d'optimisation combinatoire consiste à trouver une solution S* appartenant à X permettant d'optimiser la fonction coût f . La solution S* est appelée solution optimale ou optimum global [14] .

Une définition proposée dans [4] est la suivante :

Un problème d'optimisation se définit comme la recherche, parmi un ensemble de solutions possibles S (appelé aussi espace de décision ou espace de recherche), de la (ou des) solution(s) x^* qui rend(ent) minimale (ou maximale) une fonction mesurant la qualité de cette solution. Cette fonction est appelée fonction objectif ou fonction coût. Si l'on pose f : S → R la fonction objectif à minimiser (respectivement à maximiser) à valeurs dans R, le problème revient alors à trouver l'optimum x^* S tel que f(x^*) soit minimal (respectivement maximal).

Lorsque l'on veut résoudre un problème d'optimisation, on recherche la meilleure solution possible à ce problème, c'est-à-dire l'optimum global. Cependant, il peut exister des solutions intermédiaires, qui sont également des optimums, mais uniquement pour un sous-espace restreint de l'espace de recherche : on parle alors d'optimums locaux. Cette notion est illustrée dans la figure 1.1.

La seule hypothèse faite sur S est qu'il s'agit d'un espace topologique, sur lequel est définie une notion de voisinage. Cette hypothèse est nécessaire pour définir la notion de solutions locales du problème d'optimisation. On peut alors définir un optimum local (relativement au voisinage V) comme la solution x^* de S telle que

$f(x^*) \leq f(x)$;$\forall x \in V(x^*)$

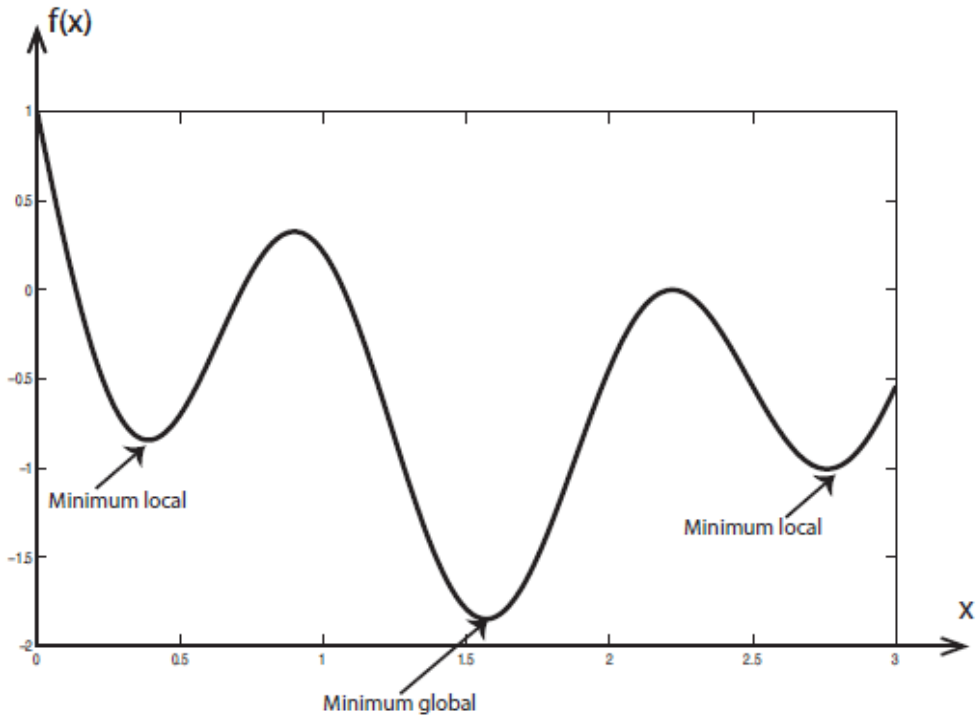


Figure 1.1 Différence entre un optimum global et des optima locaux

dans le domaine d'optimisation on distingue deux types de minimums :

Minimum Local une solution s est minimum local par apport à une structure de voisinage N si $\forall s' \in N(s)$, $f(s) \leq f(s')$.

Minimum Global : une solution s est minimum global si $\forall s' \in S$, $f(s) \leq f(s')$

Voisinage : le voisinage est une fonction notée N qui associe un sous ensemble de S à toute solution s, les voisins de s sont $s' \in N(s)$.

3. La complexité et la théorie de la complexité

3.1 Complexité d'un algorithme

La complexité d'un algorithme c'est le temps et l'espace mémoire nécessaires pour son exécution. Elle est calculée en fonction du nombre N des données appelées aussi taille du problème. Dans le cas le plus défavorable, la théorie de complexité permet de majorer le

nombre d'opérations nécessaires par une fonction de la taille N du problème posé. La complexité d'un algorithme est $f(N)$ noté $o(f(N))$, s'il $\exists$ une constante C et un entier A , tels que le nombre d'instructions élémentaires I(N) vérifie $I(N) \leq Cf(N)$ pour tout $N \geq A$. Si $f(N)$ est un polynôme alors l'algorithme est dit polynomial [14].

Un algorithme polynomial

est un algorithme dont le nombre d'opérations élémentaires nécessaires pour résoudre un exemple de taille n est une fonction polynomiale en n.

Un algorithme efficace

on dit qu'un algorithme est efficace si le nombre des opérations nécessaires pour résoudre un problème est bornée par une fonction polynomiale d'un paramètre caractérisant la taille du problème considéré.

Algorithme non polynomiale

est un algorithme dont le nombre d'opérations n'est pas borné par un polynôme de n.

4. Les différentes classes de complexité

4.1 La classe P

La classe P contient tous les problèmes relativement faciles c'est à dire ceux pour lesquels on connaît des algorithmes efficaces. Plus formellement, ce sont les problèmes pour lesquels on peut construire une machine déterministe dont le temps d'exécution est de complexité polynomiale (le sigle P signifie « Polynomial time ») [6].

4.2 La classe NP

Les problèmes de la classe **NP** sont ceux pour lesquels on peut construire une machine de Turing non déterministe dont le temps d'exécution est de complexité polynomiale (le signe NP provient de « Nondéterministic Polynomial time ») et (non de « Non Polynomial »).

Contrairement aux machines déterministes qui exécutent une séquence d'instructions bien déterminée, les machines non déterministes ont la remarquable capacité de toujours choisir. la meilleur séquence d'instructions qui mène à la bonne réponse lorsque celle-ci existe. Ce concept abstrait est en fait la base de toute la théorie de la **NP**-complétude [6] .

4.3 La classe NP-complets

Parmi l'ensemble des problèmes appartenant à NP, il en existe un sous ensemble qui contient les problèmes les plus difficiles : on les appelle le problèmes NP complets.

Un problème NP-complets possède la priorité que tout problème dans NP peut être transformé (réduit) en celui-ci en temps polynomial. C'est à dire qu'un problème est NP-complets quand tous les problèmes appartenant à NP lui sont réductibles. Si on trouve un algorithme

polynomial pour un problème NP-complets, on trouve alors automatiquement une résolution polynomiale de tous les problèmes de la classe NP [6] **.**

4.4 La classe NP-difficiles

Un problème est NP-difficile s'il est plus difficile qu'un problème NP-complet , c'est à dire s'il existe un problème NP-complet se réduisant à ce problème par la réduction de Turing. Ceci explique pourquoi, lors de l'étude d'un nouveau problème, on commence par chercher à classer ce problème. Si l'on parvient à montrer qu'il est polynomial, le problème sera résolu. Si par contre, on parvient à montrer qu'il est NP-complet, la recherche d'un algorithme exact pour résoudre un tel problème ne sera pas de première priorité, et il sera approprié de se concentrer sur des méthodes heuristiques que la plupart des spécialistes de l'optimisation combinatoire ont orienté leurs recherches pour les développer.

Une méthode heuristique est souvent définie comme une procédure exploitant au mieux la structure du problème considéré, dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible [6].

5. Exemples de problèmes d'optimisation combinatoire

5.1 Le problème de sac à dos

Le "problème du sac-à-dos" est un problème de sélection qui consiste à maximiser un critère de qualité sous une contrainte linéaire de capacité de ressource. Il doit son nom à l'analogie qui peut être faite avec le problème qui se pose au randonneur au moment de remplir son sac-à-dos : il lui faut choisir les objets à emporter de façon à avoir un sac le plus possible, tout en respectant son volume. Plus formellement, on peut le décrire de la façon suivante. Soit un ensemble de n éléments et une ressource disponible en quantité limitée, b. Pour j = 1 à n, on note pj le profit associé à la sélection de l'élément j et on note aj la quantité de ressource que nécessite l'élément j, s'il est sélectionné. Les coefficients pj et aj prennent des valeurs positives pour tout j = 1 à n. Le problème du sac-à-dos consiste à choisir un sous-ensemble des n éléments qui maximise le profit total obtenu, en respectant la quantité de ressource disponible [5] .

On associe à chaque élément j une variable de sélection, x_j, binaire, égale à 1 si j est sélectionné, égale à 0 sinon. Le profit total obtenu peut alors s'écrire comme la somme : $\sum_{j=1}^{n} p_j . . x_j$ et la quantité totale de ressource utilisée comme la somme : $\sum_{j=1}^{n} a_j . . x_j$. Le problème du sac-à-dos se modélise donc sous la forme :

$$\begin{cases} Max \ \sum_{j=1}^{n} p_{j..} x_j \\ S.C. \ \sum_{j=1}^{n} a_{j..} x_j \ \leq b \\ \quad x_j \in \{0,1\} \ \ \forall j = 1..n \end{cases} \qquad (1.1)$$

5.2 Problème du voyageur de commerce

Le problème du voyageur de commerce (en anglais Travelling Salesman Problème : TSP) ou PVC. C'est un problème classique d'optimisation combinatoire NP-complet.

Le voyageur du commerce désire visiter un certain nombre de villes (ou clients). Il doit débuter et finir sa tournée par la même ville de départ, en visitant chacune des autres villes, une et une seule fois, l'objectif ′étant de trouver la tournée qui minimise la distance totale parcourue. Le PVC permet la formulation de nombreuses situations réelles. Depuis son apparition il n'a pas cessé d'attirer l'attention des chercheurs [10].

Formulation :

Soit G = (X, U) un graphe, où X est l'ensemble des sommets et U l'ensemble des arrêts (ou arcs si le graphe est orienté) . une formulation mathématique du PVC, où ils définissent :

- Une variable binaire :

$$X_{ij} = \begin{cases} 1 \text{ si } & \text{l'arc (i , j) est utilise dans la tournée} \\ 0 \quad & \text{sinom} \end{cases}$$

- C_{ij} le coût du parcours de l'arc (i, j).

Donc on obtient la formulation suivante :

$$\begin{cases} \text{Minimiser} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_{ij} & (A) \\ \text{Sous} \quad \sum x_{ij} = 1; \quad \forall j \in X & (B) \\ \quad \sum_{j \in X} x_{ij} = 1; \quad \forall i \in X & (C) \\ \quad \sum_{ij \in S} \leq |S| - 1; \ \forall S \subset X; 2 \leq |S| - 2 & (D) \\ \quad x_{ij} \in 0,1 \quad \forall i \in X, \forall j \in X & (E) \end{cases} \qquad (1.2)$$

La (A) formule l'objectif du PVC qui est la minimisation de la longueur totale parcourue par le voyageur. Les contraintes (B) et (C) assurent que le voyageur passe une et une seule fois par chaque sommet (client). La contrainte (D) garantit l´élimination de la formation de sous-tour.

5.3 Le problème de coloration des graphes

Il s'agit de colorier tous les sommets d'un graphe tel que chaque sommet ait une couleur différente de la couleur de ses sommets adjacents, tout en minimisant le nombre de couleurs. La coloration de graphe est appliquée dans multiples domaines tels que l'emploi de temps, la coloration des cartes, l'affectation de fréquence en téléphonie mobile, la gestion de chaînes logistiques, la gestion du trafic aérien, l'affectation des registres dans les compilateurs, etc.
Formellement, ce problème est défini comme suit : soit le graphe G = (V,E) avec V l'ensemble des sommets et E l'ensemble des arêtes. Une coloration valide d'un graphe
G = (V,E) est une fonction c : v ← c(v) associant à tout sommet v ∈ V une couleur c(v), en s'assurant que c(v) ≠ c(u) pour toute arête [u,v] ∈ E. Si le nombre de couleurs utilisé est k, la coloration de G est une k-coloration. La valeur minimale de k pour laquelle une k-coloration est réalisable est appelé nombre chromatique du graphe, noté $\chi(G)$. La résolution du problème de la coloration de graphe consiste à décider s'il y a une coloration avec k couleurs (problème de décision NP-complet), soit à trouver son nombre chromatique (problème d'optimisation NP-difficile) [9].

5.4 Le problème d'affectation quadratique

Le problème d'affectation quadratique est l'un des problèmes d'optimisation combinatoire NP-complets les plus connus et ayant suscité de nombreux travaux [9] .
Plus de détail sur ce problème est présenté dans le chapitre 3.

6. Conclusion

Tout au long de notre chapitre, nous avons présenté les concepts fondamentaux de l'optimisation combinatoire, les classes des problèmes NP-difficile et leurs complexité, ensuite nous avons présenté quelques problèmes d'optimisation qui sont très connus Dans le chapitre suivant nous procèderons à une revue sur les métaheuristiques d'optimisation combinatoire.

CHAPITRE 2

REVUE SUR LES METAHEURISTIQUE D'OPTIMISATION COMBINATOIRE

1. Introduction

Deux catégories de méthodes se présentent à nous pour la résolution des problèmes d'optimisation combinatoire : les méthodes dites exactes et les méthodes approchées. Malgré leur efficacité, les premières coûtent cher en temps et espace mémoire des ordinateurs. Alors, que si les secondes sont plus économiques, elles sont moins efficaces que les premières. Les métaheuristiques sont un moyen médian pour équilibrer entre le temps nécessaire à l'exécution du programme concerné et la qualité de la solution obtenue. Nous rappelons que la plupart des métaheuristiques sont d'inspiration biologique au point qu'elles sont appelées « algorithme inspirés de la nature ».

Ce chapitre se focalise sur la présentation de quelques méthodes utilisées pour la résolution des problèmes NP-difficiles avec une concentration sur les métaheuristiques.

2. Méthodes de résolutions

La résolution de différentes sortes de problèmes rencontrés dans notre vie quotidienne a poussé les chercheurs à proposer des méthodes de résolution et à réaliser de grands efforts pour améliorer leurs performances en termes de temps de calcul nécessaire et/ou de la qualité de la solution proposée. [2]

Au fil des années, de nombreuses méthodes de résolution de problèmes de différentes complexités ont été proposées. Ainsi, une grande variété et des différences remarquables au niveau du principe, de la stratégie et des performances ont été discernées. Cette variété et ces différences ont permis de regrouper les différentes méthodes de résolution de différents problèmes en deux classes principales : la classe de méthodes exactes, la classe des méthodes approchées comme dans la Figure 2.1.

2.1 Méthode exactes

Dans Les méthodes exactes toutes les solutions de l'espace de recherche sont énumérées implicitement en utilisant des mécanismes qui détectent des échecs (calcul de bornes). Grâce à Ces méthodes on peut trouver des solutions optimales. Mais ces méthodes s'avèrent, malgré les progrès réalisés, plutôt inefficaces à mesure que la taille du problème devient importante. Dans cette classe des méthodes exactes [1].

Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable et rencontrent généralement des difficultés face aux applications de taille importante. Dans cette classe des méthodes exactes, on peut trouver les algorithmes classiques suivants : La programmation dynamique, La programmation linéaire, Les méthodes de recherche arborescente (Branch & bound) .

2.2 Méthodes approchées

Une méthode approchée (incomplètes) est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objective en un temps raisonnable, mais sans garantie 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 .

Ils peuvent être classés en deux catégories : les heuristiques et les métaheuristiques.

2.2.1 Heuristique

Le terme heuristique vient du verbe grec heuriskein signifiant trouver. Une heuristique permet de trouver une "bonne" solution, en un temps raisonnable, seulement elle n'offre aucune garantie sur l'optimalité de la solution trouvée [2].

Plusieurs définitions des heuristiques ont été proposées par plusieurs chercheurs dans la littérature, parmi lesquelles:

Définition1 : En (1963) Feignebaum et Feldman « Une méthode heuristique (ou simplement une heuristique) est une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillible de ce qui est la meilleure chose à faire »[8].

Définition2 : En **(**1980) Newell, « Les heuristiques sont des règles empiriques et des morceaux de connaissances, utiles (mais non garanties) pour effectuer des sélections différentes et des évaluations » [11].

2.2.2 Métaheuristique

Méta est un préfixe signifiant « au-delà », « plus que ça », « `a un niveau supérieur », lorsqu'il est rajout à l'heuristique il donne Métaheuristique qui veut dire trouver au-delà, ou trouver plus que ça .

En 1996, I.H. Osman et G. Laporte définissaient « la métaheuristique comme « un processus itératif qui subordonne et qui guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales » » [7].

Les méthodes dites métaheuristiques sont des méthodes générales, des heuristiques polyvalentes applicables sur une grande gamme de problèmes. Elles peuvent construire une alternative aux méthodes heuristiques lorsqu'on ne connaît pas l'heuristique spécifique à un problème donné.

Les métaheuristiques groupées en deux classes: des métaheuristiques à base de solution unique (SA, TS, …) et des métaheuristiques à base de population de solutions (GA, PSO, …).

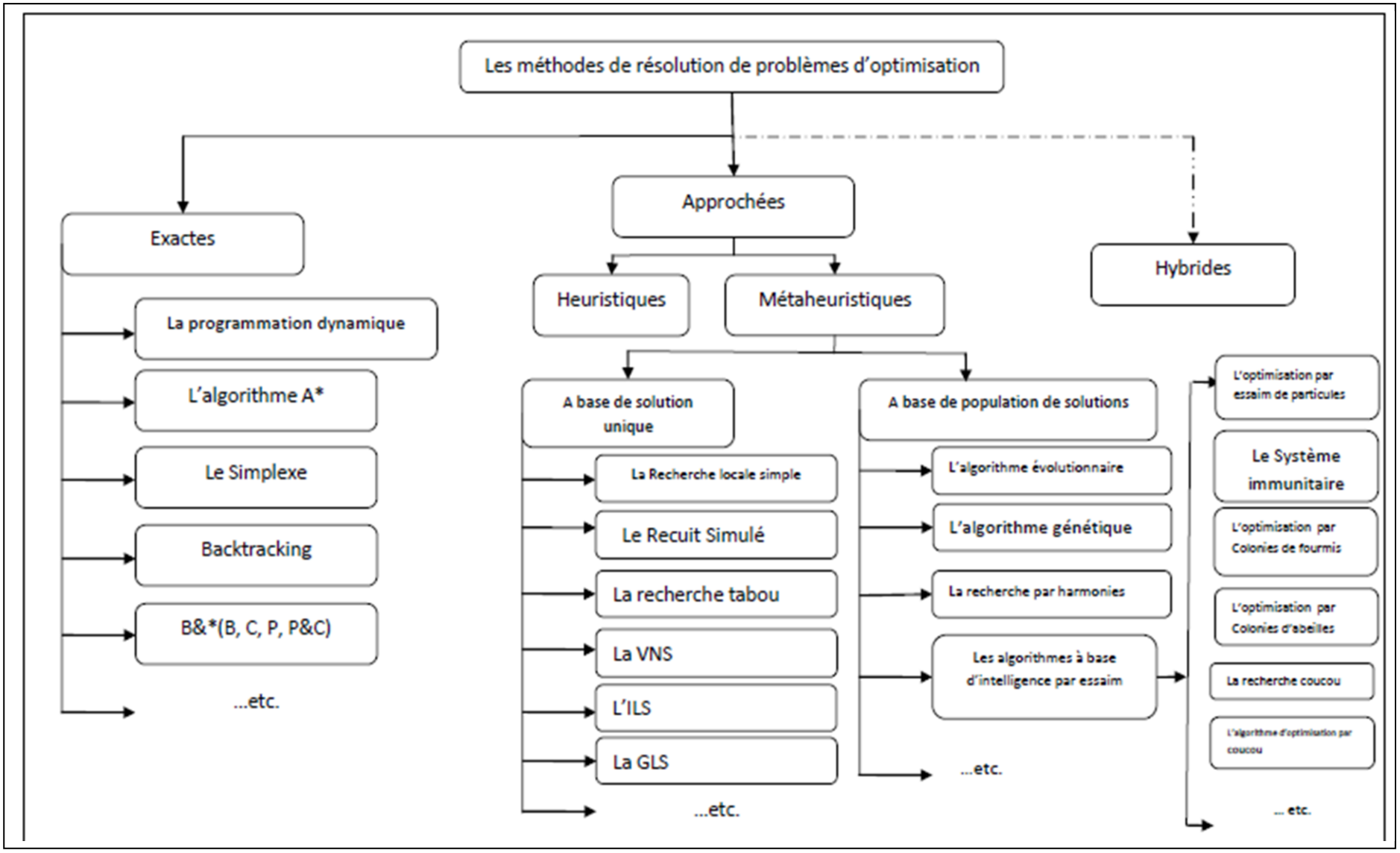


Figure 2.1 Classification de méthodes de résolution de problèmes d'optimisation.

3. Classification des métaheuristiques

3.1 Les métaheuristiques à base de solution unique

Nous présentons les métaheuristiques à base de solution unique, aussi appelées méthodes de trajectoire. Contrairement aux métaheuristiques à base de population, les métaheuristiques à solution unique commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, la méthode du recuit simulé, la recherche tabou, la recherche à voisinage variable, la recherche locale itérée, et leurs ariantes.

3.1.1 La méthode de descente

La recherche locale simple ou la descente (DM : Descent method) est un algorithme d'amélioration très ancien. Son principe consiste à explorer le voisinage de la solution courante afin d'améliorer sa qualité progressivement, comme le montre la figure 2.2 qui représente un schéma d'évolution d'une recherche locale simple [2] .

Le principe de la méthode de descente (dite aussi basic local search) consiste à partir d'une solution s on choisit une solution s' qui appartient au voisinage de s, telle que s'améliore la recherche (généralement telle que f(s') < f(s)).

L'algorithme de méthode de descente est :

Algorithme 2.1 Méthode de descente

Procédure descente _ simple (solution initiale s)
Répéter :
Choisir s' dans N(s)
Si f(s') < f(s)
alors s ← s'
Jusqu'à ce que f(s') ≥ f(s), s' ∈ S
Fin.

On peut décider : soit d'examiner toutes les solutions du voisinage et prendre la meilleure de toutes (ou prendre la première trouvée), soit d'examiner un sous-ensemble du voisinage.

La méthode de descente est la méthode de recherche locale la plus élémentaire. On peut la schématiser comme suit :

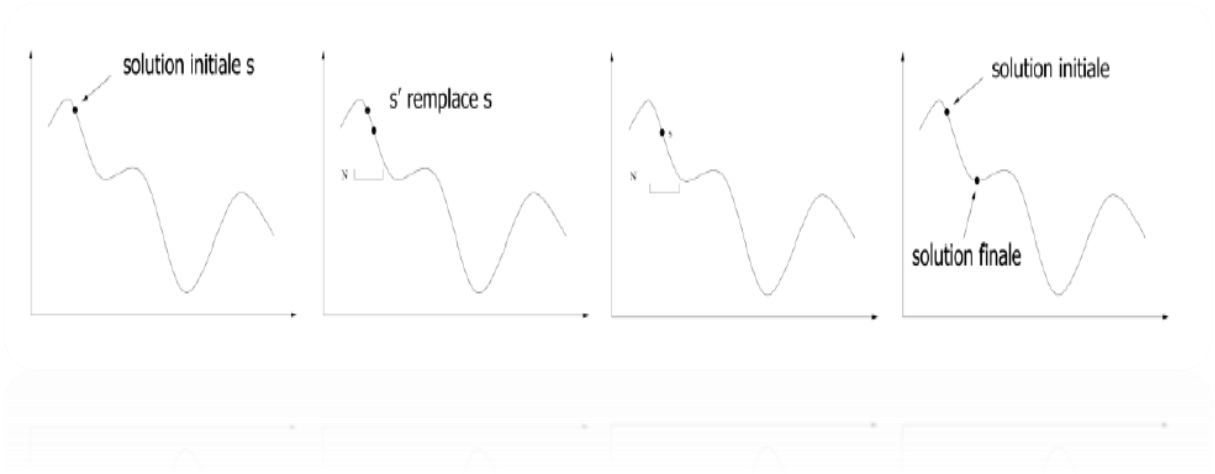


Figure 2.2 évolution d'une solution dans la méthode de descente.

En général, l'efficacité des méthodes de descente simples est très peu satisfaisante. D'abord, par définition, la recherche s'arrête au premier minimum local rencontré, c'est là leur principal défaut. Pour améliorer les résultats, on peut lancer plusieurs fois l'algorithme en partant d'un jeu de solutions initiales différentes, mais la performance de cette technique décroît rapidement.

3.1.2 Méthode de recherche local itérée

La méthode RLI (ILS :Iterated local search) est une méthode dotée d'un mécanisme permettant au processus de résolution d'échapper aux optimums locaux. Ce mécanisme consiste à alterner une méthode de recherche locale et une perturbation [10].

Une première stratégie pour pallier l'arrêt brutal de la recherche sur un optimum local est d'itérer la méthode de descente. On déroule les étapes suivantes à partir de l'optimum trouvé :

- (i) appliquer une perturbation sur la solution courante
- (ii) appliquer une méthode de descente sur cette solution
- (iii) choisir via un critère d'acceptation si le nouvel optimum devient la solution courante et revenir en (i) jusqu'à ce que le critère d'arrêt soit atteint.

La perturbation peut consister à :

- Redémarrer d'une solution prise aléatoirement dans l'espace de recherche ;
- Choisir une solution dans un voisinage lointain de l'optimum ;
- Ou encore à choisir un voisin de même qualité que l'optimum...

L'algorithme de la recherche locale itérée est :

Algorithme 2.2 La recherche local itérée (ILS)

Début

Générer une solution initiale S0

S*← Recherche locale (S0) // *Appliquer la méthode de descente.*

Répéter

S'← Perturbation (S*)

S*'← Recherche locale (S')

S*← Critère d'acceptation (S*,S*') // *Acceptation conditionnelle.*

Jusqu'à la condition d'arrêt est atteint

Fin

Le critère d'acceptation est généralement comme suit :

Accepter la solution S*' SSI f(S*') ≤ f(S*)

3.1.3 La méthode Tabou

RT est une métaheuristique à base d'une solution unique. Elle a été proposée en 1986 par Glover [2]. RT est une méthode de recherche locale avancée, elle est basée sur deux astuces: l'utilisation de la notion du voisinage et l'utilisation d'une mémoire permettant le guidage du recherche.

- En parcourant le voisinage de la solution courante s, RT examine un échantillonnage de solution du voisinage de s et retient toujours la meilleure solution voisine s', même si celle-ci est de piètre qualité que la solution courante s, afin d'échapper de la vallée de l'optimum local et donner au processus de la recherche d'autres possibilités d'exploration de l'espace de recherche afin de rencontrer l'optimum global. Cependant, cette stratégie peut créer un phénomène de cyclage.
- Afin de pallier à ce problème, RT propose l'utilisation d'une mémoire pour le stockage des dernières solutions rencontrées pour ne pas les visiter dans les prochaines itérations et tomber dans le problème du cyclage. Cette mémoire est appelée « la liste tabou ».
- La taille de la liste tabou est limitée, ce qui empêche l'enregistrement de toutes les solutions rencontrées. C'est la raison pour laquelle la liste tabou procède comme une pile FIFO, où la plus ancienne solution.
- Le critère d'aspiration est un mécanisme permet de lever le statut tabou d'une configuration, sans pour autant introduire un risque de cycles dans le processus de

recherche. La technique la plus la plus simple consiste à révoquer le statut tabou d'un mouvement si ce dernier permet d'atteindre une solution de coût inférieur à celui de la meilleure solution trouvée jusqu'à présent.

- L'algorithme de recherche tabou est comme suit:

Algorithme 2.3 La méthode Tabou

Début

Construire une solution initiale s ;
Calculer la fitness f(s) de s ;
Initialiser une liste tabou vide ;
sbes t=s ;

Tant que le critère d'arrêt n'est pas vérifié **faire**

Trouver la meilleure solution s' dans le voisinage de s qui
ne soit pas tabou ou qui vérifie le critère d'aspiration ;
Calculer f(s') ;
Si fitness de (s') est meilleure que fitness de (sbest) **alors**
sbest = s' ;
Fin Si
Mettre à jour la liste tabou ;
s= s' ;

Fin Tant que

Retourner sbest ;

Fin

Les avantages et inconvénients

- Elle a un paramétrage simplifié
- RT exige une gestion de la mémoire de plus en plus lourde
- Efficace dans plusieurs POC: le PVC, le Pb. d'ordonnancement, le Pb. de tournées de véhicules, etc.

3.1.4 La méthode recuit simule

L'algorithme de recuit simule (Simulated Annealing SA) est inspire du principe de la thermodynamique et a été propose par Kirkpatrick , Gelatt et Vecchiet en 1983 [9]. L'idée de cet algorithme consiste a reproduire le phénomène de recuit obtenu lors de refroidissement d'un matériau solide préalablement chauffe : la structure du matériau obtenu varie en fonction de la vitesse de refroidissement. En partant d'une haute température a laquelle le matériau solide est devenu liquide. avec plus de liberté aux atomes qui le composent, la phase de refroidissement conduit la matière liquide a retrouver sa forme solide par une diminution progressive de la température. Si le refroidissement est trop brusque, les atomes peuvent s'éloigner de leur état d'équilibre et causer une imperfection au morceau de métal. Le refroidissement du métal est donc lent et régulier pour permettre aux atomes de se stabiliser peu à peu dans une position d'énergie minimale. L'algorithme de recuit simule reproduit les variations d'énergie observées lors de processus d'un recuit. L'énergie désigne la valeur de la fonction coût de la solution courante et l'état correspond à une solution du problème traité. Il consiste donc en une recherche aléatoire de l'espace d'état à favoriser les descentes, mais sans interdire tout à fait les remontrées. Cependant, une augmentation du niveau d'énergie permet de sortir des minima locaux. L'algorithme montre qu'on tend très rapidement vers un minimum local proche de la meilleure solution .

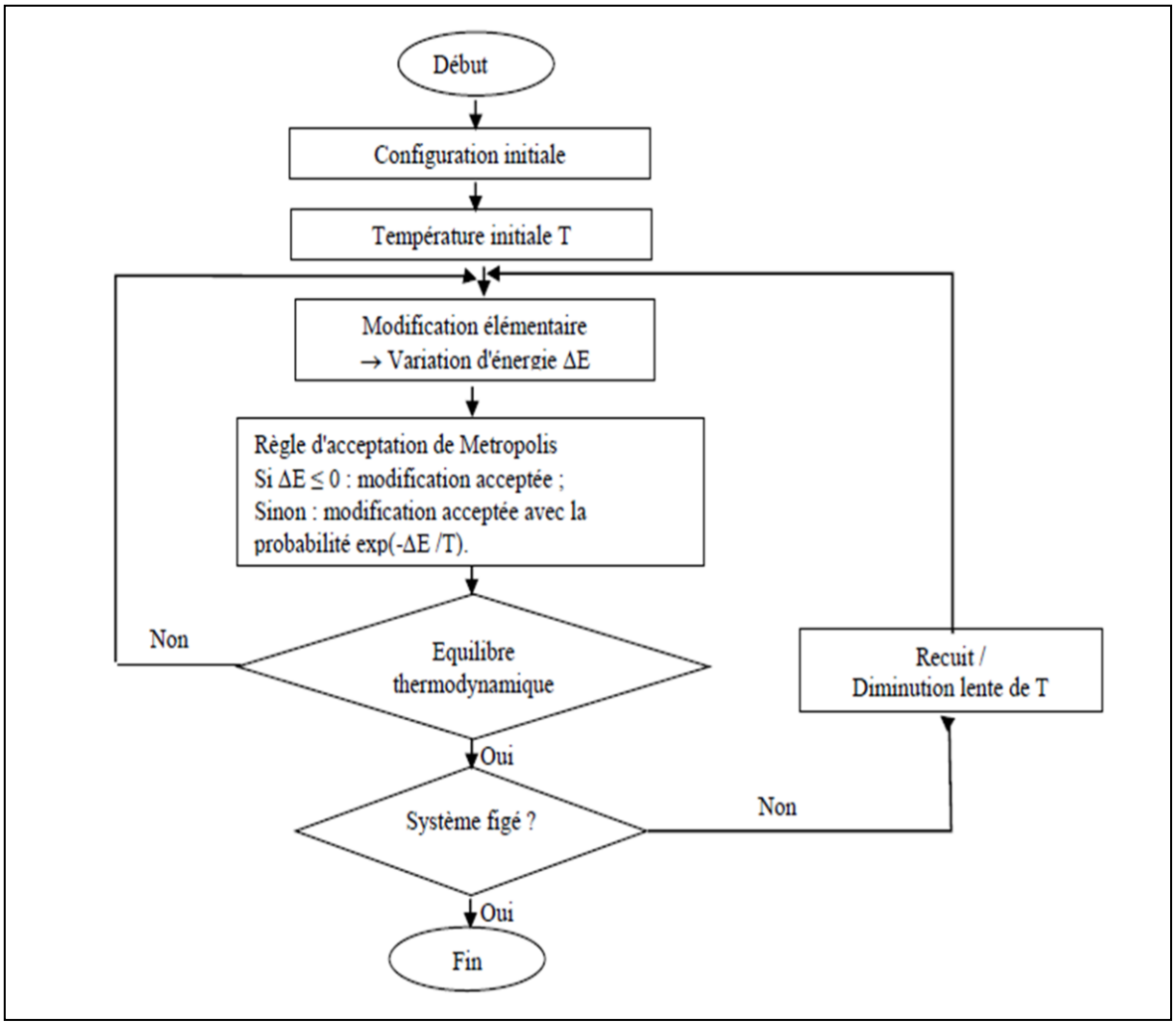


Figure2.3 Organigramme général du recuit simulé.

Initialement on donne le système une très haute température puis on le refroidit petit à petit Le refroidissement du système doit se faire très lentement pour avoir l'assurance d'atteindre un état d'équilibre à chaque température T.

- Le voisinage N(s) d'une solution s. s'appartient à l'ensemble des états atteignables depuis l'état courant en faisant subir des déplacements élémentaires aux atomes du système Physique .
- A chaque itération, une seule solution voisine s'est générée et elle est acceptée si elle est meilleure que la solution courante s dans le cas contraire on a les cas :
- si T grande, exp(-ΔE/T) est de l'ordre de 1, on garde toujours le mouvement, même il est mauvais.

- si T très petit, exp(-ΔE/T) est de l'ordre de 0, donc les mouvements qui augmentent l'énergie (la différence) sont disqualifiés.

Méthode : on tire au hasard dans l'intervalle [0,1[, si le nombre est <exp(-ΔE/T) , on garde sinon on jette.

- La meilleure solution trouvée est mémorisée dans la variable s*.

Les avantages

- Traite les fonctions de coût avec les degrés tout à fait arbitraires de non linéarité, la discontinuité, et l'imprévisibilité.
- Processus assez arbitraire sur les conditions limites et les contraintes imposées sur les fonctions de coût.
- Simple à implémenter.
- Garantie Statistique de trouver une solution optimale.

Les inconvénients

- le non déterminisme.
- Difficulté de choisir les paramètres efficace par exemple le schéma de refroidissement.
- Le compromis entre la vitesse et l'optimisation.

3.2 Les métaheuristiques à base de population de solutions

Les métaheuristiques à base de population de solutions débutent la recherche avec une panoplie de solutions. Elles s'appliquent sur un ensemble de solutions afin d'en extraire la meilleure (l'optimum global) qui représentera la solution du problème traité . L'idée d'utiliser un ensemble de solutions au lieu d'une seule solution renforce la diversité de la recherche et augmente la possibilité d'émergence de solutions de bonne qualité [2].

On distingue dans cette catégorie, les algorithmes évolutionnaires, qui sont une famille d'algorithmes issus de la théorie de l'évolution par la sélection naturelle, et les algorithmes d'intelligence en essaim qui, de la même manière que les algorithmes évolutionnaires, proviennent d'analogies avec des phénomènes biologiques naturels.

3.2.1 L'algorithme par colonies d'abeilles

L'optimisation par colonies d'abeilles artificielles (ABC: Artificial Bee Colony) est une nouvelle métaheuristique qui a enrichi le nombre des méthodes d'optimisation basées sur l'intelligence par essaim. Elle a été proposée en 2005 par Karaboga [2].

L'algorithme par colonies d'abeilles est comme suit:

Algorithme 2.4 L'algorithme de colonies d'abeilles

Début
Initialiser une population de N solution ;
Evaluer les N solutions ;
Cycle=1 ;
Tant que cycle<=MCN **faire**
Construire une nouvelle solution v_i pour chaque ouvrière i
Sélectionner les ouvrières ;
Calculer les valeurs de probabilités P_i pour les solutions x_i en
Construire la nouvelle solution de chaque abeille spectatrice à partir de la solution x_i sélectionnée en fonction de la probabilité p_i ;
Evaluer les nouvelles solutions ;
Sélectionner les spectatrices ;
Déterminer les solutions à abandonner par les scoutes si elles existent et les remplacer par des solutions aléatoires;
Enregistrer la meilleure solution trouvée.
Cycle= Cycle+1 ;
Fin Tant que
Retourner la meilleure solution ;
Fin

L'algorithme ABC s'inspire du modèle naturel du comportement des abeilles mellifères lors de la recherche de leur nourriture. Le processus de recherche de nourriture chez les abeilles est fondé sur un mécanisme de déplacement très efficace. Il leur permet d'attirer l'attention d'autres abeilles de la colonie aux sources alimentaires trouvées dans le but de collecter des ressources diverses. En fait, les abeilles utilisent un ensemble de danses frétillantes comme moyen de communication entres elles. Ces danses permettent aux abeilles de partager des informations sur la direction, la distance et la quantité du nectar avec ses congénères. La collaboration et la connaissance collective des abeilles de la même colonie sont basées sur l'échange d'information sur la quantité du nectar dans la source de nourriture trouvée par les

différents membres. Des études sur le comportement de danses frétillantes des abeilles ont montré.

- direction des abeilles indique la direction de la source de nourriture par rapport au soleil.
- L'intensité de la danse indique la distance de la source de nourriture.
- La durée de la danse indique la quantité du nectar dans la source de nourriture trouvée.

Dans un algorithme d'optimisation par colonies d'abeilles, une source de nectar correspond à une solution possible au problème à traiter. La colonie d'abeilles artificielle est composée de trois types d'abeilles: les ouvrières, les spectatrices et les scoutes.

3.2.2 L'algorithme génétique

Les algorithmes génétiques (Genetic Algorithm GA) sont des algorithmes d'optimisation inspirés de la théorie de l'évolution des espèces de Charles Darwin. Les premier travaux de John Holland remontent aux années 1960 et ont trouvé un premier aboutissement en 1975 avec la publication de « *Adaptation in Natural and Artificial Systems* ». C'est cependant l'ouvrage de David Goldberg qui a largement contribué à développer les algorithmes génétiques . Un algorithme génétique est basé sur une population d'individus dont chacun est une solution candidate du problème. Chaque solution doit être codée. Cette représentation codée est appelée chromosome, et est composée de gènes. Le degré d'adaptation d'un individu à l'environnement est exprimé par la valeur de la fonction coût (fonction objectif) correspondante. La taille de la population reste constante tout au long de l'algorithme génétique. La recherche de la solution est réglée par trois opérateurs qui sont appliqués successivement. La phase de coopération est gouvernée par un opérateur de sélection et un opérateur de croisement alors que la phase d'adaptation individuelle fait appel à un opérateur de mutation. La création d'une nouvelle génération est obtenue par itération de l'algorithme génétique qui va créer de nouveaux individus et en détruire d'autres (mécanisme de sélection naturelle) ce qui permet le renouvellement de la population (l'ensemble des solutions courantes). L'exploration de l'espace de recherche est alors réalisé par les opérateurs de mutation et assure la diversification des individus de la population (et donc des solutions). L'exploitation, quant à elle, est assurée par les opérateurs de croisement, qui recombinent les solutions, afin de les améliorer en conservant leurs meilleures caractéristiques [9].

- **Principe de l'algorithme**

Les algorithmes génétiques sont la combinaison de deux domaines la biologie et l'informatique. Pour définir le fonctionnement d'un algorithme génétique, on commence par définir les mots techniques utilisés :

Genèse: c'est la première phase de l'algorithme, il s'agit d'une population initiale de taille N.

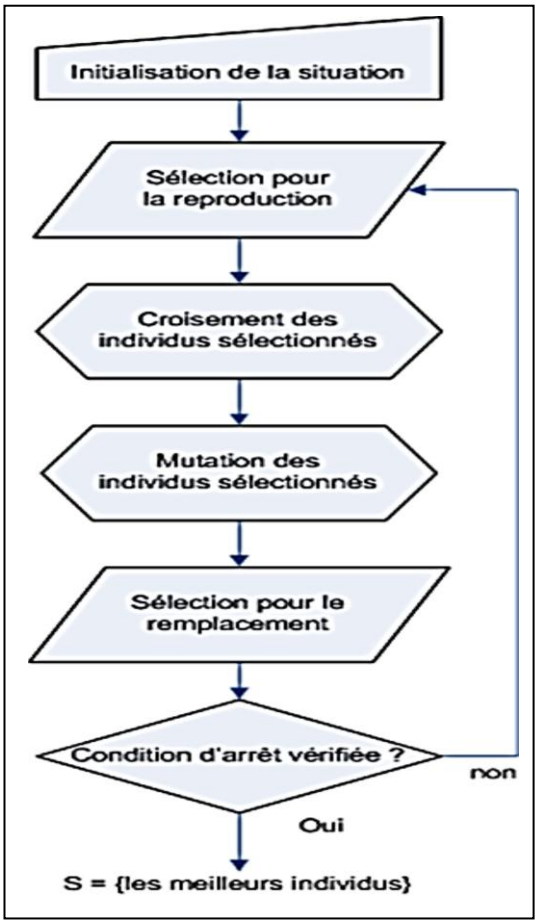
Chromosome : c'est une chaîne représentant les caractéristiques de l'individu.

Phénotype : c'est un ensemble de paramètres ou une structure décodée.

Evaluation : c'est la phase de calcul de la fonction De fitness.

Sélection : c'est le choix des individus qui vont se reproduire. Croisement : c'est la phase de production des descendants.

Mutation : c'est la modification d'un chromosome dans le but d'améliorer les caractéristiques de l'individu.



- **L'algorithme se compose de cinq étapes**

1. Initialisation de la situation.

2. Sélection pour la reproduction

3. Croisement des individus sélectionnés

4. Mutation des individus sélectionnés

5. Sélection pour le remplacement

6. Si la condition d'arrêt est vérifiée STOP,

S = {les meilleurs individus} Sinon retour à l'étape (2)

Figure 2.4 Architecture générale d'un algorithme génétique

- **Types de codage du chromosome**

Les individus de la population doivent être codés selon une représentation spécifique. Le choix d'adopter un tel codage ou un autre est une question qui dépend des caractéristiques du problème posé. Chaque paramètre d'une solution du problème traité est assimilé à un gène. Parmi les techniques les plus fréquemment utilisées pour coder les individus, on distingue :

Le codage en binaire

Dans ce type de codage, pour un individu on code ses variables et on les concatène par exemple, la chaîne binaire 1000| 0110| 1101, correspond à un individu défini par 3 variables (8, 6, 13) en codage binaire naturel sur 4 bits chacune.

C'est le codage le plus utilisé pour plusieurs raisons : pour des raisons historiques, ce codage a été utilisé par J. Holland et ses étudiants ; plusieurs résultats théoriques sont basés sur ce codage, et il est facile de mettre en place les opérateurs génétiques avec ce codage .Ce pendant si la longueur de la chaîne augmente la performance de l'algorithme diminuera.

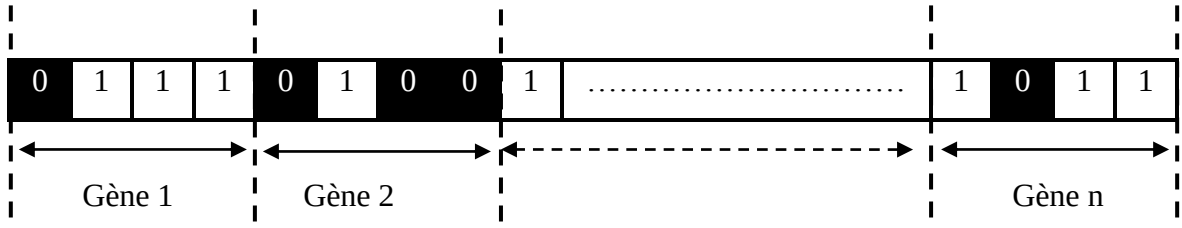


Figure 2.5 Structure d'un chromosome en codage binaire.

Le codage réel

Il sagaie de concaténation des variables xi d'un individu x. par exemple un individu x (25, 31, 8) est codé 25| 31| 8 , ce codage est plus précis que le codage binaire, l'espace de recherche est le même que l'espace du problème, l'évaluation de la fonction coût est plus rapide ; mais son alphabet est infini et il a besoin d'opérateurs appropriés.

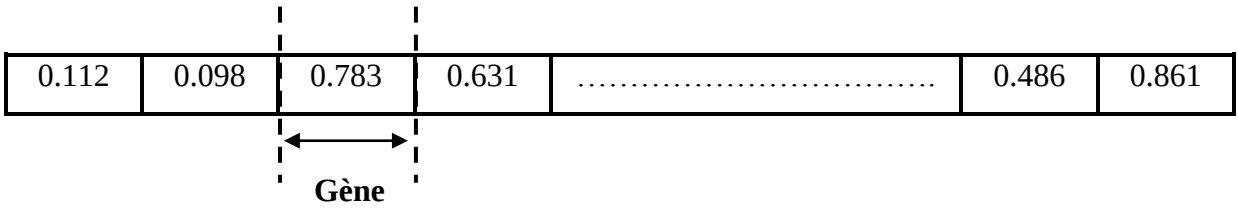


Figure 2.6 Structure d'un chromosome en codage en nombres réels.

- **Opérateurs de reproduction**

Les algorithmes génétiques sont basés sur un phénomène naturel : l'évolution. Plus précisément, ils supposent, qu'a priori, deux individus adaptés à leur milieu donnent, par recombinaison de leurs gènes, des individus mieux adaptés. Pour ce faire, trois opérateurs sont à disposition : la sélection, le croisement et la mutation, plus un opérateur optionnel, l'élitisme.

Sélection

La sélection sert à choisir dans l'ensemble de la population les individus qui participeront à la reproduction. Plusieurs méthodes existent et sont, généralement, basées sur la théorie de Darwin. Ainsi les meilleurs individus ont plus de chance de survivre et de se reproduire.

Roulette

Cette méthode exploite la métaphore d'une roulette de casino. La roue est divisée en autant de secteurs que d'individus dans la population. La taille de ces secteurs est proportionnelle à l'adaptation de chaque individu. En faisant tourner la roue, l'individu pointé à l'arrêt de la boule est sélectionné. Les individus les mieux adaptés ont donc plus de chance d'être tirés au sort lors du déroulement du jeu figure 2.6.).

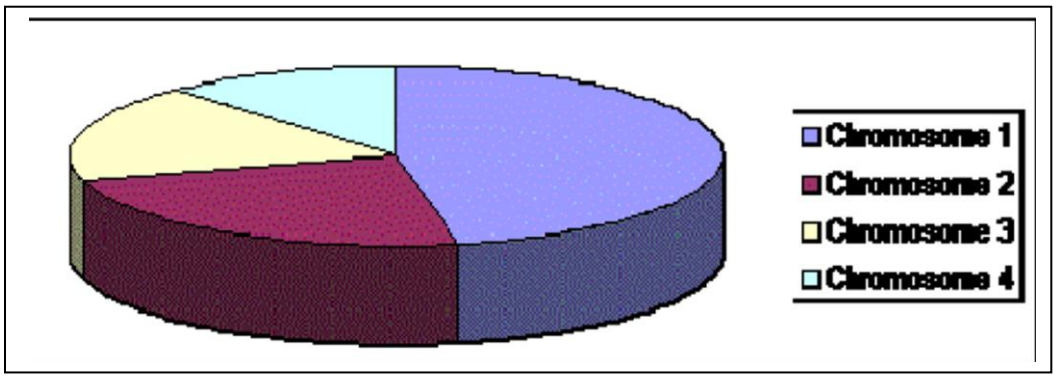


Figure 2.7 Schéma de la sélection par la roulette.

- **Elitisme :**

A la création d'une nouvelle population, il y a de grandes chances que les meilleurs chromosomes soient perdus après les opérations d'hybridation et de mutation. Pour éviter cela, on utilise la méthode d'élitisme. Elle consiste à copier un ou plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on génère le reste de la population selon l'algorithme de reproduction usuel. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleures solutions.

- **Opérateurs de croisement**

Le croisement consiste à générer deux enfants à partir de deux parents avec une probabilité Px appelé probabilité de croisement ceci dans le but d'enrichir la diversité de la population .Les opérateurs sont de deux types ; croisement par point et croisement bipoint.

opérateur à un point

Il consiste à diviser chacun de deux parents en deux parties à la même position, choisie au hasard. Le premier enfant est composé de deux parties des deux parents. La première partie est celle du premier parent et la deuxième partie est celle du deuxième parent et le deuxième

enfant est composé de deux parties, une première partie du deuxième parent et la deuxième partie du premier parent.

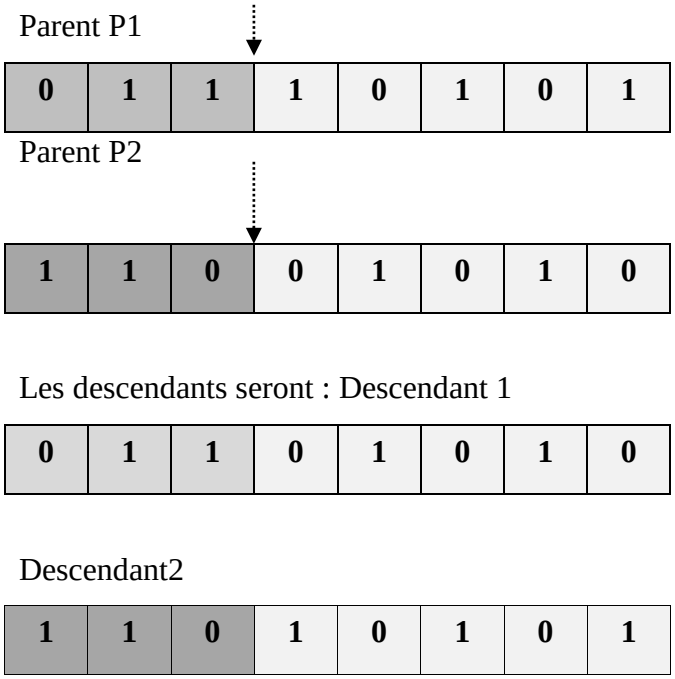
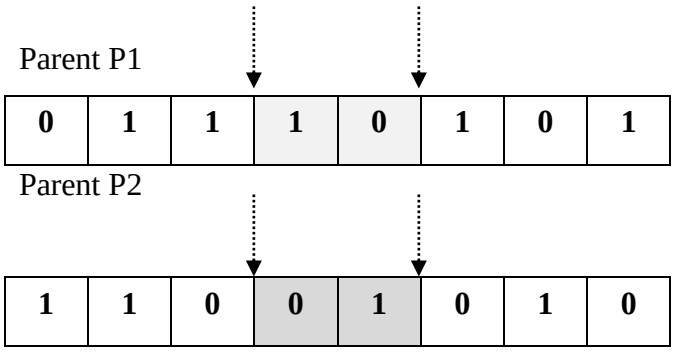


Figure 2.8 Croisement 1-point.

Opérateur à deux points

Cette méthode consiste à fixer deux positions, le premier enfant sera la copie du premier parent en remplaçant sa partie entre les deux positions par celle du deuxième parent. La même opération sera appliquée pour déterminer le deuxième.

enfant en inversant les rôles du premier parent et du deuxième parent.



Les descendants seront : Descendant 1

0	**1**	**1**	**0**	**1**	**1**	**0**	**1**

Descendant2

1	**1**	**0**	**1**	**0**	**0**	**1**	**0**

Figure 2.9 Croisement 2-points.

Opérateur de mutation

Conformément à la mutation naturelle, Cet opérateur permet de changer la valeur d'un chromosome dans le but d'améliorer les caractéristiques de l'individu. Elle permet à l'algorithme génétique d'explorer efficacement l'espace de recherche. Il garantit aussi une susceptibilité d'atteindre la plupart des points du domaine réalisable.

Individus I

0	**1**	**1**	**0**	**1**	**1**	**0**	**1**

Apres mutation

0	**1**	**1**	**1**	**1**	**1**	**0**	**1**

Figure 2.10 Mutation

Les avantages

- Ils sont adaptables aux plusieurs types des problèmes.
- Robustes.
- Facile à implémenter.
- Facile à hybrider.
- Facile à paralléliser.

Les inconvénients

- Pas de garantie de convergence.
- Temps de calcule important (si la taille de population est grande).

3.2.3 L'algorithme de colonies de fourmis

Dorigo, Maniezzo et al proposent l'optimisation par colonies de fourmis (OCF) (Ant Colony Optimization ACO) [9].

L'algorithme de colonie de fourmis est une métaheuristique basée sur le comportement collectif des fourmis. Un parallèle a été établi entre la recherche de nourriture et la résolution

de problèmes complexes. En effet, pour rechercher la nourriture, les fourmis se dispersent (elles n'ont qu'une vision locale de l'environnement) et après l'avoir trouvée, elles retournent au nid en laissant des phéromones pour marquer leur passage. Les phéromones sont des substances volatiles que les fourmis utilisent pour communiquer. Les fourmis ainsi attirées répandront à leur tour des phéromones et le chemin entre le nid et la zone de nourriture sera renforcé. Ce pendant, grâce au phénomène d'évaporation des pistes de phéromones (si le trajet n'est pas ou peu emprunté, les phéromones tendent à disparaître), elles aboutissent assez vite à un trajet très marqué, représentant le plus court chemin, comparé à d'autres directions qui disparaissent (Figure 2.11).

Au début de l'expérience A la fin de l'expérience

Figure 2.11 Expérience de sélection de branches par une colonie de fourmis.

L'algorithme par colonies Colonies de fourmis est comme suit:

Algorithme 2.5 L'algorithme Colonies de fourmis

Tant que (la condition d'arrêt n'est pas atteinte)
Tant que (L'état final n'est pas atteint)
Pour (chaque fourmi)
Choisir l'état suivant en fonction de la mémoire et de l'environnement local
Mettre à jour les phéromones sur l'arc choisi.
Fin Pour
Fin Tant Que
Pour (chaque fourmi)
Evaluer la solution obtenue
Fin Pour
Mise à jour globale des phéromones
Fin Tant que
Afficher la meilleure solution trouvée

Les avantages

- Rapidité de la méthode.
- Nouvelle méthode à trouver des solutions acceptables tout en évitant des convergences prématurées.
- Robuste et basée sur une population d'individus.

Les inconvénients

- Complexe à mettre en place et son paramétrage est subtile.
- Coût relativement élevé de la génération des solutions.

3.3 Méthodes hybrides

Une méthode hybride est une méthode de recherche constituée d'au moins de deux méthodes de recherche distinctes. Elle consiste à exploiter les avantages respectifs de plusieurs méthodes en combinant leurs algorithmes suivant une approche synergétique. Une méthode hybride peut être mauvaise ou bonne selon le choix et les rôles de ses composants. Pour définir une méthode hybride efficace, il faut savoir caractériser les avantages et les limites de chaque méthode. Par exemple, les algorithmes génétiques(GA) sont très

performants lorsqu'il s'agit d'explorer l'espace de recherche, mais ils s'avèrent ensuite incapable d'exploiter efficacement la zone vers laquelle la population des solutions converge. Il est alors plus intéressant d'utiliser dans ce stade une autre méthode permettant une bonne exploitation comme par exemple le recuit simulé ou une autre heuristique d'amélioration. Il faut souligner qu'il faut être prudent sur le choix des méthodes à hybrider ainsi sur le problème de multiplication des paramètres.

Les auteurs ont clairement démontré l'effet bénéfique de l'intégration d'une méthode de descente à l'intérieur d'une méthode évolutive. D'autres travaux ont montrés que lors de la résolution du problème de coloration des graphes, les meilleurs résultats sont obtenus en hybridant une méthode évolutionnaire avec une méthode de recherche locale [9].

Actuellement, les approches hybrides gagnent en popularité car ce type d'algorithme produit généralement les meilleurs résultats pour plusieurs problèmes d'optimisation combinatoire . En effet , les approches hybrides ont permis d'obtenir de bons résultats dans une grande variété de problèmes théoriques d'optimisation combinatoire tels le problème du voyageur de commerce, le problème de coloration de graphe, le problème d'affectation quadratique , etc . Les premières approches hybrides proposées ont combiné des métaheuristiques à base de populations (algorithmes génétiques (GA)) avec des métaheuristiques à solution unique (recuit simulé (SA) ou recherche tabou (TS)).

Plusieurs approches hybrides combinant les métaheuristiques et les méthodes exactes ont été proposées dans la littérature. Etant donné que les méthodes exactes se limitent généralement à de petites instances pour les problèmes d'optimisation combinatoire difficiles, tel que décrit précédemment, l'hybridation de métaheuristiques avec les méthodes exactes peut devenir une alternative très intéressante car les deux méthodes ont des particularités bien différentes qui peuvent être associées pour produire de meilleurs résultats. Donc Plusieurs types d'hybridations sont possibles, nous citons :

- Hybridation de méthodes exactes-exactes
- Hybridation de méthodes heuristiques-exactes
- Hybridation de méthodes heuristiques-heuristiques

4. Motivation de l'hybridation

Malgré le succès des métaheuristiques pour résoudre les problèmes complexes, plusieurs problèmes ont été rencontrés, à savoir :

- la présence de bruit lors de l'évaluation des solutions.
- la distinction entre une solution optimale locale et une autre globale est parfois difficile à percevoir.
- le problème de la convergence prématurée
- le problème d'ajustement des paramètres.

L'hybridation de méthodes bio-inspirées a connu une grande popularité et a permis de bénéficier des points forts de chacune de ces méthodes et de surmonter leurs limites. L'hybridation a permet d'avoir un compromis entre l'exploration et l'exploitation de l'espace de recherche des solutions. Pour avoir une bonne exploitation, un algorithme est utilisé pour localiser les meilleures régions de l'espace de recherche, un autre est utilisé pour converger vers l'optimum global. L'hybridation est utilisée aussi pour optimiser les paramètres généraux. Par exemple un algorithme génétique est utilisé pour trouver les meilleurs paramètres d'un algorithme d'optimisation par colonie de fourmis [9].

On peut conclure que l'hybridation de métaheuristiques est la voie la plus prometteuse pour l'amélioration de la qualité des solutions dans beaucoup d'applications réelles. Ainsi, Le choix d'une approche hybride devient aujourd'hui déterminant pour obtenir de meilleures performances lors de la résolution des problèmes complexes.

5. Conclusion :

Nous pouvons, pour l'instant, remarquer que quelque soit la méthode, elle manque d'efficacité malgré certains points forts. Les inconvénients qui jaillissent peuvent être palliés par l'injection d'une approche différente ou par la combinaison de différentes approches nommée approche hybride ou méthode hybride.

Cette méthode ou approche est celle que nous avons adoptée dans le chapitre suivant traitant de la méthode hybride.

CHAPITRE 3

METHODE PROPOSEE

1. Introduction

Ce chapitre est spécialement réservé au problème de l'affectation quadratique qui est le sujet essentiel de notre présente étude. En premier temps, on va présenter le problème de l'affectation quadratique (QAP). Ensuite on va détailler les composants essentiels de l'algorithme hybride proposée nommée QAP-GASA, cette approche est l'hybridation entre les algorithmes génétique (GA) et l'algorithme de recuit simulé (SA). Enfin une série d'expérimentations commentée sera décrite pour évaluer la performance de la méthode développée.

2 Approche pratique du problème

2.1 Un peu d'histoire sur le problème d'affectation quadratique

Pour notre étude, le problème d'affectation quadratique QAP est traité en tant que problème formel. Cependant, la formulation mathématique du QAP provient de la modélisation de certains problèmes de placement qu'on rencontre dans le domaine de la production industrielle. Très souvent, des industriels se demandent comment agencer les différentes unités d'un site de production pour minimiser le coût inhérent au transport des produits à l'intérieur du site. « Faut-il implanter l'unité d'extrusion à côté du secteur d'assemblage ou plutôt à côté du magasin des matières brutes? ». Avant l'ère informatique, on utilisait des méthodes graphiques pour concevoir la disposition d'une usine mais ces méthodes ne permettent pas une subdivision du site en plus d'une dizaine d'unités. Dans les années cinquante et soixante, quelques chercheurs se sont intéressés à la méthodologie du placement pour répondre aux besoins des industriels. Ils ont proposé une méthode heuristique qu'ils ont implantée sur un ordinateur. Pour notre étude, nous avons appréhendé le QAP à travers une définition formelle; mais nous Pavons presque toujours imaginé comme un problème d'optimisation du placement d'éléments sur des emplacements [13].

2.2 Les instances du problème d'affectation

Presque toutes les instances du QAP que nous avons traitées dans notre étude sont des instances « standard », issues de la bibliothèque QAPLIB Cette bibliothèque, reconnue par la communauté œuvrant sur le QAP, facilite grandement la comparaison des méthodes de résolution , c'est pourquoi nous l'utilisons. Les instances QAPLIB sont d'origines variées: ce sont des instances générées par des programmes informatiques et/ ou des instances modélisant des problèmes réels. Les instances générées ont été produites à l'aide de générateurs de nombres pseudo aléatoires; cependant, certains auteurs ont introduit des contraintes qui leur procurent une nature non uniforme, imitant, pour certains, la nature des instances réelles. On trouve des instances réelles du QAP dans une grande variété de domaines dont voici quelques exemples: l'architecture (Elshafei) , le sport: élaboration d'une équipe de relais (Heffiey), l'imagerie : trames de niveaux de gris (Taillard), le câblage en électronique (Steinberg), la conception de claviers et baies de contrôle (Burkard), la conception de turbines hydrauliques (Laporte) , l'analyse de données statistiques ou l'ordonnancement de chaînes de production parallèles (Geoffrion). Nous détaillons, ci-après, quelques uns de ces problèmes qui ont été modélisés par des QAP. Tous ces exemples peuvent être imaginés comme un problème de placement de n éléments sur n sites

pour lequel les données sont les distances entre les sites et les flux entre les éléments [13].

2.3 Présentation du problème d'affectation quadratique

Le problème d'affectation se présente fréquemment en recherche opérationnelle. Il consiste à réaliser une bijection des éléments *i* d'un ensemble *I* sur ceux *J* d'un ensemble grand *J, de* même cardinalité, de telle manière qu'une certaine fonction de coût, dépendant du choix des couples (i,j) soit minimale . Lorsque cette fonction de coût est linéaire, c'est un problème classique et sa solution est donnée par un algorithme polynomial [en 0 (n^4), si $|I| = |J| = n]$: la méthode hongroise [13] .

Le problème d'affectation quadratique, ou en anglais Quadratic Assignment Problem (QAP), est un problème NP-difficile. Il représente un défi important pour différents domaines.

Ce problème est connu pour ses multiples applications en chimie, transport, industrie et plusieurs autres disciplines. Nous pouvons trouver quelques travaux connus de l'application du QAP . (Kadluczka & Wala, 1995; Bhaba et al., 1998; Polak, 2005; Zhang et al., 2005; Duman & Or, 2007, Ulutas & Konak, 2012; Wu & Hao, 2015). Le QAP a été introduit par Koopmans & Beckmann (1957) pour modéliser l'affectation des ressources dans des localisations. L'objectif est de trouver le coût minimum d'affectation en considérant deux matrices de variables. La première matrice est celle des flux entre les ressources à affecter, et la deuxième matrice est celle des distances entre les localisations. Ce problème est formulé dans l'équation 3.1

$$\min_{p \in P} z(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)} \qquad (3.1)$$

où f et d sont les matrices des flux et des distances, respectivement ; P est l'ensemble de toutes les permutations des n localisations possibles ; pi est la localisation assignée à la ressource i. L'objectif est de minimiser z(p), qui représente le coût total des affectations de la permutation p.

figure 3.1 présente un exemple simple du problème du QAP. Dans cet exemple, les usines i et j représentent les ressources et les sites h et l représentent les localisations. Fij est le flux entre les deux usines et Dlh est la distance entre les deux sites. L'objectif est de trouver l'affectation optimale des usines dans les sites de façon à minimiser le produit des flux multiplié par les distances.

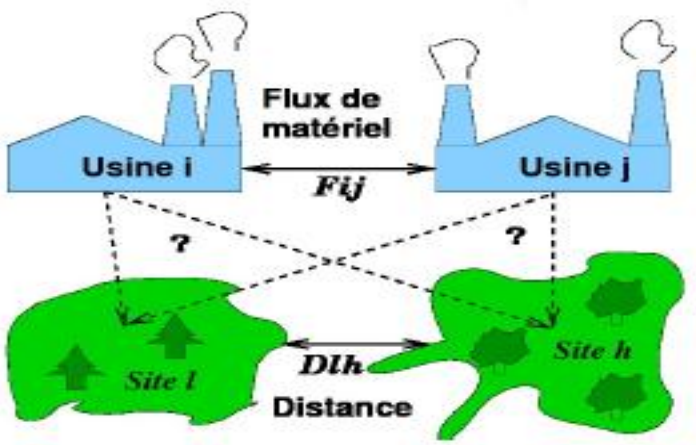


Figure 3.1 Un exemple simple du QAP

3 Aspects formels du QAP

3.1. Formalisation mathématique du QAP

Une instance du QAP est caractérisée par sa taille n et deux matrices carrées de réels. Lorsque le QAP est considéré comme un problème de placement de n éléments sur n emplacements, ces matrices sont désignées comme suit :

- $(d)_n$, la matrice des distances entre deux emplacements;
- $(f)_n$, la matrice des coûts des flux de produits entre deux éléments;

Bien que le problème d'affectation s'énonce simplement: « Trouver un placement dont le coût est minimal », de nombreuses formulations mathématiques du QAP ont été proposées.

Pour notre étude, nous utilisons la formulation la plus répandue, directement dérivée de la proposition originale de Koopmans et Beckmann [12] .

3.1.1 La formulation de Koopmans-Beckmann

En 1957, Koopmans et Beckmann proposent une des toutes premières formulations du QAP. Ils représentent un placement possible par un tableau où les éléments sont en ligne et les emplacements en colonne. Une croix dans ce tableau indique l'affectation d'un élément à un emplacement[12]. Pour le QAP, chaque élément étant associé à un et un seul emplacement, il ne peut pas y avoir deux croix sur une même ligne ou sur une même colonne (Figure3.2)

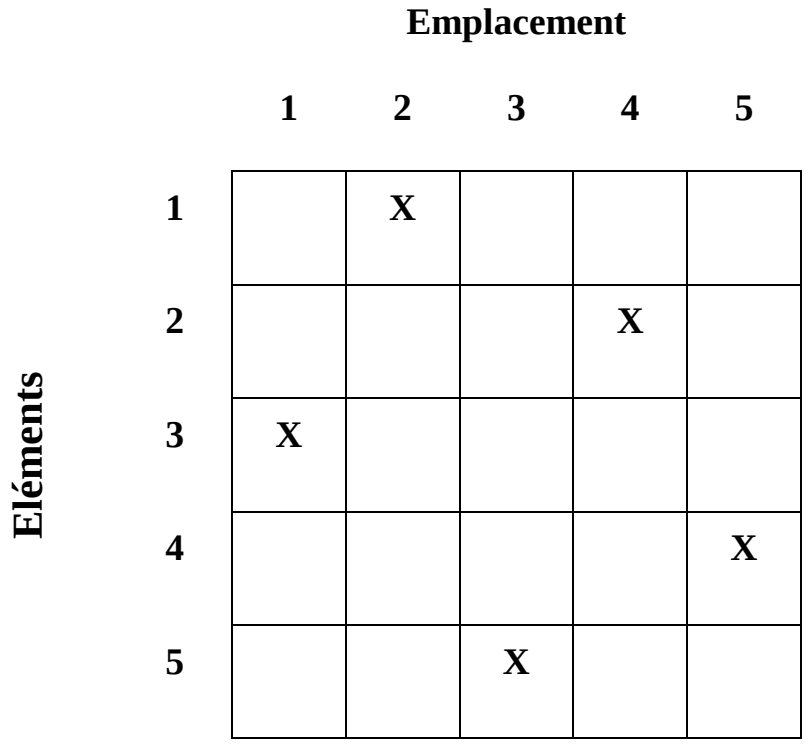


Figure 3.2 Représentation matricielle d'un placement**.**

Ainsi, en appelant i et j, les indices respectifs des éléments et des emplacements; en posant $X_{ij} = 1$ si l'élément i est placé en jet $X_{ij} = 0$ sinon; un placement possible est représenté par la matrice de permutation $(x)_n$ caractérisée par:

$$\begin{cases} \sum_{i\in I} x_{ij=}1 & ,\forall j \in J \\ \sum_{j\in J} x_{ij=}1 & ,\forall i \in I \\ x_{ij} \in \{1,0\} & i \in I, j \in J, \ avec \ |I| = |J| = n \end{cases} \qquad (3.2)$$

Le problème revient à trouver la matrice $(x^*)_n$ qui minimise la fonction d'évaluation φ définie comme suit,

$$\min[\, Q(x)] = \sum_{i\in I}\sum_{j\in J}\sum_{k\in I}\sum_{l\in J} f_{ik}d_{jl}x_{ij}x_{kl} \qquad (3.3)$$

où P_n est l'ensemble des matrices de permutations de taille n.

3.1.2 La formulation retenue

À partir de la proposition de Koopmans et Beckmann, on obtient une nouvelle formulation en « compactant » la représentation matricielle d'un placement. En effet, en indiquant les numéros de chaque élément à la place de chaque croix, puis en sommant les lignes on a une représentation du placement sous la forme d'un vecteur de taille n comme le montre la figure 3.3.

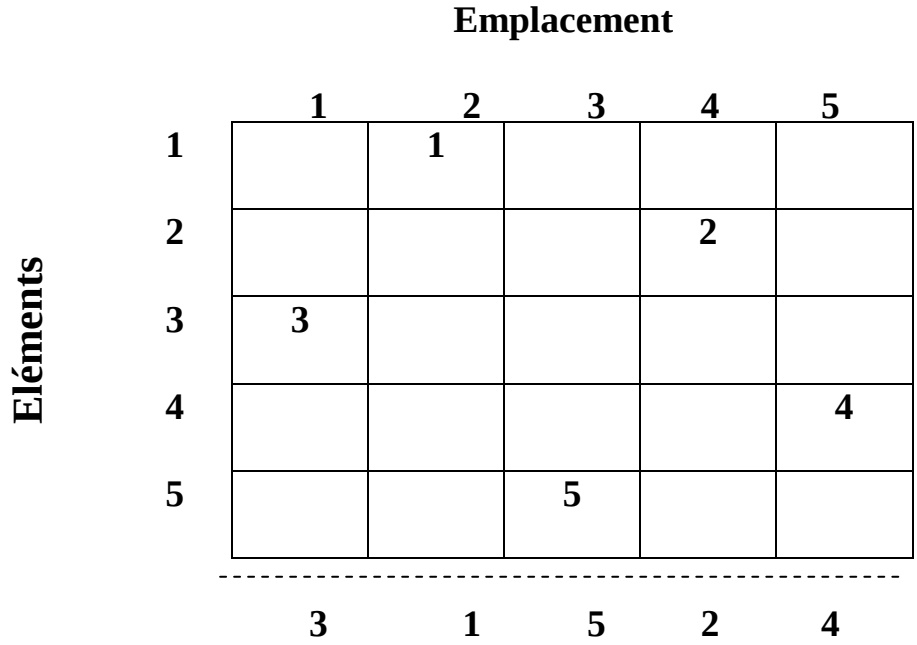


Figure 3.3 Représentation vectorielle d'un placement.

Ainsi, un placement est représenté par un n-uplet P qui est un élément de P_n, qui représente l'ensemble des permutations de taille n. La formulation du QAP revient alors à trouver P_* tel que :

$$Q(p_*) = min_{p \in} p_n \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)} \qquad (3.4)$$

4. Exemple simple sur le problème d'aménagements d'usine

Dans l'exemple qui suit (figure 3.4) on dispose de six zones et d'un ensemble six objets. Le but est d'affecter l'ensemble des objets à l'ensemble des zones tout en minimisant la fonction objective qui présente le coût total engendré.

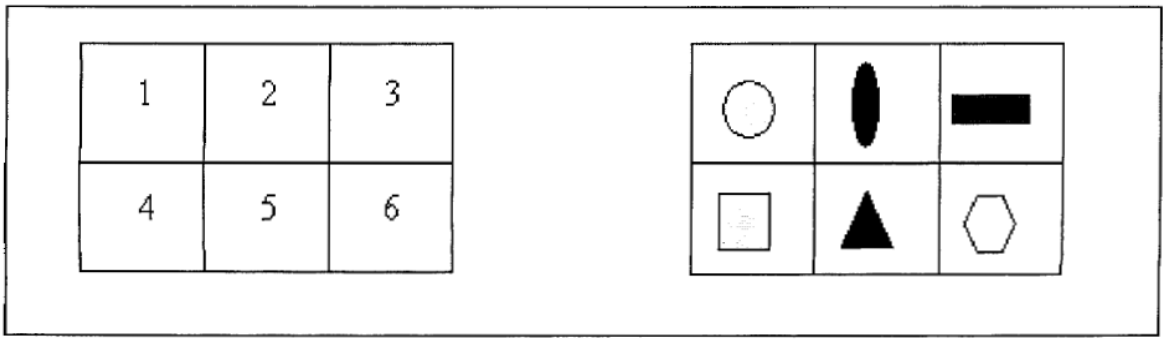


Figure 3.4 Exemple d'emplacements et affectation des équipements aux emplacements

Soient:

- *D(n* x *n)* La matrice telle que *d(i, J)* représente la distance entre la location *i* et la Location *j*.
- *F(n* x *n)* La matrice de dimension *(n* x *n*)telle que *F(k,l)* représente le flux entre les objets *k* et *l*.

Le problème peut alors s'écrire sous la forme suivante:

La Fonction de objectif:

$$Min \quad L = \sum_{i=1}^{n-1} \sum_{j=j+1}^{n} f(i,j) \times d(p(i),p(j)) \qquad (3.5)$$

Problème d'aménagement d'usine est formulé pour trouver l'arrangement le plus efficace des départements comme expliqués dans les schémas suivants :

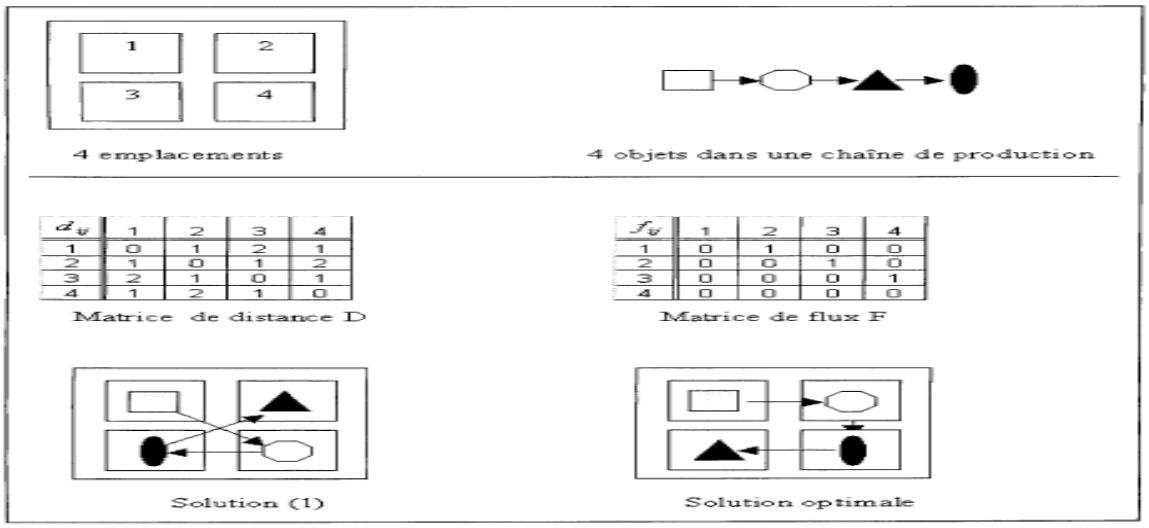


Figure. 3.5. Un exemple simple de l'affectation quadratique

Le coût associé à la solution ***(1)*** est :

$$L = 2+1+2 = 5 \quad (3.6)$$

Alors que le coût de la solution *(2)* est :

$$L = 1+1+1 = 3 \quad (3.7)$$

Ceci démontre que la solution *(2)* est l'une des solutions optimales et alors le meilleur arrangement

5. Complexité du problème d'affectation quadratique

Du point de vue de la complexité algorithmique, le problème d'affectation quadratique (QAP) apparaît comme l'un des problèmes les plus difficiles à résoudre. En effet, de nombreux problèmes NP- difficiles bien connus peuvent être formulés sans réelle difficulté, comme des cas particuliers du QAP. Parmi ceux-là, citons le problème du voyageur de commerce (TSP- Traveling Salesman Problem),le problème de partitionnement de graphe (GP-Graph Partitionning) ou encore le problème de la clique maximale (MCP -Maximum Clique Problème). En 1976, Sahni et Gonzales ont montré que le QAP est NP-difficile. De plus, ils ont montré que trouver une approximation est aussi NP-difficile. Cependant, pour quelques cas très particuliers de QAP, la complexité devient polynômiale (comme le problème d'affectation linéaire).

6. Domaines d'applications

Le problème d'affectation quadratique a de très nombreuses applications en pratique, tant en informatique, qu'en productique électronique ou architecture, mais il est très délicat à résoudre.

En économie : le problème fut posé pour la première fois par Koopmans et Beckmann en 1957, sous le nom anglais de « indivisible ressources allocation problème ». Il s'agit d'implanter diverses unités de production à des emplacements différents déjà déterminés. Des quantités fixées de produits (marchandises, matières premières,...) doivent circuler entre ces usines, et leur coût de transport dépend de la distance parcourue [3].

L'implantation des usines cherchée est évidemment celle qui assurera un coût de transport minimal :

Soit I est l'ensemble des n usines

Soit J l'ensemble des n emplacements possibles

X_{ij} = 1 si l'usine i est implantée en J =0 sinon;

f_{ik} la quantité de produits à transporter de l'usine i à l'usine k;

d_{jk} le coût de transport d'une unité de produits de l'emplacement j à l'emplacement *l*

Dans l'industrie des automatismes et ordinateurs : le problème se pose lors de la conception de la plaquette qui doit supporter les circuits intégrés. Pour obtenir un bon tracé de connexions, il faut chercher à implanter les modules de circuits intégrés sur cette plaquette de manière à minimiser la longueur totale des liaisons [3].

Soit I l'ensemble des modules;

Soit J l'ensemble d'emplacements possibles pour ces modules sur la plaquette;

$$Xij = \begin{cases} 1 \text{ si le module i est placé en plaquette j} \\ 0 \quad \text{sinom} \end{cases}$$

f_{ik} : le nombre de liaisons du module i au module k;

d_{jk} : la longueur d'une liaison, c'est-à-dire la distance euclidienne ou rectangulaire entre l'emplacement j et l'emplacement I .

Lors de la conception des projets architecturaux : tels que planning de bureaux, réorganisation de services dans l'hôpital..., l'architecte peut répertorier les emplacements où peuvent se dérouler différentes activités sur un site, puis se fonder pour implanter ces activités sur un critère de contiguïté. Plus certaines activités apparaissent comme complémentaires, c'est-à-dire, plus les flux de personnel, information ou matériel, circulant entre elles, sont importants, plus ces activités devront être voisines sur le futur plan. L'architecte cherche alors le « schéma de plan » qui minimise les distances de circulations entre activités [3].

Soit I l'ensemble des activités devant se dérouler dans le projet;

Soit J l'ensemble des emplacements possibles;

$$X_{ij} = \begin{cases} 1 \text{ si l}'\text{activite i est implantée dans emplacement j} \\ 0 \quad \text{sinom} \end{cases}$$

$\int_{ik}^{h}$ un nombre traduisant l'importance de la liaison entre deux activités i et k mesurée par la quantité de flux h s'écoulant entre i et k;

$\int_{jk}^{h}$ la distance sur le site mesurée à travers les axes de circulation existant pour le flux h, entre les emplacements j et I .

7. Méthode proposée

7.1 Composants de l'algorithme GA pour le problème QAP

- **Codage de la solution**

Dans notre travail, un chromosome est représenté par une permutation d'entier 1, 2, .. n, où n est le nombre de unités et des cites, c'est-à-dire la taille du problème. Par exemple :
(2, 5, 1, 4, 3, 9 , 7 , 6 , 8 , 10) est un chromosome possible pour une instance de QAP avec une taille de problème de 10. Dans ce cas, "2" de (2, 5, 1, 4, 3, 9 , 7 , 6 , 8 , 10) signifie que la flux1 est placée à cite 2, et qu'il s'agit de ce chromosome. De même, "5" signifie que cite flux 2 est placée au cite 5 et ainsi de suite. n éléments sur n emplacements

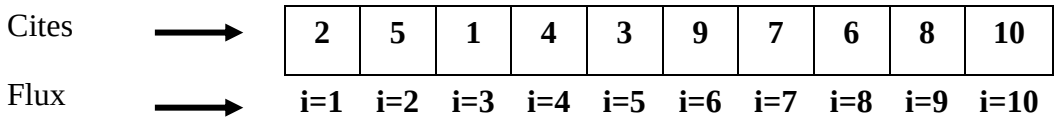


Figure 3.6. Représentation de la solution

- **Etape d'initialisation**

Une population est un ensemble de chromosomes. Dans notre algorithme, un chromosome est représenté par une permutation d'entiers. Une permutation est générée aléatoirement et ne sera insérée que si elle n'existe pas encore dans la population. Il s'agit d'éviter la duplication de chromosome dans la population initiale et donc d'améliorer la diversité.

- **Etape de Sélection**

Les chromosomes (parents) sont sélectionnés dans la population pour être combinés afin de produire de nouveaux chromosomes (enfants), pour appliquer des opérateurs génétiques. Ici, nous utilisons la méthode de roulette pour sélectionner les parents, chaque chromosome a une chance d'être sélectionné proportionnellement à sa performance. Dans la sélection de la roue de roulette, les individus sont donnés une probabilité d'être sélectionnés qui est directement proportionnelle à leur forme physique.

- **Operateur de Croisement**

Le rôle d'un opérateur de croisement est de combiner des éléments provenant de deux chromosomes parents pour générer deux chromosomes enfants. Nous utilisons le croisement à deux points sélectionnées on les deux basé sur la position, le schéma proposé est illustré à la Figure 3.7.

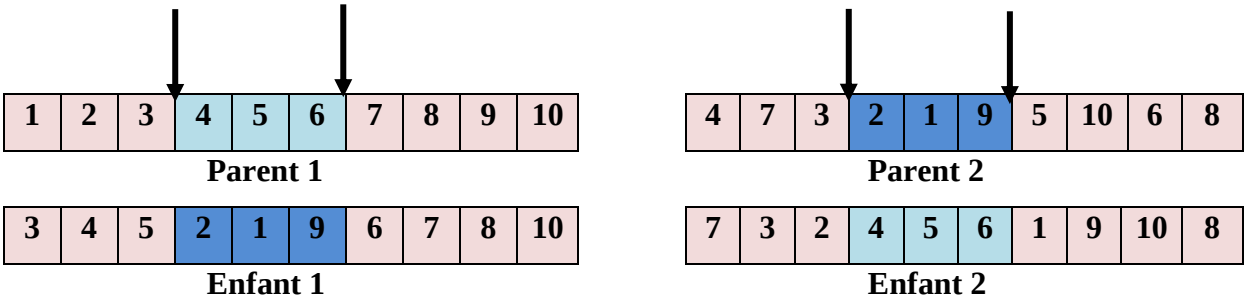


Figure 3.7 Croisement de deux individus à deux point.

- **Operateur de mutation**

Le rôle d'un opérateur de mutation est d'assurer et de maintenir la diversité d'une population afin que d'autres opérateurs puissent continuer à travailler. Pour cela, il modifie un ou plusieurs gènes avec une probabilité égale au taux de mutation.

La procédure de mutation utilisée est l'algorithme de recuit simulé (SA). Cet algorithme est une technique qui permet de modéliser et de référencer le processus de recuit des métaux en fusion lors du refroidissement.

- **Remplacement**

La stratégie de remplacement utilisée ici consiste à sélectionner les meilleurs chromosomes de la population actuelle et leurs descendants. Ils formeront une nouvelle population pour survivre dans la prochaine génération, cette étape est très importante pour construire la prochaine génération de l'algorithme génétique.

- **Critère d'arrêt**

Le critère d'arrêt utilisé est le nombre de génération égal à $Iter_{max}$. Après $Iter_{max}$ générations. L'algorithme génétique s'arrête et donne le meilleur chromosome qui possède un $\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{p(i)p(j)}$ minimale. Dans notre travail, nous avons essayé de prendre un nombre d'itérations $Iter_{max}$ qui a été fixée après plusieurs testes.

7.2 Composants du recuit simulé pour le problème de QAP

Le recuit simulé est inspiré par l'étude de la stabilité thermique d'un système physique. La méthode part d'une solution initiale admissible et continue l'exploration de l'espace d'états en effectuant des perturbations mineures sur la solution courante. Si la nouvelle solution obtenue est améliorée alors elle est retenue. Si elle est détériorée par rapport au critère d'optimisation alors elle est retenue avec une probabilité inversement proportionnelle au nombre d'itérations.

- **La solution initiale**

Pour notre approche, la solution initiale S_0 est le chromosome objet de la mutation. Ce dernier est retournée par l'algorithme génétique à chaque génération.

- Choisir une température de départ T_0
- A chaque itération de l'algorithme, un changement élémentaire est effectué sur la solution, cette modification fait varier l'énergie du système.

- **Génération du voisinage**

Pour générer une solution voisine à la solution courante, on choisi aléatoirement deux positions dans le chromosome courant et en change leurs positions pour obtenir une nouvelle solution.

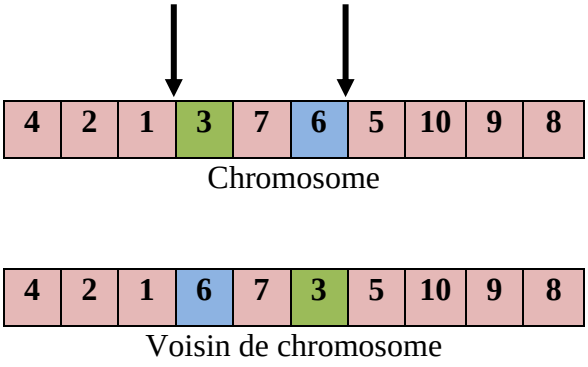


Figure 3.8 voisinage de deux positions.

- **Choix de la température**

Le choix de la température initiale dépend de la qualité de la solution de départ. Si cette solution est choisie aléatoirement, il faut prendre une température relativement élevée. Pour la mise à jour de la température, le schémas de la décroissance géométrique est choisi et la valeur de T est modifiée à chaque itération comme suit : $T_{K+1} \leftarrow T_K * \propto$. En général α = 0.9 à 0.99 . notons ici que nous avons testé notre algorithme sur plusieurs valeurs de T et nous avons retenu la valeur T=100 car elle donne le meilleure résultat.

- **Le critère d'arrêt**

un nombre maximal Itermax d'itération est aboutis

la température a atteint la valeur finale prédéfinie Tf.

pas d'amélioration de la solution après un certain nombre d'itérations

Le pseudo-code de l'algorithme de recuit simulé est donné par :

Algorithme 3.1 L'algorithme de recuit simulé

Début
Construire une solution initiale s ;
Calculer la fitness f(s) de s ;
Initialiser une valeur de la température T
Poser sbest=s ;
Tant que la condition d'arrêt n'est pas satisfaite faire
Générer une solution s' voisine de s;
Calculer $f(s')$;
Calculer $\Delta(f) = f(s') - f(s)$;
Si $\Delta(f) \leq 0$ alors
Si s< sbest alors
sbest = s'
Fin Si;
s = s' ;
Sinon
Choisir une valeur aléatoire $\theta \in U(0,1)$
Si $\theta < e^{(-\Delta(f)/T)}$ alors
s = s' ;
Fin Si
décrémenter T :T ← T*α;
Fin Tant que
Retourner sbest ;
Fin

Apres le description des différents composants des algorithmes GA et SA, le pseudo-code de l'algorithme hybride QAP-GASA est résumé comme suit :

Algorithme 3.2 Algorithme QAP-GASA:

Initialiser *Population*, *T*croi, *Tmut*, Itermax

Générer des séquences de *Psize* aléatoirement

Répéter

- Sélectionnez deux parents
- Appliquer l'opérateur de croisement avec la probabilité *T*croi
- Appliquer l'opérateur de mutation sur les enfants obtenus avec la probabilité *Tmut* en utilisant la méthode SA
- Évaluer tous les chromosomes (2*Psize*, parents et enfants) à l'aide de la fonction fitness
- Organiser les parents et les enfants par ordre croissant en utilisant leur fonction fitness
- Enregistrez les meilleurs chromosomes de population dans Bestpop et supprimez le reste des chromosomes
- Remplacer Psize par Bestpop

8. Présentation des instances considérées

Un benchmark, en anglais est un point de référence servant à effectuer une mesure. En informatique, un benchmark est un banc d'essai permettant de mesurer les performances d'un system pour le comparer à d'autres.

Parmi les problèmes de benchmark que nous avons utilisé on peut citer wil50 études par Wilhelm and T.L. Ward est une instance où la taille de problème change n = 50 ainsi que les matrices de flux et de distances, pour les instances Tai25 et Tai30 sont des problèmes de taille n=25 et n=30 traités par Taillard , pour l'instance Els19 est un problème de taille n=19 traité par A.N. Elshafei et pour les instances Esc16d et Esc32e sont des problèmes de taille n=16 et n=32 traités par Eschermann and H.J. Wunderlich.

9. Etude expérimentale

Dans cette section on veut présenter un ensemble des résultats expérimentaux pour déterminer en premier temps l'importance du l'utilisation des méthodes approchées pour résoudre ce type de problème et montrer l'importance et l'efficacité des métaheuristique, en particulier les algorithmes génétiques pour la qualité de solution trouvée.

Notons que les deux méthodes : la Méthode RS (recuit simulé) et AG (Algorithme Génétique) sont implémentées en C++ sous Windows 7 et sont testées sur un PC avec un Processeur Intel Pentium 2.40 GHz et une mémoire de 4 Go. Avec le compilateur Visual Studio 2019. Les données utilisées sont générées comme suit :

- Temp-exe : Temp d'exécution
- Coût optimum : la meilleure solution jusqu'à présent
- Coût moyen : Le coût moyen (la somme des solutions est divisée par 10)
- PRD :l'écart en pourcentage relatif par rapport à la solution la plus connue est calculé comme suit : PRD = [(coût-moy – coût-opt)/coût-opt] * 100

Les différents paramètres de notre algorithme GASA sont comme suit :

Paramètre	**Valeur**
Taille de la population	30 - 100
Nombre d'iterations	300
Taux de croisement	0.7
Taux de mutation	0.01
Temperature initiale	100
Temperature finale	0.001
A	0.98

Table3.1 Réglages des paramètres

Les résultats numériques obtenus sont dressés dans la table 3.1

Problème	Coût-opt	QAP-GASA					
		Pop = 30			Pop =100		
		Coût-moy	PRD%	Temp-exe(S)	Coût-moy	PRD%	Temp-Exe(S)
bur26a	5426670	5438575	0.219	126,589	5426670	**0.000**	381,97
bur26b	3817852	3831812	0.365	88,723	3817852	**0.000**	689,17
bur26c	5426795	5458252	0.579	103,141	5431755	0,091	390,15
bur26d	3821225	3822474	0.032	103,048	3821249	**0.000**	393,22
bur26e	5386879	5411230	0.452	108,013	5387728	0,016	390,21
bur26f	3782044	3796331	0.377	108,175	3796053	0,370	384,11
bur26g	10117172	10169719	0.519	143,248	10118970	0,018	381,67
bur26h	7098658	7111422	0.179	167,843	7098686	**0.000**	383,93
chr12a	9552	13154	37.709	40,12	9552	**0.000**	110,94
chr12b	9742	10198	4.680	35,877	10102	3,695	126,29
chr18b	1534	1534	**0.000**	62,382	1534	**0.000**	236,33
esc16a	68	68	**0.000**	65,38	68	**0,000**	236,33
esc16b	292	292	**0.000**	49,731	292	**0,000**	200,18
esc16c	160	160	**0.000**	54,569	160	**0,000**	194,23
esc16d	16	16	**0.000**	57,65	16	**0,000**	186,31
had12	1652	1652	**0.000**	35,459	1652	**0,000**	119,15
had14	2724	2726	0.073	36,183	2744	0,734	119,34
had16	3720	3726	0.161	68,884	3720	**0.000**	133,69
had18	5358	5400	0.783	82,543	5360	0,037	164,96
had20	6922	6978	0.809	67,837	6922	**0.000**	194,35
kra30a	88900	99640	12.081	151,521	91120	2,497	406,98
kra30b	91420	100110	9.505	15,417	92820	1,531	389,16
kra32	88700	94820	6.899	202,511	90860	2,435	432,11
nug12	578	586	1.384	30,445	578	**0.000**	98,59
nug14	1014	1028	1.380	43,695	1026	1,183	118,76
nug15	1150	1180	2.608	46,839	1152	0,174	116,62
nug16a	1610	1700	5.590	59651	1622	0,745	143,57
nug16b	1240	1240	**0.000**	55,337	1240	**0.000**	134,33
nug17	1732	1794	3.579	45,777	1748	0,924	141,03
nug18	1930	1948	0.932	46.588	1946	0,829	165,18
nug20	2570	2580	0.389	57,742	2624	2,101	217,92
nug21	2438	2562	5.086	86,008	2458	0,820	243,84
nug22	3596	3664	1.890	88,098	3596	**0.000**	278,98
nug24	3488	3686	5.676	75,065	3510	0,631	333,84
nug25	3744	3964	5.876	74,309	3746	0,053	357,33
nug27	5234	5468	4.470	132,061	5234	**0.000**	453,02
nug28	5166	5386	4.258	133,165	5228	1,200	440,31
nug30	6124	6418	4.800	164.368	6150	0,425	406,71
rou12	235528	243916	3.561	29,834	240038	1,915	100,27
rou15	354210	364746	2.974	33,112	354210	**0,000**	142,81
rou20	725522	753268	3.824	61,463	737396	1,637	229,77
scr12	31410	33632	7.074	26,22	31410	**0,000**	113,04
scr20	110030	116746	6.103	72,745	110868	0,762	243,40
sko42	15812	17218	8.891	248,807	16094	1,783	889,75
tai12a	224416	244286	8.854	25,189	224416	**0,000**	109,22
tai15a	388214	402066	3.568	33,476	388988	0,199	148,64
tai17a	491812	505298	2.742	54,025	498192	1,297	181,79
tai20a	703482	728530	3.560	67,566	717896	2,049	240,33
tai30a	1818146	1934506	6.399	129,604	1852588	1,894	14,9
wil50	48816	51204	4.891	419,929	49216	0,819	243,84

Table 3.2 Résultats numériques de l'algorithme QAP-GASA

D'après la table 3.2 : on peut remarquer que les meilleurs résultats sont obtenus avec : Pop = 100 et nombre itérations = 300. Pour évaluer la performance de l'algorithme QAP-GASA, une étude comparative est réaliser avec l'algorithme génétique classique (GA). Les résultats de cette comparaison sont résumée dans la table 3.3.

Problème	Coût-opt	GA			QAP-GASA		
		Pop = 100			Pop = 100		
		Coût-moy	PRD%	Temp-exe(S)	Coût-moy	PRD%	Temp-Exe(S)
bur26a	5426670	5450413	0.437	18	5426670	**0.000**	381,97
bur26b	3817852	3851425	0.879	18	3851425	**0.000**	689,17
bur26c	5426795	5463410	0.674	20	5431755	0,091	390,15
bur26d	3821225	3849772	0.747	17	3821249	**0.000**	393,22
bur26e	5386879	5407552	0.383	18	5387728	0,016	390,21
bur26f	3782044	3815650	0.888	18	3796053	0,370	384,11
bur26g	10117172	10188063	0.700	18	10118970	0,018	381,67
bur26h	7098658	7176171	1.091	18	7098686	**0.000**	383,93
chr12a	9552	9552	**0.000**	14	9552	**0.000**	110,94
chr12b	9742	10102	3.695	12	10102	3,695	126,29
chr18b	1534	1792	16.818	16	1534	**0.000**	236,33
esc16a	68	74	8.823	11	68	**0,000**	236,33
esc16b	292	292	**0.000**	12	292	**0,000**	200,18
esc16c	160	164	2.5	16	160	**0,000**	194,23
esc16d	16	16	0.000	13	16	**0,000**	186,31
had12	1652	1660	0.484	13	1652	**0,000**	119,15
had14	2724	2756	1.174	13	2744	0,734	119,34
had16	3720	3770	1.344	11	3720	**0.000**	133,69
had18	5358	5426	1.269	11	5360	0,037	164,96
had20	6922	7056	1.935	12	6922	**0.000**	194,35
kra30a	88900	105020	18.132	16	91120	2,497	406,98
kra30b	91420	104780	14.613	16	92820	1,531	389,16
kra32	88700	102760	15.851	20	90860	2,435	432,11
nug12	578	606	4.844	11	578	**0.000**	98,59
nug14	1014	1072	5.719	11	1026	1,183	118,76
nug15	1150	1188	3.304	12	1152	0,174	116,62
nug16a	1610	1692	5.093	11	1622	0,745	143,57
nug16b	1240	1346	8.548	11	1240	**0.000**	134,33
nug17	1732	1836	6.004	11	1748	0,924	141,03
nug18	1930	2058	6.632	11	1946	0,829	165,18
nug20	2570	2716	5.680	12	2624	2,101	217,92
nug21	2438	2664	9.269	12	2458	0,820	243,84
nug22	3596	3902	8.509	13	3596	**0.000**	278,98
nug24	3488	3824	9.633	17	3510	0,631	333,84
nug25	3744	4078	8.920	19	3746	0,053	357,33
nug27	5234	5686	8.635	19	5234	**0.000**	453,02
nug28	5166	5702	10.375	19	5228	1,200	440,31
nug30	6124	6866	12.116	19	6150	0,425	406,71
rou12	235528	240122	1.950	9	240038	1,915	100,27
rou15	354210	380722	7.484	12	354210	**0,000**	142,81
rou20	725522	772350	6.454	14	737396	1,637	229,77
scr12	31410	31410	0.000	12	31410	**0,000**	113,04
scr20	110030	134970	22.666	16	110868	0,762	243,40
sko42	15812	17726	12.104	28	16094	1,783	889,75
tai12a	224416	243762	8.6206	12	224416	**0,000**	109,22
tai15a	388214	403016	3.812	13	388988	0,199	148,64
tai17a	491812	521158	5.966	13	498192	1,297	181,79
tai20a	703482	753460	7.104	15	717896	2,049	240,33
tai30a	1818146	1968222	8.254	20	1852588	1,894	14,9

Table 3.3 Comparaison entre QAP-GASA avec l'algorithme GA classique.

Pour évaluer la performance de notre méthode proposée, une deuxième étude comparative a été réalisé sur les mêmes données avec les algorithmes QAP-GASA, GA, et BBO [15] . Les résultats sont dressés dans table 3.4

Problème	PRD%		
	GASA	BBO	GA
esc16a	**0,000%**	8.235%	11,764
esc16b	**0,000%**	0.000	6,25
esc16c	**0,000%**	5.875	6,25
esc16d	**0,000%**	22.500	12,5
had12	**0,000%**	1.877	3,026
had16	**0,000%**	3.468	3,387
Had18	**0,000%**	3.763	2,948
had20	**0,000%**	4.143	3,640
nug16b	**0,000%**	12.694	10,322
rou15	**0,000%**	12.105	8,695

Table 3.4 Résultats de comparaison.

La figure 3.8 compare la déviation en pourcentage relative de quelques cas de QALIB pour notre algorithme proposé QAP-GASA, GA et BBO. Le résultat montre que GASA a plus de qualité que les autres algorithmes pour résoudre le QAP.

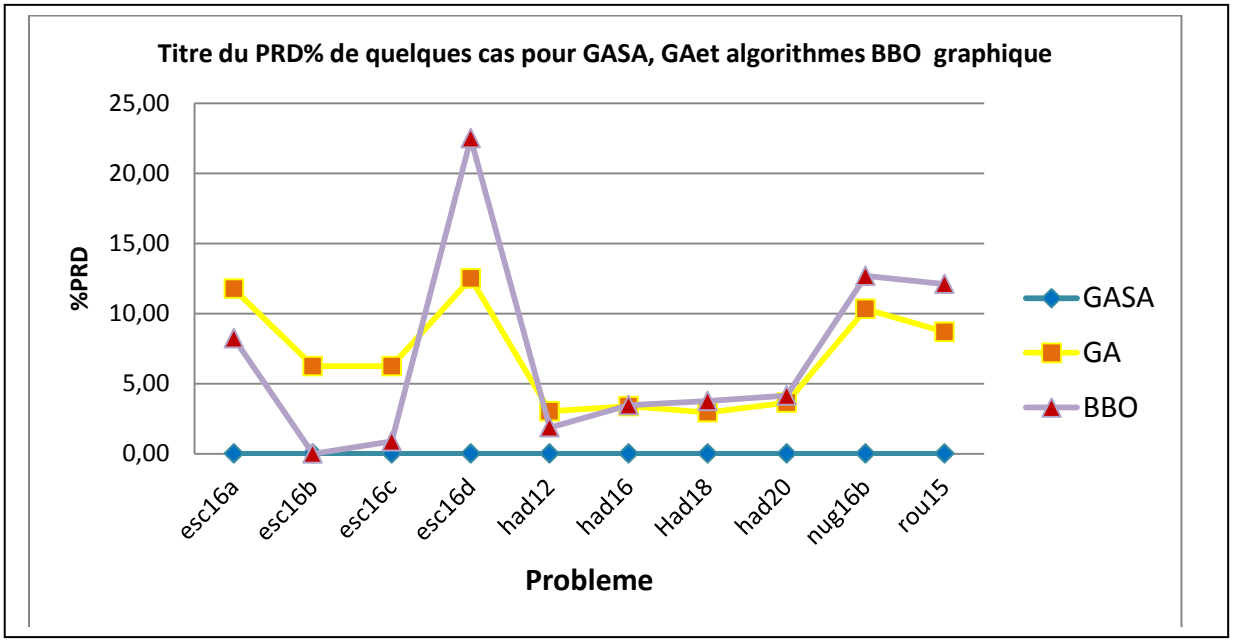


Figure 3.9 %PRD de quelques cas pour QAP-GASA, GA et algorithmes BBO.

Une dernière comparaison a été faite avec d'autres méthodes hybrides de la littérature, en particulier BBOTS [15], Les différents résultats sont présentés dans la table 3.5.

Problème	Coût-opt	GASA		BBOTS	
		PRD%	Temp-exe(S)	PRD%	Temp-Exe(m)
bur26a	5426670	**0.000**	381,97	0.028	5
bur26b	5426670	**0.000**	689,17	0.000	10
bur26c	5426795	0,091	390,15	0.000	10
bur26d	3821225	**0.000**	393,22	0.000	10
bur26e	5386879	0,016	390,21	0.000	10
bur26f	3782044	0,370	384,11	0.000	10
bur26g	10117172	0,018	381,67	0.000	10
bur26h	7098658	**0.000**	383,93	0.000	10
chr12a	9552	**0.000**	110,94	0.000	10
chr18b	1534	**0.000**	236,33	0.000	10
esc16a	68	**0,000**	284,42	0.000	10
esc16b	292	**0,000**	200,18	0.000	10
esc16c	160	**0,000**	194,23	0.000	10
esc16d	16	**0,000**	186,31	0.000	10
had12	1652	**0,000**	119,15	0.000	10
had14	2724	0,734	119,34	0.000	10
had16	3720	**0.000**	133,69	0.000	10
had18	5358	0,037	164,96	0.000	10
had20	6922	**0.000**	194,35	0.000	10
kra30a	88900	2,497	406,98	0.090	9
kra30b	91420	1,531	389,16	0.060	6
kra32	88700	2,435	432,11	0.311	7
nug12	578	**0.000**	98,59	0.000	10
nug14	1014	1,183	118,76	0.000	10
nug15	1150	0,174	116,62	0.000	10
nug16a	1610	0,745	143,57	0.000	10
nug16b	1240	**0.000**	134,33	0.000	10
nug17	1732	0,924	141,03	0.012	9
nug18	1930	0,829	165,18	0.000	10
nug20	2570	2,101	217,92	0.000	10
nug21	2438	0,820	243,84	0.000	10
nug22	3596	**0.000**	278,98	0.065	10
nug24	3488	0,631	333,84	0.000	10
nug25	3744	0,053	357,33	0.000	10
nug27	5234	**0.000**	453,02	0.000	10
nug28	5166	1,200	440,31	0.209	4
nug30	6124	0,425	406,71	0.000	2
rou12	235528	1,915	100,27	0.000	10
rou15	354210	**0,000**	142,81	0.000	10
rou20	725522	1,637	229,77	0.062	4
scr12	31410	**0,000**	113,04	0.000	10
scr20	110030	0,762	243,40	0.000	10
sko42	15812	1,783	889,75	0.028	9
tai12a	224416	**0,000**	109,22	0.000	10
tai15a	388214	0,199	148,64	0.000	10
tai17a	491812	1,297	181,79	0.093	8
tai20a	703482	2,049	240,33	0.677	0
tai30a	1818146	1,894	14,9	1.795	0

Table 3.5 Comparaison entre QAP-GASA avec l'algorithme BBOTS

10. Conclusion

Dans ce chapitre nous présenté le problème d'affectation quadratique ainsi les résultats expérimentaux obtenus par notre méthode hybride proposée (QAP-GASA) pour la trouver une solution approchée au problème d'affectation quadratique QAP.

L'étude comparative a montré l'efficacité de notre approche par rapport à l'algorithme génétique classique et quelques méthodes de littérature.

Conclusion générale

Nous avons tout au long de ce mémoire mis l'accent sur l'utilité et l'efficacité de l'hybridation des méthodes à solution unique en particulier l'algorithme de recuit simulé (SA) et une méthode de population tel que les algorithmes génétiques pour la résolution du problème QAP. Les résultats obtenus soulignent la capacité des métaheuristiques pour trouvé une solution approchée à plusieurs problèmes d'optimisation NP-difficiles, en particulier le problème QAP. Les expérimentations dressées dans notre travail et l'étude comparative montrent clairement les avantages des méthodes hybrides pour résoudre le problème QAP.

Il faut noter que l'efficacité des méthodes bio-inspirées n'a pas résolu la complexité de tous les problèmes. Elles restent toujours souffrent du problème de la convergence prématurée des solutions, les bruits lors de l'évaluation des solutions, et l'indistinction de la réalité entre une solution optimale et une autre globale.

Comme perspective à ce travail, d'autres scenarios d'hybridation sont possibles, d'autres métaheuristiques peuvent être utilisées pour améliorer la qualité de la solution et de réduire le temps d'exécution.

Bibliographie

[1] **A.**BECHIR, Résolution des problèmes d'optimisation par les systèmes multi-agents et les approches évolutionnaires .Thèse de doctorat. Université Mohamed Khider-Biskra. 2016.

[2] A.Gherboudj, Méthodes de résolution de problèmes difficiles académiques ,Université de Constantine2,Thèse 2013.

[3] A.Kamil, Application d'un algorithme hybride à colonies de fourmis au problème d'affectation quadratique, Université du Quebec en Abitibi-Témiscaminque, thèse février 2008.

[4] B.Ilhem, Perfectionnement de métaheuristiques pour l'optimisation continue université houari Boumediene , thèse juin 2013.

[5] C. Mancel, Modélisation et résolution de problèmes d'optimisation combinatoire issus d'applications spatiales. Automatique / Robotique. INSA de Toulouse, 2004.

[6] F. Troudi, Résolution du problème de l'emploi du temps : Proposition d'un algorithme révolutionnaire multi objectif ,Université Mentouri Constantine, mémoire de Magister 2005.

[7] H. Osman, G. Laporte. Metaheuristics: A bibliography. Ann. Oper. Res. Vol. 63, N° 5, pp. 513-623, 1996.

[8] J. Feldman , A. Feigenbaum Computers and thought. McGraw-Hill Inc. pp.192. New York, 1963.

[9] L. Said, Méthodes bio-inspirées hybrides pour la résolution de problèmes complexes, Université Constantine 2, thèse avril 2013

[10] M. Akli, Problème de tournées de véhicules avec contraintes et fenêtre de temps, Université de Mouloud Mammeri ,Tizi Ouzou thèse mars 2013.

[11] Newella, The heuristic of George Polya and its relation to artificial intelligence. A paper given at The International Symposium on the Methods of Heuristic. University of Bern, Switzerland, Sept. 15-18, pp.16 pp. 1980.

[12] T.C. Koopmans, M.J. Beckmann - Assignment problems and the location of economies activities. Econometrica, vol. 25, 1957, pp. 53-76.

[13] V.Bachelet, Métaheuristique parallèles hybrides :Application au problème d'affectation quadratique, Université des science et technologie de Lille, thèse 1999.

[14] ZAGHDOUD.R, Hybridation d'algorithme génétique pour les problèmes des véhicules intelligents autonomes : applications aux infrastructures portuaires de moyenne taille l'école centrale de Lille et l'institut supérieur de gestion (Tunis) , 23 Mai 2016 .

[15] W.loon lim,A. Wibowo,M.Ishak desa,H.Haron, abiogeography based optimization algorithm hybrid with tabu search for the quadratic assignment problem, (IJACSA) international . …….,vol5 , N1 , 2014.

ملخــص

مشكلة التعيين ألتربيعي (QAP) هي مشكلة تحسين اندماجي، و تنتسب إلى صنف المشاكل الصعبة و التي تم تطبيقها في مجالات مختلفة . الهدف الرئيسي من هذا العمل هو تطوير أساليب القرار التي من شأنها أن تكون فعالة بحيث تولد حلول قريبة من الأمثل ويجب استغلال مزايا التهجين من خلال الجمع بين الخوارزميات باستخدام النهج ألتآزري .

هذه الإستراتيجية المقترحة توجه محاكاة البحث الصلب للخروج من الأوبتيما المحلية و الاستكشاف بطريقة فعالة و استغلال مساحة البحث و إيجاد حل تقريبي لمشكلة QAP .

في هذه المذكرة نقترح تطبيق الخوارزمية الجينية (GA) مختلطة مع محاكاة الصلب (SA) لحل مشكلة التعيين الربيعية (QAP) ، لتقييم أداء هذه الطريقة ، أجريت عمليات محاكاة عديدة على 60 حالة من حالات QAP لمقارنة QAP-GASA بالخوارزميات الموجودة في مكتبة QAP ، وتبين النتائج العددية التي تم الحصول عليها أن هذه الخوارزمية تنتج حلولاً مثلى في وقت معقول ، وهذا العمل يبين أن تكييفنا المقترح فعال في حل مشكلة التخصيص الربيعية QAP .

كلمات مفتاحيه : مشكلة التخصيص الربيعية ، صنف المشاكل الصعبة ، التهجين ، الخوارزمية الجينية ، محاكاة الصلب .

Résumé

Le problème d'affectation quadratique (QAP) est un problème d'amélioration de l'intégration et fait partie de la catégorie des problèmes NP-difficiles qui a été appliqué dans différents domaines. L'objectif principal de ce travail est de concevoir une méthode hybride basée sur les métaheuristiques pour trouver une solution approchée au problème QAP. L'approche développée combine entre les avantages des algorithmes génétiques (GA) notamment le principe de diversification et les avantages de la recherche locale apportées par le recuit simulé (SA). Pour évaluer la performance de la méthode QAP-GASA, une série d'expérimentations via une étude comparative a été présentée.

Mots Clés : problème d'affectation quadratique (QAP), problèmes NP-difficiles, méthode hybride, algorithme génétique, recuit simulé.

Abstract

The Quadratic assignment problem (QAP), is a problem of improvement of integration and part of the category of problems NP-hard problem which was applied in different domains. The main objective of this job is to conceive a hybrid method based on métaheuristiques to find a solution approached in problem QAP. Developed approach combines between the advantages of genetic algorithms (GA) notably the principle of diversification and the advantages of local research brought by the simulated annealing (SA). To assess the performance of method QAP-GASA, a series of experimentation via a comparative study was introduced.

Keywords : Quadratic assignment problem (QAP), NP-hard problem, hybrid method, genetic algorithm, simulated annealing.