République Algérienne démocratique et populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Elbachir Al Ibrahimi de Bordj Bou Arreridj
Faculté des Mathématiques et d Informatique
Département des Mathématique

Mémoire

en vue de l'obtention du diplôme

Master en mathématique

spécialité de la recherche opérationnelle

Thème

Étude comparative des algorithmes de colonies de fourmis pour Optimisation multi-objectifs

présenté par :

BOUGUERRA YASMINA

TRIRAT NOUR EL HOUDA

Devant le jury composé de :

Président Dr :Fares Nour El Houda

Examinateur Dr :Benabid Sonia

Encadreur Dr :***Saha Adel*** MCB,Université de BBA

promotion :2021/2022

résumé

résumé :

Dans ce travail nous avons traité un sujet brûlant de l'étude comparative des algorithmes de colonies de fourmis pour l'optimisation multi-objectifs où nous avons abordé le processus de base des algorithmes de colonies de fourmis pour résoudre les MOP afin d'illustrer les différences entre chaque étape puis avons présenté une classification relative complète des les algorithmes de différents côtés Après cela, compte tenu du résultat de la classification, nous avons effectué Nous avons comparé certains algorithmes typiques provenant de différentes classes avec différentes tailles d'exemples de TSP (problème du voyageur de commerce) et nous avons analysé les résultats du point de vue de la qualité de la solution et du taux de convergence .

Mots clés : problème d'optimisation multi-objectifs ; algorithme d'optimisation multi-objectifs ; algorithme méta-heuristique ; optimisation multi-objectifs des colonies de fourmis

Abstract :

In this work we dealt with a hot topic of Comparative Study of Ant Colony Algorithms for Multi-Objective Optimization where we touched on the basic process of ant colony algorithms to solve MOPs to illustrate the differences between each step and then presented a complete relative classification of the algorithms from different sides After that, given the classification result, we performed We compared some typical algorithms that come from different classes with different sizes of TSP (traveling salesman problem) examples and we analyzed the results from the perspective of solution quality and convergence rate.

Keywords : multi-objective optimization problem ; multi-objective optimization algorithm ; meta-heuristic algorithm ; multi-objective ant colony optimization

ملخص:

في هذا العمل تناولنا موضوعًا ساخنًا هو دراسة مقارنة لخوارزميات مستعمرة النمل لتحسين الأهداف المتعددة حيث تطرقنا إلى العملية الأساسية لخوارزميات مستعمرات النمل لحل MOPs لتوضيح الاختلافات بين كل خطوة ثم قدمنا تصنيفًا نسبيًا كاملًا للخوارزميات من جوانب مختلفة بعد ذلك بالنظر لنتائج التصنيف قمنا بمقارنة بعض الخوارزميات النموذجية التي تأتي من فئات مختلفة بأحجام مختلفة من أمثلة TSP (مشكلة بائع متجول) وقمنا بتحليل النتائج من منظور جودة الحل ومعدل التقارب

الكلمات المفتاحية : مشكلة التحسين متعددة الأهداف ؛ خوارزمية التحسين متعددة الأهداف ؛ خوارزمية ما وراء الكشف عن مجريات الأمور ؛ متعددة الأهداف التحسين لمستعمرات النمل

dédicace

$\mathcal{A}$ ma très chère mère "Chouder Farida" , qui me donne toujours l'espoir de vivre et qui n'a jamais cessé de prier pour moi.

$\mathcal{A}$ mon très cher père "Nacerddine" , pour ses encouragements, son soutien, et son sacrifice afin que rien n'entrave le déroulement de mes études.

Quoi que de plus que de pouvoir partager les meilleurs moments de sa vie avec les êtres qu'on aime.

Arrivé au terme de mes études, j'ai le grand plaisir de dédier aussi ce modeste travail :

$\mathcal{A}$ mes frères (salah eddine , Zakaria , Mouhamed , Abdeslam , Amar) et sœurs(Takoua , Asma , Hanane , Meriam , Rahma)

$\mathcal{A}$ Mes amis " sara " et" soumia"

$\mathcal{A}$ mes cousins et cousines, toute ma grande famille , tous ceux que j'aime et respecte

Bouguerra Yasmina

Je dédie cette mémoire

mon père , que dieu lui fasse miséricorde,

pour son soutien et m'aide dans mon par cour académique,

et ma mère ,la fleur de ma vie,

Mes frère(Amar , Youcef ,Ayoub)

Mes sœurs(Zahia, warda, Nasima, Sara)

la famille Trirat

Mes amis : Yasmina,El chaima ,Manal, Bothaina

NOUR EL HOUDA

Remerciements

Tout d'abord,nous remercions le dieu , notre créateur de nos avoir donné les forces pour accomplir ce travail.

Nous adressons le grand remerciement a notre encadreur ***Mr.SAHA*** *qui a propose le thème de ce mémoire , pour ses conseil et ses dirigés.*

Nous tenons aussi à exprimer nos gratitudes et nos sincères remerciements à tous les membres du jury pour avoir accepté de juger et de mettre en valeur ce mémoire.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

La recherche opérationnelle est en fait née lors de la Deuxième Guerre mondiale des efforts conjugués d'un groupe d'éminents mathématiciens (dont von Neumann , Metropolis , Wald , Wiener , Dantzig et Bellman) de contribuer à leur manière à la victoire alliée .c'est ainsi qu'il a été crée une nouvelle méthodologie quantitative d'aide à la décision pouvant être caractérisée par les mots-clés modélisation , simulation et optimisation .

Rapidement , la recherche opérationnelle repousse les frontières de son domaine initial , pour s'étendre à la modélisation et à l'optimisation de système dans des domaines les plus divers , allant de la conception et la configuration à l'exploitation de systèmes techniques complexes(réseaux de communication , système informatiques), de la gestion stratégique d'investissements à celle de la chaine logistique (transports , production , stocks).

La recherche opérationnelle fait également son apparition dans des domaines tels que la santé et l'instruction publiques , la voirie , le ramassage et la distribution de courrier , la production et le transport d'énergie , les télécommunications , mais aussi dans les banques et les assurances .

De pair avec cette éclosion d'applications , les méthodes et fondements théoriques de la recherche opérationnelle continuent à se développer dans une symbiose féconde avec d'autre domaines des mathématique appliquées.

La recherche opérationnelle apporte ainsi une méthodologie de modélisation et une collection d'outils mathématique devenus indispensables pour tout ingénieur , manager ou économiste .En effet , le pouvoir d'expression de ses modèles , c'est-à-dire leur capacité à représenter en termes mathématiques les systèmes les plus complexes , et l'efficacité de ses algorithmes sont la clé de l'utilisation des" décision.

Dans la plupart des problèmes du monde réel, il ne s'agit pas d'optimiser seulement un seul critère mais plutôt d'optimiser simultanément plusieurs critères et qui sont généralement conflictuels. Dans les problèmes de conception, par exemple, il faut le plus souvent trouver un compromis entre des besoins technologiques et des objectifs de coût.L'optimisation multi-objectif consiste donc à optimiser simultanément plusieurs fonctions. La notion de solution optimale unique dans l'optimisation uni-objectif disparaît pour les problèmes d'optimisation multi-objectif au profit de la notion d'ensemble de solutions Pareto optimales.

Ces dernières années, la plupart des métaheuristiques présentées dans la littérature pour résoudre des problèmes combinatoires sont d'inspiration biologique. Parmi ces métaheuristiques, nous pouvons citer les algorithmes génétiques, le recuit simulé, les réseaux de neurones, les algorithmes à estimation de distribution, l'optimisation par essaim de particules et l'optimisation par colonie de fourmis.

Ant Colony Optimization (ACO) est une métaheuristique inspirée de l'observation de véritables colonies de fourmis et basée sur leur comportement de recherche de nourriture collectif. Les fourmis sont des insectes sociaux et vivent en colonies. Leur comportement est régi par l'objectif de survie de la colonie. Lorsqu'elles recherchent de la nourriture, les fourmis se déplacent fréquemment entre leur nid et les sources de nourriture. Au début, les fourmis explorent les environs de leur nid de manière aléatoire. En se déplaçant, les fourmis déposent sur leur chemin des substances spéciales appelées phéromones. Les fourmis peuvent sentir les phéromones. Au moment de choisir leur chemin, ils ont tendance à choisir, en probabilité, des chemins marqués par de fortes concentrations de phéromones. Dès qu'une fourmi trouve une source de nourriture, elle évalue la quantité et la qualité de la nourriture et en ramène une partie au nid. Lors du voyage de retour, la quantité de phéromones qu'une fourmi laisse au sol peut dépendre de la quantité et de la qualité de la nourriture. Les sentiers de phéromones guideront les autres fourmis vers la source de nourriture. La communication indirecte entre les fourmis via des pistes de phéromones leur permet de trouver les chemins les plus courts entre leur nid et les sources de nourriture.

L'optimisation des colonies de fourmis a été appliquée avec succès pour résoudre de nombreux problèmes d'optimisation tels que le déplacement le problème du vendeur, le problème d'ordonnancement du magasin de flux et le problème d'affectation quadratique. Outre son domaine d'origine de l'optimisation combinatoire, ACO est également désormais utilisé pour résoudre des problèmes d'optimisation continue. Certaines extensions des algorithmes ACO ont été proposées dans la littérature telles que ACS , Ant System (AS) et MMAS . Récemment, il y a également eu un certain nombre d'études étendant l'ACO au domaine de l'optimisation multi-objectifs.

Chapitre 1

Problème d'optimisation combinatoire

1.1 Introduction

Un problème d'optimisation combinatoire consiste à trouver la meilleure solution dans un ensemble discret de solutions appelé ensemble des solutions réalisables. En général, cet ensemble est fini mais de cardinalité très grande.

Dans ce chapitre, nous introduisons les bases et quelques concepts sur l'optimisation, et les méthodes de les résoudre et quelques problème d'optimisation.

1.2 Optimisation

L'optimisation est une branche de la science qui étudie et met en œuvre la recherche d'une solution ou de plusieurs solutions du problème qu'on veut déterminer le meilleur élément (optimal) d'un ensemble d'éléments ; tout en respectant un critère donné(maximiser le profit ,minimiser le cout, minimiser le temps,chemin ...).

Les solutions du problème ne sont pas toutes de même valeur d'importance, celle qui soit réalisable, ou non réalisable, soit optimale ou non optimale ; pour cela il faut chercher la meilleure solution parmi les solutions trouvées [1].

1.2.1 Problème d'optimisation

On peut trouver de nombreuses définition de problème d'optimisation, nous avons retenu de celle de **collette** et **Siarry** qui propose la définition suivant :

Définition 1 : *problème d'optimisation*

Un problème d'optimisation se définit comme la recherche du minimum ou de maximum d'une fonction donnée, mathématiquement, dans le cas d'une minimisation, un problème d'optimisation se présentera sous la forme suivante [2] :

$$(po) = \left\{ \begin{array}{lll} & & minimiser f(x) \\ g_i(x) \leq 0 & i = 1 \cdots n & \text{contraintes d'inégalités} \\ h_i(x) = 0 & i = 1 \cdots n & \text{contraintes d'égalités} \end{array} \right\} \quad \cdots (1.1)$$

En d'autre termes, résoudre un problème d'optimisation (PO) revient à déterminer une solution $s^* \epsilon S$ minimisant ou maximisant la fonction f avec S l'ensemble des solutions ou l'espace de recherche et $f : S \longrightarrow Y$ une application ou une fonction d'évaluation qui à chaque configuration s'associer une valeur $f(S) \epsilon Y$. Il est possible de passer d'un problème de maximisation à un problème de minimisation grâce à la propriété suivante [2] :

$$\max_{s \in S} f(s) \simeq \min_{s \in S}(-f(s)) \qquad \ldots(1.2)$$

Généralement, une solution $s \epsilon S$ est un vecteur d'un espace à N dimensions.

Définition 2 : *problème d'optimisation continue*
Dans le cas de variables réelles, on a [2] :

$$\mathbf{S} \subseteq \mathbb{R}^{\mathbf{N}} \qquad \ldots(1.3)$$

On parle alors de problème d'optimisation en variables continues.
Un problème d'optimisation continue (PO) peut être formulé de la façon suivante [2] :

$$(PO) \quad \min_{s \in \mathbb{R}^N} f(s) \qquad \ldots(1.4)$$

Définition 3 : *problème d'optimisation combinatoire*
Un problème d'optimisation combinatoire est un problème d'optimisation dans lequel l'espace de recherche S est dénombrable. Un problème d'optimisation combinatoire (POC) peut être formulé ainsi [2] :

$$(POC) \quad \min_{s \in \mathbb{Z}^N} f(s) \qquad \ldots(1.5)$$

Où une solution S est un vecteur composé de N valeurs entières soit[2] :

$$\mathrm{S} \subseteq \mathbb{Z}^{\mathrm{N}} \ldots(1.6)$$

La principale différence entre problème d'optimisation continue et le problème d'optimisation combinatoire repose sur l'utilisation des variables discrètes, dans les deux catégories de problèmes, une solution $s \epsilon S$ est une instanciation des variables $x_i \epsilon X$,ou i est l'indice da la variable dans $[1, N]$,et X est le vecteur de démentions N correspondant à la solution, et $f(s)$ est son évaluation, résoudre ces problèmes revient à trouver une solution optimale appelées aussi optimum global. [2]

1.2.2 Optimisation combinatoire

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire. Bien

que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont 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 valable pour toutes les données [3].

L'optimisation combinatoire trouve des applications dans des domaines aussi variés que la gestion, l'ingénierie, la conception, la production, la télécommunication, le transport, l'énergie, la médecine et l'informatique...

1.3 Exemples de problèmes d'optimisation Combinatoire

1.3.1 Le problème du voyageur de commerce

Dans ce problème, il s'agit de trouver le chemin le plus court reliant n villes données, chaque ville ne devant être visitée qu'une et une seule fois, et revenir à la ville de départ. La difficulté du problème vient de l'explosion combinatoire du nombre de chemins à explorer lorsque l'on accroît le nombre de villes à visiter [4] .

Plus formellement, ce problème peut être modélisé comme suit : Étant donné un graphe non orienté complet $\mathbf{G = (S, A)}$,et une fonction de distance $\mathbf{d : A \rightarrow \mathbb{R}^+}$, on cherche à déterminer un cycle hamiltonien de distance totale minimale [4].

Pour ce problème, trouver s'il existe un chemin hamiltonien est un problème NP-complet,la recherche du plus court chemin hamiltonien est un problème NP-difficile [4] .

1.3.2 Le problème du sac à dos multidimensionnel

Figure 1.1 -exemple de problème du sac à dos.[16]

Le problème du sac à dos multidimensionnel (Multidimensional Knapsack Problem : MKP) est un problème d'optimisation combinatoire sous contraintes NP difficile. Plusieurs problèmes pratiques peuvent être formalisés comme un MKP. Nous citons l'allocation des processeurs et des bases de données dans les systèmes informatiques distribués, le chargement des cargaisons, les problèmes de découpage de stock, etc ...

Le MKP permet de modéliser une grande variété de problèmes où il s'agit de maximiser un profit tout en ne disposant que de ressources limitées. De manière formelle, il est présenté comme suit [4] :

Soit

- m le nombre de ressources.

- n le nombre d'objets.

- $p_j \quad (j\epsilon 1..n)$ le profit apporté par l'objet j.

-$r_{ij} \quad (i\epsilon 1..m, j\epsilon 1..n)$ la consommation de la ressource i par l'objet j .

-$b_i \quad (i\epsilon 1..m)$ la quantité totale disponible de la ressource i.

On définit le MKP (n, m) par [4] :

$$MKP = \left\{ \begin{array}{l} \text{maximiser} \sum_{j=1}^{n} p_j.x_j \\ \text{tq} \sum_{j=1}^{n} r_{ij}.x_j \leq b_i, i \in 1 \cdot\cdot m \\ \\ x_j \in 0,1^n, j \in 1 \cdot\cdot n \end{array} \right\} \quad \cdots(1.7)$$

$x_j \quad (j\epsilon 1..n)$ est une variable de décision qui peut prendre pour valeur 0 (si l'objet j n'est pas sélectionné) ou 1 (s'il est sélectionné).

1.3.3 Le problème du sac à dos quadratique

Le problème du sac à dos quadratique, Quadratic Knapsack Problem (QKP), consiste à sélectionner un sous-ensemble d'objets dont le poids total ne dépasse pas une capacité donnée c du sac à dos et de maximiser le profit total.

Plus formellement,le QKP est défini comme suit : Supposons $N = \{1..n\}$ un ensemble d'objets donné où chaque objet $j\epsilon N$ a un poids positif w_j De plus nous disposons d'une matrice d'ordre n d'entiers non négatifs $P = \{p_{ij}\}$, où le p_{ij} est un profit accompli en sélectionnant deux objets différents i et $j\epsilon N$.En outre, le profit p_{ii} est accompli si l'objet $i\epsilon N$ est choisi. Des variables binaires x_j seront introduites et indiqueront si l'objet $j\epsilon N$ est sélectionné. Ce problème peut être formulé comme suit [4] :

$$\begin{array}{rl} \text{Maximiser} & \sum_{i\in N} \sum_{j\in N} p_{ij} x_i x_j \\ \text{tel que} & \sum_{j\in N} \ w_j x_j \leq c \\ & x_j \in \{0,1\}, j \in N \end{array}$$

Ce problème peut être considéré comme une variante du MKP, où on a une seule dimension, et où la fonction objectif est quadratique et non plus linéaire [4].

1.4 Résolution d'un problème d'optimisation combinatoire

Résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points particuliers :

- la définition de l'ensemble des solutions réalisables.

- l'expression de l'objectif à optimiser.

- le choix de la méthode d'optimisation à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution.

Afin de définir l'ensemble des solutions réalisables, il est nécessaire d'exprimer l'ensemble des contraintes du problème. Ceci ne peut être fait qu'avec une bonne connaissance du problème sous étude et de son domaine d'application [5].

Le choix de l'objectif à optimiser requiert également une bonne connaissance du problème.

La définition de la fonction objectif mérite toute l'attention de l'analyste car rien ne sert de développer de bonnes méthodes d'optimisation si l'objectif à optimiser n'est pas bien défini.

Comme nous le verrons par la suite, il peut être très difficile de trouver un objectif unique d'optimisation. Nous pouvons donc être amené à proposer une modélisation multi-objectif.

Enfin, le choix de la méthode de résolution à mettre en œuvre dépendra souvent de la complexité du problème. En effet, suivant sa complexité, le problème pourra ou non être résolu de façon optimale. Dans le cas de problèmes classés dans la classe P, un algorithme polynomial a été mis en évidence. Il suffit donc de l'utiliser. Dans le cas de problèmes NP-difficiles, deux possibilités sont offertes. Si le problème est de petite taille, alors un algorithme exact permettant de trouver la solution optimale peut être utilisé (procédure de séparation et évaluation (Branch & Bound), programmation dynamique...).

Malheureusement, ces algorithmes par nature énumératifs, souffrent de l'explosion combinatoire et ne peuvent s'appliquer à des problèmes de grandes tailles. Dans ce cas, il est nécessaire de faire appel à des heuristiques permettant de trouver de bonnes solutions approchées.Parmi ces heuristiques, on trouve les méta heuristiques qui fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes [5].

Nous voyons donc ici que la phase de modélisation du problème est très importante puisque c'est elle qui permettra par exemple de reconnaître un problème de la classe P d'un problème NP-difficile. En particulier, la définition de l'objectif est cruciale mais peut être difficile à réaliser, surtout lors de l'étude de problèmes réels [5].

1.5 Les méthodes de résolution de problème d'optimisation

Il existe deux grandes catégories de méthodes de résolution : les méthodes exactes et les méthodes approchées.

Les méthodes exactes permettent d'obtenir une solution optimale à chaque fois, mais le temps de calcul peut être long si le problème est compliqué à résoudre.

Les méthodes approchées, encore appelées heuristiques, permettent quant à elles d'obtenir rapidement une solution approchée, mais qui n'est donc pas toujours optimale.

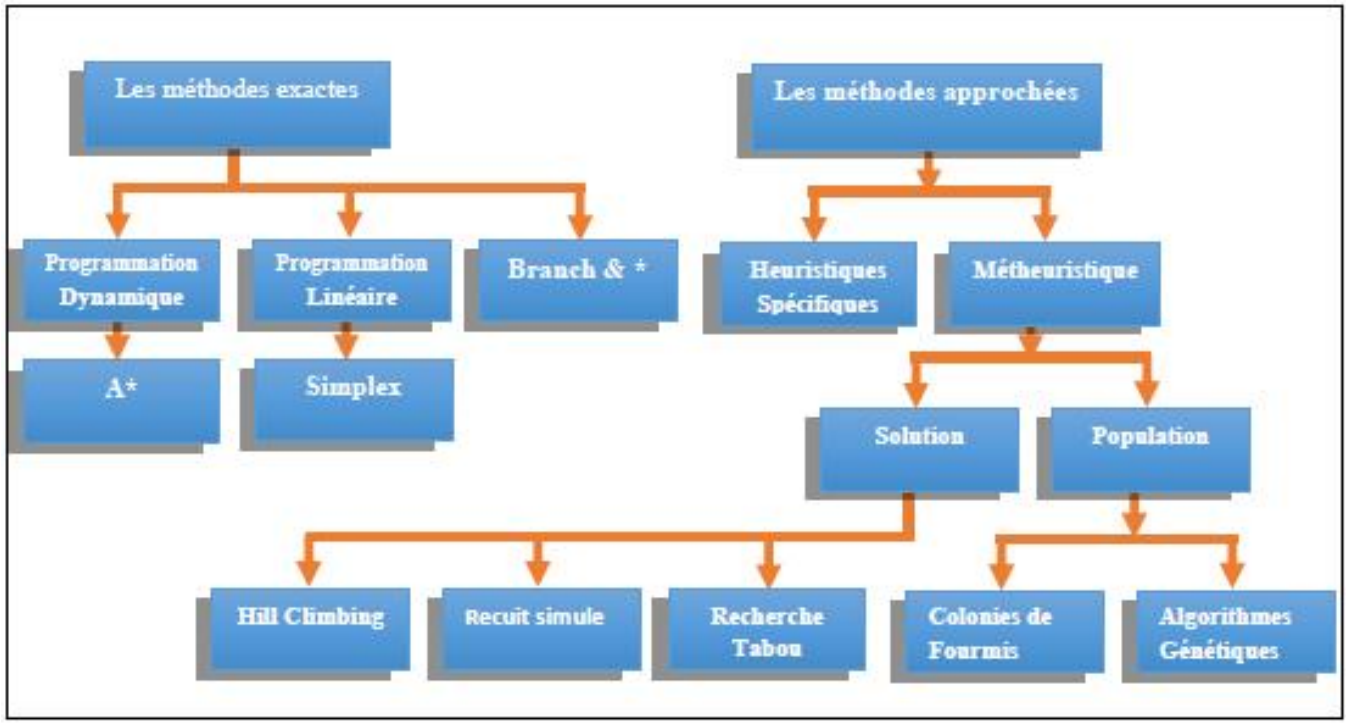

Figure 1.2- classification des méthodes de résolution des problèmes d'optimisation. [17]

1.5.1 les méthodes exactes

Les algorithmes exacts sont utilisés pour trouver au moins une solution optimale d'un problème [6].

On peut classer les méthodes exactes en quatre grandes classes [7] :

- La programmation dynamique.

- La programmation linéaire continue ou en nombres entiers.

- La programmation non linéaire avec ou sans contraintes.

- Les méthodes de recherche arborescente (Branch & Bound).

1.5.2 les méthodes approchées

Typiquement ce type de méthodes, dites heuristiques est particulièrement utile pour les problèmes nécessitant une solution en temps réel (ou très court) ou pour résoudre des problèmes difficiles sur des instances numériques de grande taille. Elles peuvent aussi être utilisées afin d'initialiser une méthode exacte (Branch & Bound par exemple) [1].

Parmi ces méthodes, il faut distinguer les heuristiques ciblées sur un problème particulier et les méta-heuristiques plus puissantes et adaptables pour résoudre un grand nombre de problèmes. Cependant une heuristique, pour être suffisamment performante sur un problème donné,nécessitera une adaptation plus ou moins fine.

Ces méthodes approchées peuvent se classer en différentes catégories [1] :

• Constructives (algorithmes glouton, méthode Pilote, GRASP)

• Recherche locale (algorithmes de descente, multi-départs, recuit simulé, algorithme à seuil, recherche Tabou, méthode de bruitage)

• Évolutionnistes (algorithmes génétiques, algorithmes d'évolution, recherche dispersée, méthode des chemins, systèmes de fourmis)

• Réseaux de neurones (Modèle de Hopfield-Tank, machine de Boltzmann, réseau auto-adaptatif, réseau élastique)

• Heuristiques Bayésiennes (optimisation globale, optimisation discrète)

• Superposition (perturbation des données, perturbation des paramètres d'une heuristique).

1.6 Conclusion

Dans ce chapitre, nous avons vu la définition de l'optimisation combinatoire et quelques exemples ,avec les méthodes de résolution en générale .

Chapitre 2

l'optimisation multi-objectif

2.1 introduction

Dans la plupart des problèmes du monde réel, il ne s'agit pas d'optimiser seulement un seul critère mais plutôt d'optimiser simultanément plusieurs critères et qui sont généralement conflictuels. Dans les problèmes de conception, par exemple, il faut le plus souvent trouver un compromis entre des besoins technologiques et des objectifs de coût. L'optimisation multi-objectif consiste donc à optimiser simultané-ment plusieurs fonctions. La notion de solution optimale unique dans l'optimisation uni-objectif disparait pour les problèmes d'optimisation multi-objectif au profit de la notion d'ensemble de solutions Pareto optimales.

2.2 Qu'est-ce qu'un problème d'optimisation multi- objectifs

On dit qu'un problème est multi-objectif s'il comporte plusieurs objectifs qui doivent être optimisés simultanément [8].

Un problème d'optimisation multiobjectifs (MOP) consiste a optimiser (minimiser/maximiser) simultanément plusieurs fonctions objectifs $f_1, f_2, ..., f_p, p \geq 2$ sur un domaine (ensemble) réalisable S appelé espace de décision définit par des inéquations $g_j(x) \leq 0, j = 1, 2, ..., m.$ Mathématiquement, ce problème peut s'écrire comme suit [9] :

$$(MOP) = \begin{cases} "min"(ou)"max" \quad (f_1(x), f_2(x), ..., f_p(x))^T, p \geq 2 \\ g_j(x) \leq 0, j = 1, 2, ..., m \end{cases} \qquad ...(2.1)$$

Tel que

- $x = (x_1, x_2, ..., x_n)^T$ est le vecteur de n variables de décision.

On pose

-$F(x) = (f_1(x), f_2(x), ..., f_p(x))^T$ est le vecteur de $p \geq 2$ fonctions objectifs a optimiser.

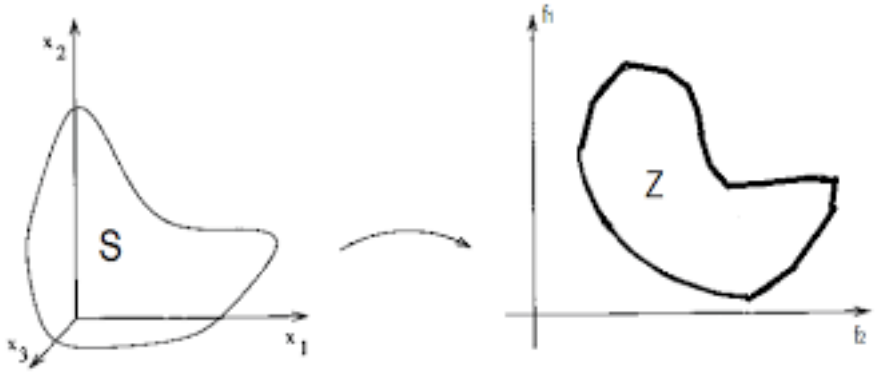

Figure 2.1-Espace de décision et l'espace des objectifs [9]

2.2.1 Fonction objective

Une ou plusieurs fonction(s) objectif(s) à optimiser (maximiser ou minimiser). La fonction objective représenté en décide le but à atteindre.

2.2.2 Espace de travail et variable de conception

L'espace de travail c'est l'espace de notre recherche et les variables de conception se sont les variables qui seront optimisées par la suite.

2.2.3 Contraintes.

Sont des contraintes d'égalités ou d'inégalités et permettent en général de limiter l'espace de travail (la solution réalisable).

2.3 Solution efficace

Solution efficace (Pareto optimale)[10] : Soit $x^* \epsilon X$

– x^* est dite efficace s'il n'existe aucune autre solution admissible $x \in X$ telle que $z(x) \leq z(x^*)$. On dit alors que $y^* = z(x^*)$ est un point non dominé.

– x^* est dite faiblement efficace s'il n'existe aucune autre solution admissible $x \in X$ telle que $z(x) < z(x^*)$. On dit alors que $y^* = z(x^*)$ est un point faiblement non dominé

2.4 Relation de dominance

Soient deux points $y^1, y^2 \in R^p$. On dit que [10] :

– y^1 domine faiblement y^2 si $y_i^1 \leq y_i^2; \forall i = 1,...,p$. On écrit alors $y^1 \leqq y^2$.

– y^1 domine strictement y^2 si $y_i^1 < y_i^2; \forall i = 1,...,p$. On écrit alors $y^1 < y^2$.

– y^1 domine y^2 si $y^1 \leqq y^2$ et $y^1 \neq y^2$. On écrit alors $y^1 \leq y^2$.

Les solution qui dominent les autres mais ne se dominent pas entre elles sont appelées solutions non dominées ou solutions optimales au sens de Pareto.

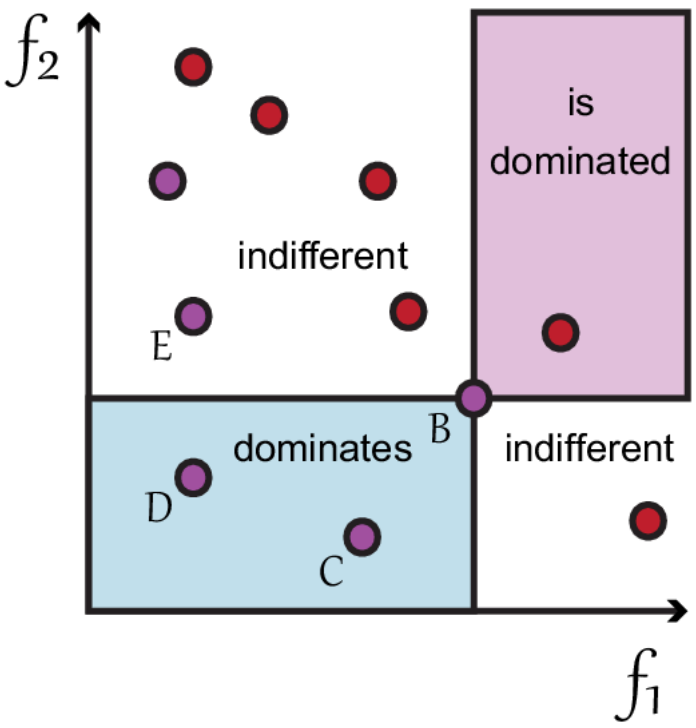

Figure 2.2-exemple de dominance [18]

2.5 Ensemble des solutions non dominées

Ensemble des solutions non dominées : Soit F l'image dans l'espace des objectifs de l'ensemble réalisable X. L'ensemble des solutions non dominées de X, est défini par l'ensemble ND(X)[11] :

$$ND(X) = \{x \in X \mid \textbf{x est non dominé par apport à } X\} \qquad (2.2)$$

Nous pouvons définir le front Pareto de F de manière analogue :

2.6 Front Pareto

Front Pareto : Soit F l'image dans l'espace des objectifs de l'ensemble réalisable X. Le front Pareto ND(F) de F est défini comme suit [11] :

$$ND(F) = \{y \in F \quad | \quad \nexists \quad z \in F, \quad z < y\} \qquad (2.3)$$

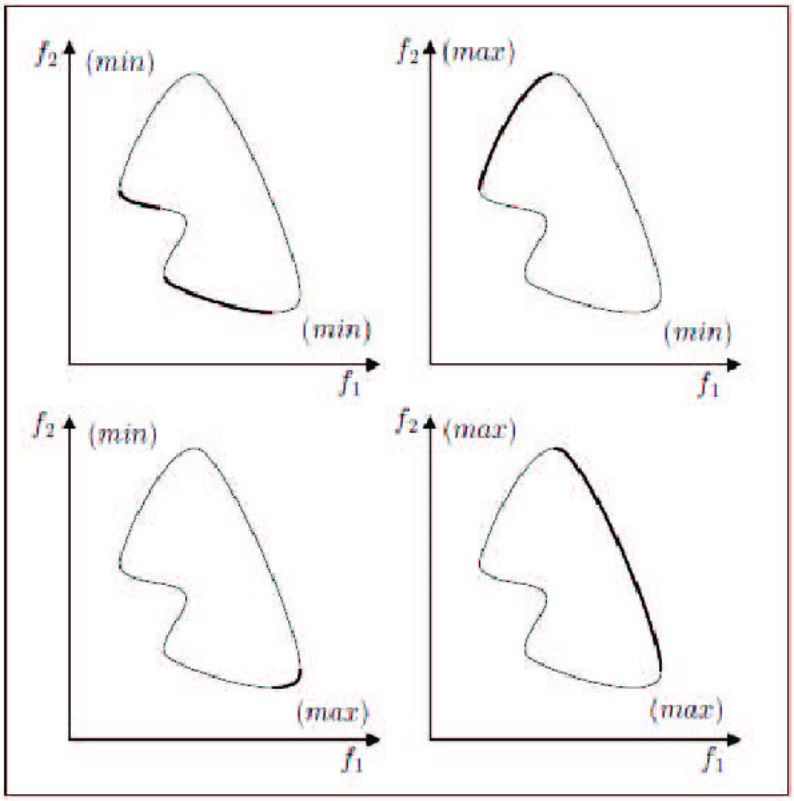

Figure 2.3 - Le front de Pareto [9]

Le front Pareto est aussi appelé l'ensemble des solutions efficaces ou la surface de compromis.

2.7 les points particuliers

Deux points particuliers apparaissent clairement : le point idéal et le point Nadir. Ces deux points sont calculés à partir du front Pareto. Le point idéal (***resp. le point Nadir***) domine(***resp. est dominé par***) tous les autres points de la surface de compromis.Bien que ces points ne soient pas forcément compris dans la zone réalisable, ils servent souvent de pole d'attraction (***resp. de répulsion***) lors de la résolution du problème. Les coordonnées de ces deux points sont maintenant définies formellement.

2.7.1 Point Idéal

Les coordonnées du point idéal (pi) correspondent aux meilleurs valeurs de chaque objectif des points du front Pareto. Les coordonnées de ce point correspondent aussi aux valeurs obtenues en optimisant chaque fonction objectif séparément [11].

$$pi_i \quad = \quad \min\{ \quad y_i \quad | \quad y \in ND(F) \quad \} \qquad (2.4)$$

2.7.2 Point Nadir

Les coordonnées du point Nadir (pn) correspondent aux pires valeurs de chaque objectif des points du front Pareto [11].

$$pn_i \quad = \quad \max\{ \quad y_i \quad | \quad y \in ND(F) \quad \} \qquad (2.5)$$

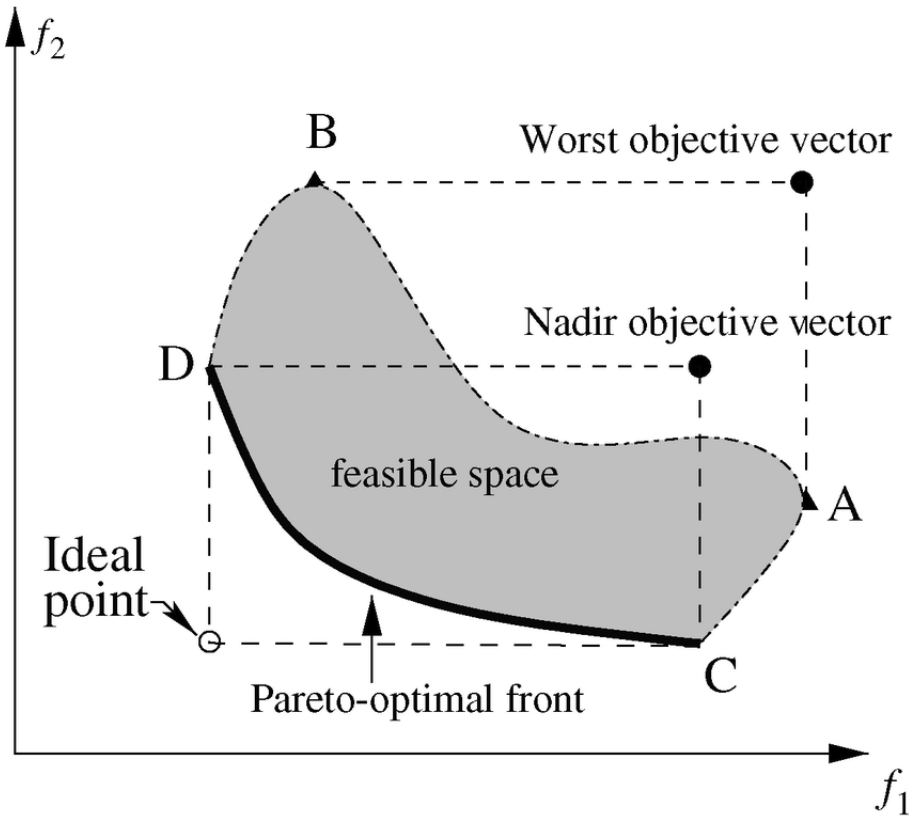

Figure 2.4 - Point Idéal et Point Nadir[19]

2.8 Classification des méthodes de résolutions des problèmes d'optimisation multi-objectifs

Dans la plupart des problèmes d'optimisations multiobjectifs (Multiobjective optimization problem, MOP), l'ensemble Pareto optimal est composé d'un grand nombre ou même infini de solutions.

Néanmoins, dans la pratique, une seule solution devrait être choisie parmi l'ensemble Pareto optimal. La personne qui a la tâche de choisir la solution la plus pratique de l'ensemble Pareto optimal est appelé le décideur (Decision Maker, DM). Les préférences du décideur sont incorporées afin d'induire un ordre total entre les solutions de Pareto. Il y a plusieurs approches dans lesquelles les préférences de DM peuvent être incorporées.

Par exemple, le DM peut classer l'ensemble des objectifs en fonction de leurs importances. Une autre possibilité est d'obtenir un échantillon du front de Pareto, puis, sélectionner une solution de cet échantillon. Une classification commune des techniques de résolution des MOPs prend en compte le moment où le DM est tenu de fournir des informations de préférence. La classification est la suivante :

Méthodes a priori Dans ce type de méthode où le décideur coopère avant le lancement du processus de résolution, le décideur doit avoir une bonne connaissance du problème traité et il doit fournir des préférences sur le problème à résoudre. Les informations données a priori aident la méthode de résolution dans sa recherche de solutions. Un exemple des méthodes a priori sont

les méthodes d'agrégation qui transforment un problème multiobjectif en un problème mono-objectif. La transformation peut se faire à l'aide d'un ensemble de poids sur les objectifs, ensuite on exécute une méthode classique qui fournira la solution recherchée en une seule exécution. Il faut noter que, le choix des poids doit se faire attentivement. Ainsi, la modélisation des préférences du décideur est une tache difficile et doit être prise en compte durant l'étape de résolution. Par ailleurs, dans la plupart des cas, un décideur peut ne pas être satisfait par les solutions obtenues après la résolution, ce qui l'oblige à relancer le processus par le biais d'une autre formulation de ses préférences [12].

Méthodes a posteriori Dans les méthodes a posteriori on cherche à trouver les solutions Pareto optimales sans que le décideur n'intervient dans le processus de résolution (il pourrait intervenir après la résolution).

Par conséquent, la méthode doit fournir au décideur la totalité de front Pareto trouvé, ensuite le décideur choisit les solutions qui lui semblent les plus appropriées. Ces méthodes ne nécessitent pas une connaissance préalable du problème, donc il n'est pas nécessaire de modéliser les préférences du décideur.

Par ailleurs, le nombre de solutions obtenues peut rendre l'ensemble Pareto optimal difficile à analyser pour le décideur de plus ces méthodes soufrent du même inconvénient que les méthodes a priori si le décideur n'est pas satisfait des solutions obtenues [12].

Méthodes interactives Les méthodes interactives visent à impliquer le décideur tout au long du processus de recherche.

Par conséquent, dans tout le processus de résolution le décideur coopère avec la méthode de résolution, le décideur peut définir ses préférences de façon claire et compréhensible. Le processus est itéré plusieurs fois jusqu'à la satisfaction du décideur. Par ailleurs, l'intervention directe du décideur dans le processus de résolution exige un long processus de recherche [12]

.

2.9 Quelques méthodes de résolution des problèmes d'optimisation multi-objectif

nous allons présenter les méthodes les plus connues et les plus utilisées dans la littérature a savoir la méthode de Somme pondérée, méthode ϵ-contraintes, une méthode hybride et la méthode du Goal Programming, la méthode d'agrégation .

2.9.1 Méthode E-contrainte

Cette méthode est basée sur la minimisation d'un fonction objectif f_i en considérant que les autres objectifs f_j avec $j \neq i$ doivent être inférieurs à une valeur ϵ_j en général, l'objectif choisir est celui que le décideur souhaite optimiser en priorité .Haimes.Y.Y et al en (1971)on suggéré la reformulation du problème d'optimisation sous forme [13] :

$$\begin{cases} min \quad f_i(x) \\ S.C \\ f_j \leq \epsilon_j, \quad j \neq i \end{cases} \qquad \ldots(2.6)$$

Et les paramètres ϵ_j sont définir par le décideur. En d'autres termes, une des fonctions qu'il faut optimiser est retenue comme unique objectif, tendis que les critères restants sont transformés en contraintes [13].

2.9.2 Méthode du Goal Programming

Cette méthode est également appelée Target vector optimisation. Le décideur fixe un but T_k à atteindre pour chaque objectif f_k $k \in \{1, 2, \cdots, r\}$, ces valeurs sont ensuite ajoutées au problème comme des contraintes supplémentaires . La nouvelle fonction objectif est modifiée de façon à minimiser la somme des écarts entre les résultats et les but à atteindre [13] :

$$min \quad \sum | f_k(x) - T_k | \quad , x \in X \quad \ldots(2.7)$$

T_k représente la valeur à atteindre pour le k^{eme} objectif.

La méthode est facile à mettre en œuvre mais la définition des poids et des objectifs à atteindre est une question délicate qui détermine l'efficacité de la méthode .Cette méthode a l'avantage de fournir un résultat même si un mauvais choix initial a conduit le décideur a donner un ou plusieurs but(s) T_k non réalisable [13].

2.9.3 Méthode d'agrégation

C'est l'une des premières méthodes utilisée pour la génération de solutions Pareto optimales.Elle consiste à transformer le problème (MOP) en un problème $((MOP)_\lambda)$ qui revient à combiner les différentes fonctions coût f_i du problème en une seule fonction objectif F généralement de façon linéaire (Hwang and Masud, 1979) [14] :

$$(MOP_\lambda) : \Big\{ \; minF(x) = \sum_{i=1}^{n} \lambda_i f_i(x) \lambda_i \in [0,1]; \sum_{i=1}^{n} \lambda_i = 1 \quad \ldots(2.8)$$

où les poids $\lambda_i \in [0,1]$ et $\sum_{i=1}^{n} \lambda_i = 1$

2.9.4 Somme pondérée

Dans cette méthode, l'idée générale est de faire associer un coefficient de poids pour chaque fonction objective et ensuite minimiser la somme pondérée de ces objectifs . De cette façon, le problème multiobjectif est transformé en un problème mono-objectif. Dans cette méthode, il est possible d'arriver à une solution optimale au sens de Pareto si les coefficients de poids sont positifs pour tous les objectifs, ou si le problème a une solution unique .

La méthode de la somme linéaire des poids peut générer des solutions Pareto optimales quelconque pour un problème d'optimisation multiobjectif convexe (c.-à-d. si toutes les fonctions objectifs et l'espace réalisable sont convexes)[12].

$$\min F(x) = \sum_{i=0}^{m} w_i f_i(x) \quad w_i \in [0,1] \quad \ldots(2.9)$$

2.9.5 Une Méthode hybride

La méthode hybride la plus connue qu'on va aborder ici est la méthode de Corley(1980) et Wendell and Lee(1977).Cette méthode utilise la méthode de sommes pondérées des fonctions objectif et la méthode de ϵ-contrainte.

Considérons le problème multiobjectif(MOP) le problème mono-objectif hybride qu'on note par(P) peut s'écrire comme suit [9] :

$$(P)\begin{cases} \min F(x) = \sum_{i=1}^{p} \lambda_i f_i(x) \\ f_i(x) \leq \epsilon_i \quad , \quad i = (1, 2, \ldots, p) \\ x \in S \end{cases} \qquad (2.10)$$

Avec $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)$ le vecteur de poids et $\forall i, \lambda_i \geq 0, \sum_{i=1}^{p} \lambda_i = 1$

Un sous-ensemble de solution efficace X_E du problème multiobjectif(MOP) peut être déterminer par la résolution du problème mono-objectif pour différentes valeurs de . Notons que les valeurs de ϵ peuvent être des bornes supérieures des objectifs f_i.

2.10 conclusion

Nous avons défini dans ce chapitre les problèmes d'optimisation multi-objectifs et les éléments de base . Nous avons présenté la Classification des méthodes de résolutions des problèmes d'optimisation multi-objectifs et Quelques méthodes de résolution.

Chapitre 3

classification des Algorithmes MOACO

3.1 introduction

Les colonies de fourmis dans la nature montrent une fascinante capacité a trouver des chemins entre des sources de nourriture et la fourmilière. A l'échelle d'une fourmi, le problème résolu est complexe tel que le plus court chemin. L'intelligence qui permet la résolution du problème est dite collective et est en fait le résultat émergeant d'un grand nombre d'interactions élémentaires. Les fourmis résolvent des problèmes complexes par des mécanismes assez simples a modéliser. Il est ainsi assez simple de simuler leur comportement par des algorithmes informatiques.

3.2 historique

Dans les sections qui viendront plus tard. nous présenterons la méta-heuristique ACO, pour le "*Ant colony optimisation*". toutes ces idées abstraites sont inspirées des travaux de Deneubourg sur les fourmis.

cette méta-heuristique est relativement récente. elle a été introduite en 1991 par Colonie, Dorigo et Maniezzo pour résoudre leproblème su voyageur de commerce . Elle s'est popularisée, puis à été l'objet d'améliorations dès 1995 et a été appliquée avec succès à d'autres problèmes d'optimisation dès 1994.

Nous allons tout d'abord exposer les différences et les points communs entre les fourmis virtuelles et les fourmis réelles,avant d'exposer en termes plus abstraits la méta-heuristique proprement dite. Ceci expliquera comment les fourmis virtuelles peuvent être exploitées pour résoudre un problème d'optimisation combinatoire.

3.3 Inspiration Biologique

Les éthologistes étudiant le comportement des insectes sociaux ont observé que la coopération au sein des colonies est auto-organisée, elle semble résulter uniquement des interactions entre les individus sans être supervisé d'une quelconque manière. Bien que ces interactions soient simples (par exemple, une fourmi se contente de suivre la trace odorante laissée par une autre), elles permettent à la collectivité de résoudre des problèmes difficiles, telle la recherche

du chemin le plus court entre le nid et une source de nourriture, parmi d'innombrables voies possibles.

En marchant du nid à la source de nourriture et vice-versa (ce qui, dans un premier temps, se fait essentiellement d'une façon aléatoire), les fourmis déposent au passage sur le sol une substance odorante appelée phéromone, ce qui a pour effet de créer une piste chimique. Les fourmis peuvent sentir ces phéromones qu'ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromone. Cela leur permet de retrouver le chemin vers leur nid lors du retour. Il a été démontré expérimentalement que ce comportement permet l'émergence des chemins les plus courts entre le nid et la nourriture, à condition que les pistes de phéromones soient utilisées par une colonie entière de fourmis.

Autrement dit, quand plusieurs chemins existent entre le nid et la nourriture, une colonie peut exploiter les phéromones déposés par des fourmis individuelles pour découvrir le chemin le plus court entre le nid et la nourriture .

Il semble donc que le comportement des fourmis réelles est un type de mécanisme d'optimisation distribué, à travers lequel chaque individu ne peut fournir qu'une très petite contribution. Il est intéressant de remarquer que, bien qu'une seule fourmi soit capable de construire une solution (i.e. de trouver un chemin du nid à la nourriture), c'est seulement le comportement du colonie qui crée le chemin le plus court. En un sens, l'optimisation est une propriété émergente de la colonie de fourmis [15].

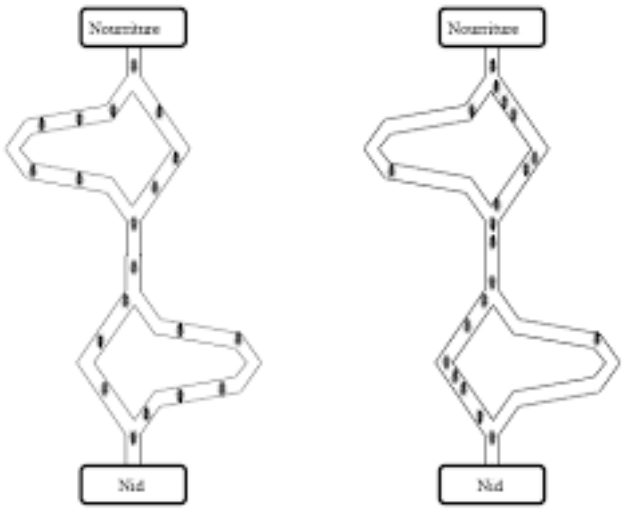

Figure 3.1 - Exemple d'une véritable colonie de fourmis [15]

3.4 Fourmis artificielles Vs fourmis réelles

Les fourmis artificielles(virtuelles) ont une double nature .

D'une part, elles modélisent les comportements abstraits de fourmis réelles et d'autre part,elles peuvent être enrichies par des capacités que ne possèdent pas les fourmis réelles , afin de les rendre plus efficaces que ces dernières .

Nous allons maintenant synthétiser ces ressemblances et différences.

3.4.1 Points communs

Les point communs entre les fourmis artificielles et les fourmis réelles sont[20] :

• **colonie d'individus coopérants** :comme pour les fourmis réelles , une colonie virtuelle est ensemble d'entités , qui se rassemblent ensemble pour trouver une bonne solution au problème considéré . chaque groupe d'individus doit pouvoir trouver une solution même si elle est mauvaise.

• **Piste de phéromone** : Ces entités communiquent par le mécanisme des pistes de phéromone cette forme de communication joue un grand rôle dans le comportement des fourmis , son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis , en fonction de l'historique laissé par ces phéromones.

• **Évaporation des phéromones** : La méta heuristique ACO comprend aussi la possibilité d'évaporation des phéromones . ce qui s'est passé avant. c'est ainsi qu'elle peut diriger sa recherche vers de nouvelles directions . sans être trop contrainte par ses anciennes décision .

• **Recherche du plus petit chemin** : Les fourmis réelles et virtuelles partagent un but commun : recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).

• **Déplacement locaux** : Les vraies fourmis ne sautent pas des cases , tout comme les fourmis virtuelles , elles se contentent de se déplacer entre sites adjacent du terrain.

• **Choix aléatoire lors des transitions** : Lorsqu'elles sont sur un site , les fourmis réelle set virtuelles doivent décider sur quel site adjacent se déplacer .cette prise de décision .se fait au hasard et dépend de l'information locale déposée sur le site courant. elle doit tenir compte des pistes de phéromones , mais aussi du contexe de départ (ce qui revient à prendre en considération les données du problème d'optimisation combinatoire pour une fourmi virtuelle).

3.4.2 Différences

Les fourmis artificielles possèdent certaines caractéristiques que ne possèdent pas les fourmis réelles[20] :

• **Elles vivent dans un monde non-continu** : Leur déplacement consistent en des transitions d'état.

• **Mémoire (état interne) de la fourmi** :Les fourmis réelles ont une mémoire très limitée.

Tandis que nos fourmis virtuelles mémorisent l'historique de leurs actions. Elles peuvent aussi retenir des données supplémentaires sur leurs performances.

• **Nature des phéromones déposées** : Les fourmis réelles déposent une information physique sur la piste qu'elles parcourent , là ou 'les fourmis virtuelles modifient des informations dans les variables d'états associées au problème . ainsi , l'évaporation des phéromones est une simple décrémentation de la valeur des variables d'états à chaque itération .

• **Qualité de la solution** :Les fourmis virtuelles une quantité de phéromone proportionnelle à la qualité de la solution qu'elles ont découverte.

• **Retard dans le dépôt de phéromone** : Les fourmis virtuelles peuvent peuvent mettre à jour les pistes de phéromone de façon non immédiate : souvent elles attendent d'avoir terminé la construction de leur solution . ce choix dépend du problème considéré bien évidemment .

• **Capacités supplémentaires** : Les fourmis virtuelles peuvent être pour vues de capacités artificielles afin d'améliorer les performances du système . ces possibilités sont liées au problème et peuvent être :

-*l'anticipation* : la fourmi étudie les états suivants pour faire son choix et non seulement l'état local.

-*le retour en arrière* : une fourmi peut revenir à un état déjà parcouru car la décision qu'elle avait prise à cet état à été mauvaise.

3.5 Le processus de base de l'algorithme de colonie de fourmis

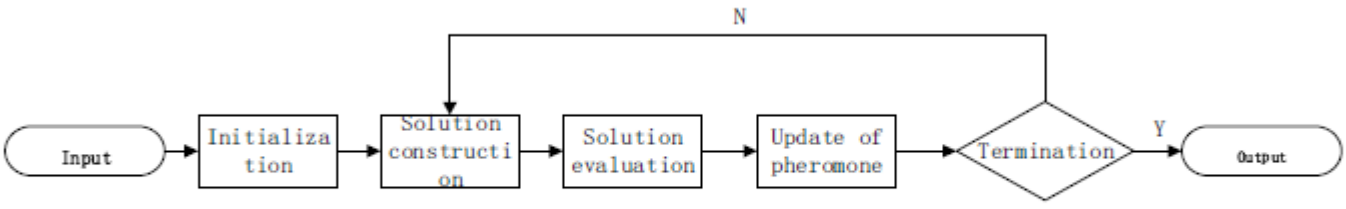

Figure 3.2 - Le processus de base de l'algorithme de colonie de fourmis[21]

Le processus de base de l'algorithme de colonie de fourmis est généralement divisé en plusieurs étapes :

•**Initialisation** : initialise les paramètres de l'algorithme, les informations phéromones et les informations heuristiques.

•**Construction de la solution** : pour chaque fourmi i, construisez une nouvelle solution en utilisant une règle probabiliste pour choisir les composants de la solution. La règle est fonction de la solution actuelle au sous-problème i, de la phéromone et des informations heuristiques.

• **Évaluation de la solution** : évaluez la solution de chaque fourmi obtenue à l'étape 2, stockez les solutions non dominées et éliminez les dominées.

• **Mise à jour des matrices de phéromones** : mettre à jour la matrice de phéromones en utilisant les informations extraites des solutions nouvellement construites. La phéromone liée aux bords dans une solution non dominée augmentera.

• **Terminaison** : si une condition d'arrêt spécifique au problème est remplie, telle que le nombre d'itérations et le temps d'exécution, l'algorithme s'arrête et génère l'ensemble de solutions non dominé, sinon, revenez à l'étape 2.

3.6 Algorithme système de fourmis

Le premier algorithme qui a été conçu pour le PVC porte le nom de système de fourmis (AS pour ***"Ant System"***). il s'agit d'un algorithme d'optimisation distribué qui à chaque itération **t**, chaque fourmi **k** parcourt les arêtes du graphe en se déplaçant d'un nœud à un autre construisant ainsi un trajet complet . Pour effectuer un déplacement , on doit définir [22] :

⇒ **La quantité de phéromone :** $\tau_{ij}(t)$ sur l'arête (i, j) reliant deux villes i et j . Elle donne l'intensité de la trace de phéromone artificielle sur le chemin (i, j) qui permet d'informer les fourmis sur l'importance de ce trajet . Ce paramètre dynamique définit l'attractivité ou encore la désirabilité de sélectionner la ville j comme une prochaine destination après la ville i . c'est , en quelque sorte , une mémoire globale du système , qui évolue par apprentissage .

⇒ **La visibilité :** η_{ij} qui présente l'inverse de la distance entre les villes . Elle permet de diriger le choix des fourmis vers des villes proches et d'éviter les villes lointaines. il s'agit d'une heuristique d'information pour choisir la ville j à partir de la ville i .

⇒ **une liste** , dite liste tabou L_k constituant une mémoire des villes déjà visitées pour éviter que la fourmis k , située au sommet i , revisite une même ville . de plus , elle permet de sauvegarder le chemin parcouru par cette fourmi .

A chaque itération t , chaque fourmi k choisit sa prochaine destination j suivant une probabilité qui dépend de la distance séparant cette ville et sa position i et de la quantité de

phéromone présente sur cette arête . la règle de transition d'une ville i vers une ville j est donnée par la relation (3.1) suivante :

$$\begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in L_k} [\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}} & \text{si } \; j \notin L_k(i), \\ 0 & \text{sinon} \end{cases} \qquad (3.1)$$

où α et β sont deux paramètre modulant l'importance relative entre l'intensité de la quantité de phéromone τ_{ij} et la visibilité . le choix des valeurs de ces deux paramètres permet de régler l'influence des comportements de diversification et d'intensification .

Trois types d'algorithmes AS ont été proposés : le ***"ant-density "***, le ***"ant-quantity"*** et ***le ant-cycle"***. selon le type de l'algorithme , le dépôt de phéromones se fait pendant la construction de la solution pour les deux premières approches. dans l'autre cas , c'est-à-dire le "ant-cycle", chaque fourmi , après avoir terminé la construction de sa solution , laisse une quantité de phéromones $\Delta\tau_{ij}^{k}$ sur le chemin choisit . cette quantité dépend de la qualité de la solution construite . la valeur de $\Delta\tau_{ij}^{k}$ diffère d'un algorithme à l'autre.

D'après les résultats donnés colorni, l'algorithme "ant-cycle" est le meilleur . la quantité de phéromones est donnée dans ce cas par la relation suivante :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L^{k}(t)} & \text{si } \; (i,j) \in T^{k}(t) \\ 0 & \text{sinon} \end{cases} \qquad (3.2)$$

Avec Q une constante positive donnée , L^k la longueur de la solution (qui est la somme des distances d'une ville à une autre) et T^k le trajet effectué par la fourmi k.

Lorsque toutes les fourmis terminent leurs tournées , il est important de mettre au point un processus d'évaporation de la quantité de phéromone afin de donner de l'importance au derniers choix réalisés .Ceci permet d'oublier les mauvaises solutions . La règle de mise à jour est donnée par la relation suivante :

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \sum_{k=1}^{nbant} \Delta\tau_{ij}^{k}. \qquad (3.3)$$

Le paramètre ρ est ainsi utilisé pour éviter l'accumulation illimitée de phéromone et permet à l'algorithme d'oublier les mauvaises décisions précédemment prises.Sur les arêtes qui n'ont pas été choisis par les fourmis la force associée va décroitre rapidement avec le nombre d'itérations .

La valeur initiale des τ_{ij} est $\tau_0 \geq 0$ (la valeur de τ_0 est généralement très petite) elle nombre de fourmis est proposé égale au nombre des villes . L'algorithme suivant donne la struc-

ture du pseudo-code de la l'algorithme AS pour le PVC [**22**].

Algorithme AS pour le PVC :

Début

Initialisation

placer aléatoirement chaque fourmi k sur un nœud i

Initialiser les traces de phéromone à τ_0

Initialiser L_k

Déterminer les différents paramètrs de l'algorithme

Pour t=1 à t_{max} **Faire**

Pour chaque fourmi k=1 à nbAnt **Faire**

pour chaque ville non visitée **Faire**

sélectionner une ville $j \notin L_k(i)$ selon la formule (3.1)

Ranger j dans $L_k(i)$

Fin Pour

calculer la longueur L_k du chemin décrit par la fourmi k

Déposer une piste $\Delta\tau_{ij}^k$ sur le parcours τ_k de la fourmi k selon la formule(3.2)

Fin Pour

3.7 Extensions de AS

Malgré ses résultats encourageants , AS n'était pas compétitif avec les algorithmes de l'état de l'art du PVC.des recherches sont alors entreprises pour l'étendre et essayer d'améliorer ses performances en contrôlant mieux l'intensification et la diversification de la recherche.

Algorithmes système de colonie de fourmis

L'algorithme système de colonie de fourmis(ACS pour "Ant Colony System") est une variante du premier algorithme AS proposé pour résoudre le PVC de grandes tailles. cette méthode

repose sur les principes suivants[22] :

⇒ **Une règle de transition**(*dite aussi règle proportionnelle pseudo-aléatoire*) dépendant d'un paramètre $q_0(0 \leq q_0 \leq 1)$ qui définit une balance diversification/intensification.

Ainsi , une fourmi k se trouvant sur une ville i choisira une ville j selon la relation :

$$j = \left\{ \begin{array}{ll} arg\max_{iu \in L_{k(1)}}(t)] * [\eta_{iu}]^{\beta}\} & \text{si} \quad q \leq q_0 \\ j & \text{si} \quad q > q_0 \end{array} \right\} \qquad (3.4)$$

β est un paramètre modulante l'importance relative entre l'intensité de la quantité de phéromones τ_{ij} et la visibilité. la variable q est choisie de manière aléatoire uniformément distribuée sur [0,1] . si $(q \leq q_0)$, le système tend vers une intensification, c'est-à-dire exploiter plus les informations récoltées par le système . Ainsi , le choix de trajet non exploré est faible.

Dans le cas contraire ,ou $(q \geq q_0)$,le choix se fait de la même façon que pour l'algorithme AS , et le système tend à effectuer une diversification. Ainsi , j est sectionné aléatoirement selon la probabilité(3.5) :

$$p_{r_{ij}} = \left\{ \frac{[\tau_{iu(t)}]*[\eta_{iu}]^{\beta}}{\sum_k u \notin L(i)[\tau_{iu}(t)] * [\eta_{iu}]^{\beta}} \right. \qquad (3.5)$$

⇒ **Une mise à jour locale** : en construisant , chaque fourmi k va modifier la quantité de phéromone sur chaque arête traversée en utilisant la relation(3.6) :

$$\tau_{ij}(t+1) = (1-\rho_l) * \tau_{ij}(t) + \rho_l * \tau_0 \qquad (3.6)$$

ou τ_0 est la valeur initiale de la piste de phéromone et ρ_l est le coefficient d'évaporation locale de phéromone. a chaque passage , la quantité de phéromones sur les arêtes visitées diminue, favorisant ainsi la diversification par la prise en compte des trajets non explorés.

⇒ **Une mise à jour globale** : qui s'effectue à chaque itération de la façon suivante :

$$\tau_{ij}(t+1) = (1-\rho_g) * \tau_{ij}(t) + \rho_g * \Delta\tau_{ij}^{k}(t) \qquad (3.7)$$

$$\left\{ \begin{array}{ll} (l_{gb}) - 1, & \text{si}(i,j) \in \text{au meilleur tour de longueur} l_{gb} \\ 0 & sinon \end{array} \right. \qquad (3.8)$$

Dans ce cas , seule la meilleure solution , de longueur , obtenue est mise à jour , ce qui participe à une intensification par sélection de la meilleure solution .

→ le système utilise une liste de candidat pour chaque fourmi k , notée S_k , représentant la liste des choix possibles à partir d'une ville .

elle permet de restreindre la liste des villes les plus proches non encore sélectionnées pour accélérer le processus de construction d'un chemin , et réduire le temps nécessaire au calcul des probabilités.

→ le système dispose aussi d'une liste tabou , pour sauvegarder le chemin parcour par chaque fourmi.

la description de cette approche est décrire dans algorithme ACS pour le PVC [**22**].

Algorithme CS pour le PVC :

Début

initialisation

placer aléatoirement chaque fourmis sur un nœud i

initialiser les traces de phéromone à τ_0 ,les listes $S_k(i) et(i)$

Déterminer les différents paramètres de l'algorithme

pour t=1 à t_{max}**Faire**

pour chaque fourmi k=1 à nb Ant **Faire**

pour chaque ville non visitée **Faire**

sélectionner une ville $j \in S_k(i)$ selon la formule (3.4)

Ranger j dans $l_k(i)$

mettre à jour localement la quantité de phéromone selon la formule(3.6)

Fin pour

Évaluer l_k

Fin pour

Mettre à jour globalement les quantités de phéromone de la meilleure solution obtenue selon la formule(3.7)

Fin pour

Fin

3.7.1 Algorithme MAX-MIN système de fourmis

L'algorithme MMAS a été proposé pour améliorer les performances de ACO , combine une exploitation améliorée des meilleur solutions trouvées durant la recherche avec un mécanisme efficace pour éviter une stagnation prématurée de la recherche .MMAS diffère en trois aspects clé de AS [22] :

⇒ pour exploiter les meilleures solutions trouvées durant une itération ou durant l'exécution de l'algorithme, après chaque itération , une seule fourmi ajoute de la phéromone. cette fourmi peut être celle qui a trouvé la meilleure solution depuis le début de l'exécution , ou bien celle durent le dernier cycle.

⇒ pour éviter une stagnation de la trace de phéromone sur chaque composant de solution sont limitées à l'intervalle$[\tau_{max}, \tau_{min}]$.

⇒ De plus , les traces de phéromone sont initialisées à τ_{max} pour assurer une exploration élevée de l'espace des solutions au début de l'algorithme.

3.8 MOACO Algorithmes

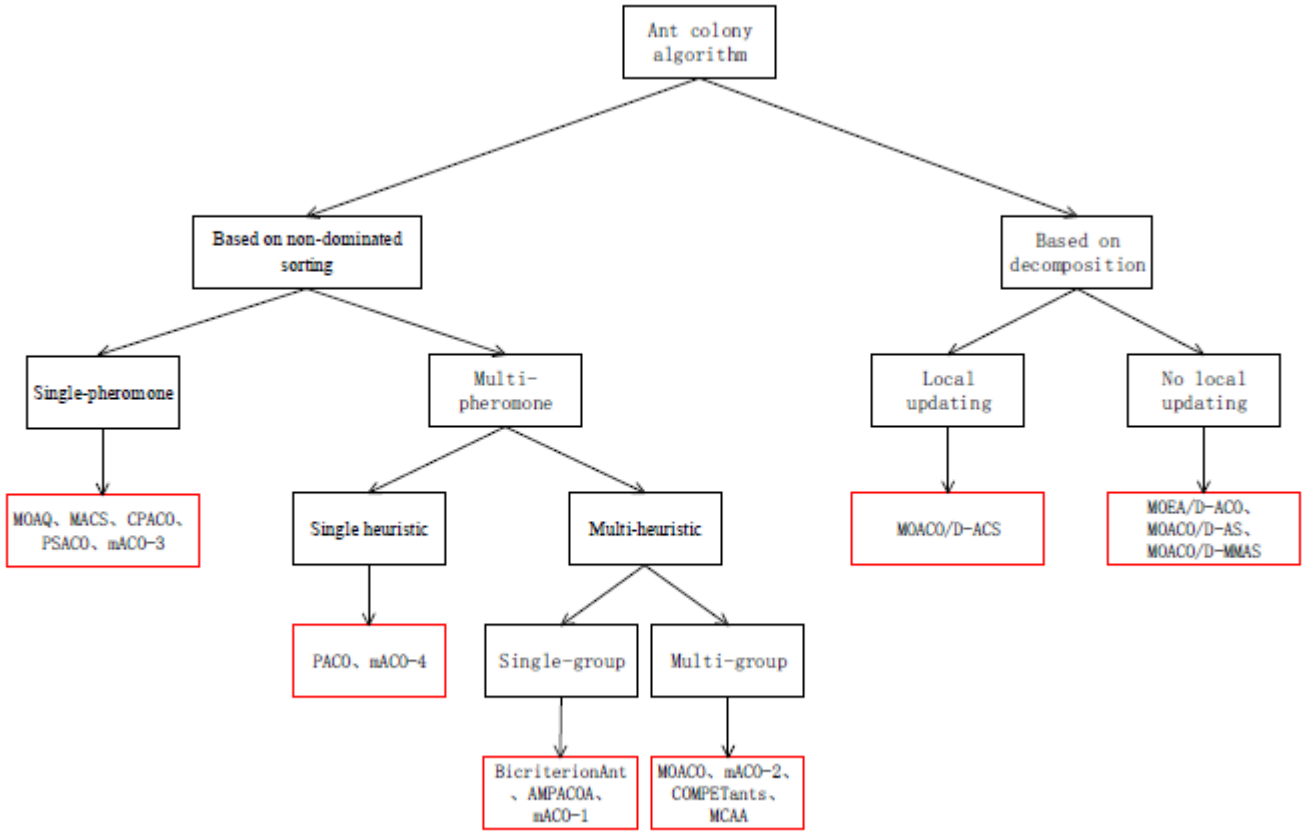

Figure 3.3 - Classification de MOACO algorithmes [21]

Comme le montre dans la figure (3.2) , l'algorithme de colonie de fourmis multi-objectifs existant est divisé en deux types. L'un est basé sur le tri non dominé, et l'autre est basé sur la

décomposition .

Différents types d'algorithmes ont de grandes différences dans les méthodes de construction. Les mêmes algorithmes de classe ont également quelques différences. Parmi tous les nœuds feuilles, nous sélectionnons un ou deux algorithmes pour introduire son principe de construction, et le comparer avec d'autres types d'algorithmes.

3.9 Algorithmes multi-objectifs de colonies de fourmis basés sur un tri non dominé

Ce type d'algorithme adopte principalement la stratégie du tri non dominé dans l'évaluation de la solution. La solution des solutions non dominées formées par cette stratégie peut ne pas être uniforme, ce qui correspond à une partie de PF.
Ce type d'algorithme est divisé en deux types selon le nombre de matrices de phéromones utilisées.

3.9.1 Matrice à phéromone unique

Ces algorithmes incluent principalement [21] **MOAQ** , **MACS** , **CPACO** , **PSACO** , **mACO-3** etc.

L'algorithme **MOAQ** utilise deux matrices heuristiques et deux poids (0,1). Quand l'heuristique correspond au vecteur de poids (0,1) et (1,0), on en choisit une parmi les deux matrices heuristiques.MACS peut être vu comme une version variante de MOAQ. La seule différence entre eux est le nombre des vecteurs de poids dans l'agrégation des informations heuristiques. MOAQ utilise deux vecteurs de poids (0,1) et (1,0), tandis que MACS utilise plus de deux vecteurs de poids. Le nombre différent de poids a une grande influence sur la solution.

L'algorithme **CPACO** utilise une règle de transition d'état et une règle de mise à jour des phéromones différentes de MOAQ et MACS. Et la partie incrémentale est liée au niveau de chaque solution non dominée.

L'algorithme **PSACO** est basé sur le système de fourmis (colonie de fourmis, AS). Le plus grand changement concerne le modèle de règle de mise à jour des phéromones.Deux ensembles de solutions existent, L'un est l'ensemble de solutions non dominé généré. L'autre est un nombre fixe de solutions optimales globales. Grâce à ces deux ensembles de solutions, la qualité de la solution de chaque ensemble de solutions non dominé est calculé et le résultat est appliqué à la formule de mise à jour des phéromones en tant que paramètre.

Le **mACO-3** utilise une matrice de phéromone, et son innovation est que la phéromone sur le même bord ne peut être mis à jour une fois lors de l'itération à travers une phéromone.

Nous choisissons un algorithme représentatif de ce type d'algorithme [4] :

•**Algorithme MOAQ** : L'algorithme " Multiple Objective $Ant-Q$ " (MOAQ) a été proposé par **Mariano** et **Morales** pour résoudre le problème de distribution de l'eau dans les réseaux d'irrigation. Cet algorithme est basé sur un algorithme d'apprentissage renforcé distribué appelé

Ant-Q .

L'idée de base de MOAQ est de réaliser un algorithme d'optimisation avec une famille d'agents (fourmis) pour chaque objectif. Chaque famille k essaye d'optimiser un objectif en considérant les solutions trouvées pour les autres objectifs et leur fonction correspondante HE^k. De cette manière, tous les agents des différentes familles travaillent dans le même environnement en proposant des actions et une valeur de récompense r qui dépend de comment leurs actions ont participé à trouver des solutions de compromis parmi les autres agents.

MOAQ présente d'autres caractéristiques spécifiques. D'abord, lors de la construction de solution,la j^{eme} fourmi de la i^{eme} famille utilise la solution trouvée par la j^{eme} fourmi de la famille $i-1$. De plus, lorsqu'une solution trouvée n'est pas faisable, l'algorithme applique une pénalité à ses composants sur les valeurs de Q. Finalement, tout au long du processus, les solutions non-dominées sont enregistrées dans un ensemble externe.

3.9.2 Matrice multi-phéromones

Ces algorithmes sont divisés en deux types en fonction du nombre de matrices heuristiques utilisées.

1) Matrice heuristique unique

la matrice heuristique unique signifie que toutes les fourmis de l'algorithme partagent une matrice heuristique dans le processus de construction de la solution, y compris PACO, mACO-4, etc.

L'algorithme PACO est mis à jour d'une manière spéciale, en utilisant la meilleure solution et les solutions de deuxième choix pour mettre à jour les informations sur les phéromones. La méthode d'agrégation de phéromones utilisée dans mACO-4 est la même que mACO-1, mais mACO-4 utilise une seule matrice d'informations heuristiques [21].

- **Algorithme PACO :** "Pareto Ant Colony Optimization" (P-ACO) a été proposé par Doerner et al. Il a été initialement appliqué au problème de sélection de portefeuille multi-objectif. Il suit le schéma général de l'algorithme ACS, mais la mise à jour globale de phéromone est réalisée en utilisant deux fourmis, la meilleure et la deuxième-meilleure solution générée dans le cycle courant pour chaque objectif k.

Dans P-ACO, plusieurs structures phéromone T^k sont considérées, une pour chaque objectif. A chaque cycle de l'algorithme, chaque fourmi calcule un ensemble de poids $p = (p_1, ..., p_k)$, et l'utilise pour combiner les traces de phéromone et l'information heuristique.

Les solutions non-dominées trouvées tout au long de l'exécution sont là aussi enregistrées dans un ensemble externe[4].

2) Matrice multi-heuristique

ce type d'algorithme utilise un certain nombre de matrices de phéromones et de nombreuses matrices heuristiques. Selon le nombre de groupes, ces algorithmes sont divisés en mono-groupe

et multi-groupe [21].

a) Groupe unique : les algorithmes représentatifs de ce type d'algorithmes sont BicriterionAnt , AMPACOA , mACO-1 , etc.

Chaque fourmi dans **BicriterionAnt** a un vecteur de poids pour agréger les informations sur les phéromones et les informations heuristiques.

AMPACOA est basé sur l'algorithme de colonie de fourmis de première génération AS, et sa caractéristique est sa manière de mettre à jour les phéromones. Cet algorithme attribue un poids à chaque solution non dominée en fonction de la durée d'ajout de la solution non dominée. Le poids est utilisé pour calculer les coefficients de cette solution non dominée, et le coefficient de chaque solution non dominée est lié à l'incrément de phéromone.

Le **mACO-1** utilise de nombreux groupes de fourmis. Chaque groupe de fourmis est responsable d'un objectif, et il a également un groupe de fourmis supplémentaire. Il agrège les informations sur les phéromones et les informations heuristiques de chaque groupe de fourmis de manière aléatoire par la somme pondérée. La meilleure solution pour chaque groupe est utilisée pour mettre à jour les matrices de phéromones respectives. La solution optimale pour chaque objectif généré par le groupe supplémentaire est de mettre à jour les matrices de phéromones des autres groupes.

- **Algorithmes BicriterionAnt** :

L'algorithme BicriterionAnt a été proposé par Iredi et al.spécialement pour résoudre le problème de tournée de véhicules bi-critère. Pour ce faire, il utilise deux structures de traces de phéromone différentes,τ et τ' une pour chaque critère considéré.

A chaque génération, chacune des m fourmis de la colonie génère une solution au problème. Durant l'étape de construction, la fourmi choisit le prochain sommet j à visiter relativement à la probabilité suivante [4] :

$$p(j) = \frac{\tau_{ij}^{\lambda\alpha}.\tau_{ij}^{'(1-\lambda)\alpha}.\eta_{ij}^{\lambda\alpha}.\eta_{ij}^{'(1-\lambda)\alpha}}{\sum_{u\in\Omega}\tau_{iu}^{\lambda\alpha}.\tau_{iu}^{'(1-\lambda)\alpha}.\eta_{iu}^{\lambda\alpha}.\eta_{iu}^{'(1-\lambda)\alpha}} \qquad (3.9)$$

où η_{ij} et η'_{ij} ij sont les valeurs heuristiques associées à l'arête a_{ij} relativement au premier et second objectif, respectivement, Ω est le voisinage faisable de la fourmi, et λ est calculé pour chaque fourmi h, $f\epsilon 1,...,m,$ comme suit :

$$\lambda_h = \frac{h-1}{m-1} \qquad (3.10)$$

Une fois que toutes les fourmis ont construit leurs solutions, les traces de phéromone sont évaporées par la règle habituelle. Ensuite, chaque fourmi qui a généré une solution dans le front Pareto au cycle courant est autorisée à mettre à jour les deux structures phéromone τ etτ' , en déposant une quantité égale à $\frac{1}{l}$, où l est le nombre de fourmis qui sont en train de mettre à jour les traces de phéromone. Les solutions non-dominées générées tout au long de l'exécution de l'algorithme sont mises dans un ensemble externe.

b) Multi-groupe : ce type d'algorithme diffère du groupe unique en utilisant plus d'un groupe de fourmis, y compris **MOACO** , **mACO-2** , **COMPETants** , **MACC** et ainsi de suite.

L'algorithme **MOACO** utilise plusieurs poids, mais si le nombre de poids est inférieur au nombre de fourmis, le même poids peut être utilisé par différentes fourmis. La phéromone utilisée par **mACO-2** est la somme des phéromones de tous les groupes, plutôt que la méthode de la somme pondérée utilisée par mACO-1. L'algorithme **COMPETants** divise les fourmis en trois groupes, chacun avec un poids w = (0,0.5,1). Chaque groupe de fourmis utilise son vecteur de poids pour agréger les matrices de phéromones et la matrice heuristique pour construire la solution.

L'algorithme **MACC** divise les fourmis en plusieurs groupes. Chaque groupe est responsable d'un objectif et possède ses informations heuristiques et ses informations sur les phéromones. La matrice de phéromones de chaque groupe est mise à jour en fonction de la meilleure solution générée dans l'itération en cours[21].

• **Algorithme MOACO** : l'algorithme MOACO utilise plusieurs matrices de phéromones et plusieurs matrices heuristiques. Chaque matrice de phéromones et matrice heuristique est responsable d'un objectif.

Toutes les fourmis sont divisées en plusieurs groupes. Chaque groupe a plusieurs poids et chaque fourmi du groupe a un vecteur de poids. Si le nombre de poids dans le groupe est inférieur au nombre de fourmis, les fourmis supplémentaires se voient attribuer des poids depuis le début. C'est-à-dire qu'il est possible que deux fourmis ou plus d'un groupe utilisent le même vecteur de poids. La fourmi utilise son vecteur de poids pour agréger les informations sur les phéromones et des informations heuristiques par la méthode du produit pondéré. Il calcule ensuite la probabilité que la ville non visitée se déplace et choisit la prochaine ville à visiter par sélection à la roulette pour construire la solution.

Enfin, il utilise la solution non dominée générée par l'itération courante pour mettre à jour les informations sur les phéromones.

3.10 Algorithmes multi-objectifs de colonies de fourmis basés sur la décomposition

Ce type d'algorithme décompose un problème d'optimisation multi-objectif en plusieurs problèmes d'optimisation à objectif unique. sous-problèmes. Chaque sous-problème a un poids. De cette façon, résoudre un problème multi-objectif est transformé en résolution de plusieurs sous-problèmes à objectif unique, et chaque sous-problème a une meilleure solution. La distribution de la solution non dominée par cette stratégie est relativement uniforme, mais le nombre de solutions non dominées est faible. Cet algorithme n'a pas de mise à jour locale dans le processus de construction de la solution. Selon cela, ce type d'algorithme est divisé en deux types : mise à jour locale et pas de mise à jour locale.

3.10.1 Mise à jour locale

Cet algorithme de colonie de fourmis multi-objectifs est basé sur la décomposition et les fourmis utilisent la règle de mise à jour locale lorsqu'elles construisent la solution. Le représentant de l'algorithme est MOACO/D-ACS .

• **Algorithme MOACO/D-ACS** : cet algorithme utilise l'approche de Tchebycheff pour décomposer un problème multi-objectif en plusieurs sous-problèmes à objectif unique. Chaque

sous-problème se voit attribuer un vecteur de poids, et cet algorithme préserve les matrices de phéromones agrégées et la matrice heuristique avant de construire la solution.

Chaque fourmi peut résoudre plusieurs sous-problèmes dans une itération, et plus d'une fourmi peut résoudre un sous-problème. Lors de la construction de la solution, générez un nombre uniformément aléatoire à partir de (0,1). S'il est inférieur à un paramètre de contrôle, choisissez la ville avec la plus grande probabilité. Sinon, sélectionnez la ville en fonction de la sélection de la roulette. Lors de la mise à jour de la phéromone de les sous-problèmes, utilisez la meilleure solution de ce sous-problème.

Étant donné que dans chaque processus d'itération, une fourmi peut résoudre plusieurs sous-problèmes, il en résultera plusieurs solutions. Il peut donc y avoir différentes phéromones qui sont mises à jour par la même solution, puis le partage de phéromones entre les sous-problèmes est mieux réalisé.

L'algorithme MOACO/D-ACS ajoute une règle de mise à jour locale. Les fourmis visitent les bords et changent leur niveau de phéromone en appliquant la règle de mise à jour locale.

L'idée principale est de réduire la probabilité d'être sélectionné par d'autres fourmis, d'encourager d'autres fourmis à explorer les nouveaux bords ainsi que d'augmenter la diversité de la sélection des fourmis[21].

3.10.2 Pas de mise à jour locale

Aucun algorithme de mise à jour locale n'utilise la règle de mise à jour locale lorsque les fourmis construisent les solutions. Il s'agit principalement de **MOEA/D-ACO** ,**MOACO/D-AS**, **MOACO/D-MMAS**.

L'algorithme **MOEA/D-ACO** alloue un certain nombre de fourmis voisines pour chaque fourmi en fonction de la distance euclidienne des poids des sous-problèmes. Cela facilite la communication entre les voisins marginaux des différents groupes et renforce le lien au sein du groupe.

L'algorithme **MOACO/D-AS** est basé sur la décomposition et il combine les avantages de l'algorithme MOACO/D avec l'algorithme de colonie de fourmis de première génération AS.

L'algorithme **MOACO/D-MMAS** définit la valeur maximale et minimale de la phéromone et initialise la phéromone à la valeur maximale.

• **Algorithme MOEA/D-ACO :** après l'algorithme de colonie de fourmis basé sur la décomposition, Liangjun Ke et al. combiné l'algorithme évolutif multiobjectif (EA) avec un algorithme de colonie de fourmis basée sur la décomposition évolutive.

Cet algorithme décompose un problème d'optimisation multi-objectif en un certain nombre de problèmes d'optimisation à objectif unique par l'approche de Tchebycheff. Chaque Le sous-problème a un vecteur de poids, une matrice d'information heuristique et une solution optimale. Selon la distance euclidienne des poids des sous-problèmes, les fourmis sont divisées en groupes et chaque fourmi a des voisins. Chaque groupe a sa matrice de phéromones et les fourmis du même groupe partagent cette matrice de phéromones.

Chaque fourmi est responsable de la résolution d'un sous-problème. Les fourmis construisent un nouveau solution en agrégeant la phéromone de son groupe, l'information heuristique du sous-problème et la solution optimale du sous-problème.

Selon la fonction objectif du sous-problème, chaque fourmi utilise la nouvelle solution construite par ses voisines pour mettre à jour sa solution optimale. Cela permet la collaboration entre différents groupes de fourmis, puisque deux voisins peuvent ne pas être dans le même groupe [21].

3.11 conclusion

Nous avons présenté dans cette section la classification des Algorithmes MOACO et plusieurs algorithmes MOACO proposés dans la littérature. Ces algorithmes proposent différentes stratégies pour la résolution des problèmes multi-objectifs.

Chapitre 4

Discussion des expérimentations

4.1 introduction

Dans ce chapitre, nous utiliserons le problème TSP multi-objectifs standard comme cas d'étude et cas test pour analyser les performances des MOACO.

Nous notons seulement que les expériences et les résultats discutés ne sont pas nos résultats et ont été élaborés conformément à l'article [21].

4.2 le problème TSP multi-objectifs

TSP multi-objectifs : Il est défini comme suit : un homme d'affaires itinérant part d'une ville. Chaque ville doit être visitée et ne peut être visitée qu'une seule fois. Après avoir visité toutes les villes, il doit retourner à la ville de départ. Ce problème consiste à trouver le chemin le plus court. le problème TSP multi-objectifs est pris comme exemple, et son organigramme est un graphe non orienté. Afin de mieux comprendre le problème TSP multi-objectifs„ nous avons l'exemple suivant :

exemple

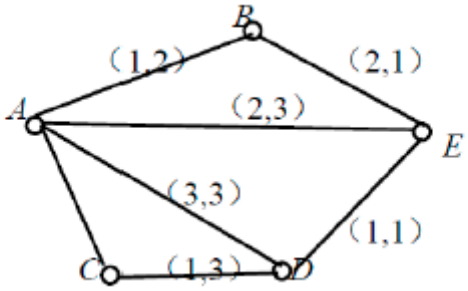

Figure 4.1 - Exemple de problème TSP multi-objectifs [21]

Dans cet exemple le graph n'est pas orientée c'est à dire que le (1,2) sur l'arête (A → B) qui représente deux valeurs d'objectif différentes (par exemple, la distance et le temps) du problème TSP multi-objectif est la même sur l'arête (B → A) En supposant qu'un homme d'affaires se déplace de la ville A et visite toutes les villes du graphique une seule fois et retourne à la ville A et ici Nous voulons trouver toutes les solutions non dominées parmi les chemins possibles.

D'après la figure (4.1)

Le processus de résolution de ce problème par l'algorithme de colonie de fourmis est le suivant :

1) **Initialisation** : initialise les paramètres de l'algorithme, les informations phéromones et les informations heuristiques. Initialiser l'objectif 1, l'objectif 2 et la phéromone t sur chaque bord. Tels que l'arête (A, B) et initialiser les paramètres : α, β, λ, ρ etc. Selon le poids λ^i, calculer la matrice de phéromones τ^i et la matrice heuristique η^i de chaque fourmi.

2) **Construction de la solution** : chaque fourmi construit une solution basée sur une fonction de probabilité P consistant d'informations sur les phéromones et d'informations heuristiques. Par la fonction de probabilité suivante, chaque fourmi i construit un chemin selon ses τ^i et η^i

$$P^i_{m,n} = \frac{(\tau^i_{m,n})^\alpha (\eta^i_{m,n})^\beta}{\sum_{l \in C} (\tau^i_{m,l})^\alpha (\eta^i_{m,l})^\beta} \qquad (4.1)$$

où C représente un ensemble de toutes les villes qui sont connectées à la ville m et non visitées par le fourmi i .

3)**Évaluation de la solution** : évaluez la solution de chaque fourmi à l'étape 2. Stockez la solution non dominée dans l'ensemble de solutions non dominées en fonction de la relation de dominance. De plus éliminer la solution dominée.

4)**Mise à jour des matrices de phéromones** : mise à jour des informations de phéromone τ sur toutes les arêtes : Si l'arête est dans la solution non dominée, elle est mise à jour par incrément de phéromone. Sinon, volatile la phéromone, c'est-à-dire que l'incrément de phéromone est de 0. La formule est la suivante :

$$\tau^i_{m,n} = \rho \tau^i_{m,n} + \Delta\tau \qquad (4.2)$$

où ρ est le facteur de volatilisation et $\Delta\tau$ est l'incrément de phéromone.

5)**Résiliation** : déterminer le nombre d'itérations, qu'elles aient atteint la valeur maximale ou que le temps d'exécution soit terminé. S'il n'y a pas de fin, revenez à l'étape 2 pour continuer. Sinon, sortez l'ensemble des solutions non dominées, puis l'algorithme est terminé.

4.3 Discussion et comparaison

Selon une comparaison effectuée, a comparaison de ces algorithmes se fait en résolvant le problème TSP multi-objectif spécifique. Dans l'expérience, nous avons sélectionné les sept algorithmes décrits ci-dessus : MACS, PACO, MOACO, MOACO/D-ACS, MOEA/D-ACO, et neuf cas de test TSP multi-objectifs standard kroAB100, kroAC100, kroAD100, kroCD100, kroCE100, euclidAB100 , euclidCE100, kroAB150, kroAB200

Les paramètres de ces algorithmes [21] :

Les paramètres **MOEAD/ACO** sont définis comme $\alpha = 1$, $\beta = 10$, ρ= 0.95 , $\tau = 0.9$ et T = 10, où T est la taille de l'essaim.

Les paramètres **MACS** sont définis comme $\alpha = 0.9$, $\beta = 1$, $\rho = 0.1$, T = 10

Les paramètres **PACO** sont définis comme $\alpha = 1.0$, $\beta = 1.0$, $\rho = 0.1$, $\tau = 1.0$, T = 10.

les paramètres **MOACO** sont définis comme $\alpha = 1.59$, $\beta = 2.44$,$\rho = 0.54$, $\tau = 0.69$, T = 10

Les paramètres **MOACO/D** sont définis comme $\alpha = 1.0$, $\beta = 2.0$, $\rho = 0.02$, $\Gamma = 2$, N = 20 , où Γ est le nombre de fourmis dans chaque sous-colonie et N est le nombre N de sous-problèmes.

La performance de l'algorithme a une grande relation avec le nombre de fourmis et la taille de la ville. Afin d'éviter que l'algorithme n'entre prématurément dans l'optimum local, il est important de laisser les fourmis explorer de nouvelles voies. La formule de terminaison de l'algorithme généralement utilisée est la suivante : $N \times 300 \times (n/100)/2$. Où N représente le nombre de fourmis et n représente le nombre de villes.

Toutes les expérimentations sont menées dans le même environnement et implémentées en langage C, Nous exécutons 30 fois chaque algorithme pour résoudre le même problème et comparons l'ensemble de solutions non dominées avec les autres algorithmes par l'indicateur d'hyper-volume (indicateur H) et le test d'hypothèse. Parallèlement, la convergence de ces algorithmes comprenant MOEA/D-ACO, MACS, MOACO/D-ACS, MOACO et PACO a été analysée sur les cas de test comprenant kroAB100, kroAB150 et kroAB200 de différentes tailles.

La performance de ces algorithmes est évaluée sur la base de l'indicateur d'hyper-volume, car il possède la caractéristique hautement souhaitable d'une stricte conformité Pareto. Chaque fois qu'une approximation domine complètement une autre approximation, l'hyper-volume de la première sera supérieur à l'hyper-volume de la seconde.

4.3.1 Comparaison basée sur l'indicateur H

Dans cette section, chaque algorithme s'exécute 30 fois de manière indépendante en fonction du meilleur ensemble de paramètres dans leur article sous les neuf cas de test. Puis nous transformons les résultats expérimentaux de chaque algorithme en une valeur d'indicateur H. Après cela, les performances de chaque algorithme sont analysées par un diagramme en boîte et un tableau moyenne-variance.

Les box plots basées sur l'indicateur H sont présentées à la figure(4.2) . Les indicateurs H ont été obtenus par sept algorithmes résolvant les neuf TSP standard à objectifs multiples. Il y a cinq lignes pleines et une ligne pointillée dans la box plots. Le trait plein de haut en bas représente successivement le point maximum (il peut y avoir des points anormaux), 1/4 chiffres, la médiane, 3/4 chiffres et le minimum (il peut y avoir des points anormaux). La ligne pointillée représente la valeur moyenne. Parce que nous voulons trouver la valeur optimale minimale, plus la valeur de l'indicateur H est petite, meilleure est la qualité de la solution.

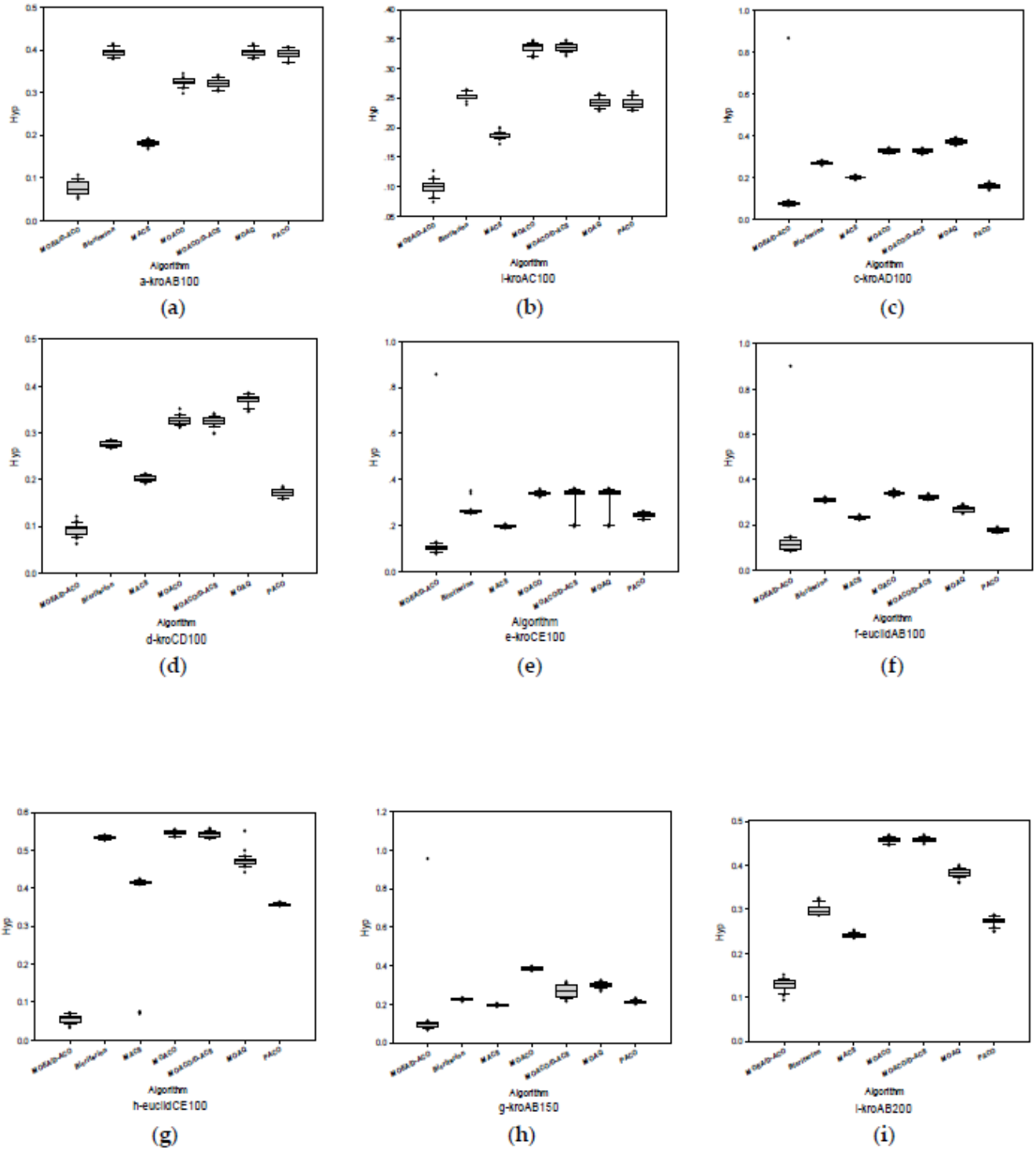

Figure 4.2 - les box plots basées sur l'indicateur H sur certaines instances illustrées comme dans (a) a-kroAB100, (b) i-kroAC100, (c) c-kroAD100, (d) d-kroCD100, (e) e-kroCE100, (f) f-euclidAB100, (g) euclidCE100, (h) h-kroAB150 et (i) i-kroAB200. [21]

Ainsi, à partir des diagrammes en boîte (4.2), nous pouvons voir de manière simple quel algorithme est le meilleur pour résoudre le problème. Si l'algorithme a une valeur carrée plus petite ou une valeur moyenne plus petite du problème.

l'algorithme a de meilleures solutions à ce problème. Si ces cases sont agrégées, on peut considérer que la solution de l'algorithme sur ce problème est relativement stable. À travers box plots des neuf questions ci-dessus, on peut voir que la performance de l'algorithme MOEA/D-ACO dans la résolution de ces neuf problèmes est meilleure que celle des six autres algorithmes. Cependant, en plus du kroAD100, l'algorithme MOEA/D-ACO n'est pas aussi stable que la plupart des autres algorithmes sur les autres cas de test.

L'indicateur d'hyper-volume est strictement monotone et est largement utilisé dans la comparaison de algorithmes d'optimisation multi-objectifs. Cet indicateur détermine la distance entre l'ensemble de solutions avec le front de Pareto en calculant la taille du volume entre chaque ensemble de solutions et le front de Pareto. Si la solution est plus proche du front de Pareto, l'ensemble de solutions est meilleur, sinon il est pire.

Nous exécutons chaque algorithme 30 fois sur chaque problème. Nous transformons ensuite l'ensemble de solutions non dominé en H-indicateur. Enfin, nous comparons et analysons selon ces quatre aspects : maximum (plus c'est petit , le meilleur) , minimum , moyenne , écart-type comme indiqué dans le tableau (4.1) ci-dessous.

Case	Algorithm	Max	Min	Mean	Standard
kroAB100	MOEA/D-ACO	**0.1062**	**0.0505**	**0.0767**	0.0144815
	MACS	0.1924	0.1675	0.1823	**0.0054743**
	MOACO/D-ACS	0.3412	0.3037	0.3236	0.0105447
	MOACO	0.3452	0.2985	0.3254	0.0093617
	PACO	0.4076	0.3695	0.3912	0.011176
kroAC100	MOEA/D-ACO	**0.1264**	**0.0734**	**0.0989**	0.0115925
	MACS	0.1995	0.1723	0.1885	**0.0051235**
	MOACO/D-ACS	0.3475	0.3211	0.3352	0.0063372
	MOACO	0.3472	0.3182	0.3352	0.0077542
	PACO	0.2612	0.2285	0.2413	0.0086281
kroAD100	MOEA/D-ACO	0.8672	**0.0638**	**0.1035**	0.1443142
	MACS	0.2103	0.1881	0.2022	**0.0050040**
	MOACO/D-ACS	0.3404	0.3102	0.3283	0.0066690
	MOACO	0.3412	0.3173	0.3312	0.0074212
	PACO	**0.1802**	0.1401	0.1612	0.0089611
kroCD100	MOEA/D-ACO	**0.1201**	**0.0624**	**0.0925**	0.0122063
	MACS	0.2123	0.1912	0.2021	**0.0057761**
	MOACO/D-ACS	0.3412	0.2971	0.3251	0.0103092
	MOACO	0.2121	0.1912	0.2021	**0.0057761**
	PACO	0.1853	0.1582	0.1713	0.0074041

Table 4.1 - Écart moyen [21]

Case	Algorithm	Max	Min	Mean	Standard
kroCE100	MOEA/D-ACO	0.8363	**0.0857**	**0.1337**	0.1378296
	MACS	**0.2090**	0.1932	0.2017	**0.0040522**
	MOACO/D-ACS	0.3828	0.3482	0.3601	0.0083833
	MOACO	0.3803	0.1986	0.3602	0.0509902
	PACO	0.2784	0.2451	0.2793	0.0072993
euclidAB100	MOEA/D-ACO	0.9044	**0.0853**	**0.1427**	0.1452584
	MACS	0.2469	0.2254	0.2355	**0.0054608**
	MOACO/D-ACS	0.3359	0.3102	0.3236	0.0070820
	MOACO	0.3568	0.3224	0.3371	0.0074314
	PACO	**0.1926**	0.1693	0.1808	0.0066197
euclidCE100	MOEA/D-ACO	**0.0703**	**0.0319**	**0.0548**	0.0101184
	MACS	0.4239	0.0681	0.3925	0.0924695
	MOACO/D-ACS	0.5572	0.5305	0.5432	0.0074305
	MOACO	0.5597	0.5352	0.5454	0.0055957
	PACO	0.3638	0.3543	0.3582	**0.0026483**
kroAB150	MOEA/D-ACO	0.9661	**0.0797**	**0.1395**	0.1555635
	MACS	**0.2010**	0.1998	0.2086	**0.0036208**
	MOACO/D-ACS	0.4149	0.3901	0.4066	0.0309838
	MOACO	0.3102	0.2758	0.2913	0.0060745
	PACO	0.3061	0.2887	0.2976	0.0071204
kroAB200	MOEA/D-ACO	**0.1673**	**0.0901**	**0.1387**	0.0139102
	MACS	0.2506	0.2333	0.2398	**0.0042814**
	MOACO/D-ACS	0.4664	0.4439	0.4566	0.0055929
	MOACO	0.4665	0.4455	0.4569	0.0044351
	PACO	0.2833	0.2459	0.2697	0.0095473

Table 4.1 -cont [21]

Comme nous le notons ci-dessus dans le tableau que l'algorithme MOEA/D-ACO a la valeur H moyenne minimale sur chaque problème. La valeur minimale est la meilleure.mais en ce qui concerne l'écart type, les performances de cet algorithme ne sont pas bonnes dans la plupart des cas. Ceci montre que la stabilité de cet algorithme pour ces problèmes est faible, ce qui est cohérent avec les conclusions des box plots.

En outre,Nous analysons les performances de chaque algorithme à partir de six aspects qui sont le maximum, le minimum, la moyenne, l'écart type, la médiane et la qualité de la solution (les indices H sont aussi petits que possible). En même temps, dans chaque aspect de chaque problème, nous avons répertorié le meilleur algorithme. Les résultats sélectionnés sont affichés dans le tableau (4.2)

	Max	Min	Mean	Standard	Middle	Quality
kroAB100	MOEA/D-ACO	MOEA/D-ACO	MOEA/D-ACO	MACS	MOEA/D-ACO	MOEA/D-ACO
kroAC100	MOEA/D-ACO	MOEA/D-ACO	MOEA/D-ACO	MACS	MOEA/D-ACO	MOEA/D-ACO
kroAD100	PACO	MOEA/D-ACO	MOEA/D-ACO	MACS	MOEA/D-ACO	MOEA/D-ACO
kroCD100	MOEA/D-ACO	MOEA/D-ACO	MOEA/D-ACO	BicriterionAnt	MOEA/D-ACO	MOEA/D-ACO
kroCE100	MACS	MOEA/D-ACO	MOEA/D-ACO	BicriterionAnt	MOEA/D-ACO	MOEA/D-ACO
euclidAB100	PACO	MOEA/D-ACO	MOEA/D-ACO	MACS	MOEA/D-ACO	MOEA/D-ACO
euclidCE100	MOEA/D-ACO	MOEA/D-ACO	MOEA/D-ACO	PACO	MOEA/D-ACO	MOEA/D-ACO
kroAB150	MACS	MOEA/D-ACO	MOEA/D-ACO	MACS	MOEA/D-ACO	MOEA/D-ACO
kroAB200	MOEA/D-ACO	MOEA/D-ACO	MOEA/D-ACO	MACS	MOEA/D-ACO	MOEA/D-ACO

Table 4.2 - Comparaison de plusieurs algorithmes [21]

4.3.2 Analyse de convergence basée sur l'indicateur H

Dans cette partie, plusieurs algorithmes sont sélectionnés pour comparaison. De plus, l'analyse de convergence de ces algorithmes est testée dans des cas tests à différentes échelles. Le but est d'obtenir le nombre d'évaluations de fitness maximales de l'algorithme dans différents cas de test d'échelle. Enfin, nous avons évalué les performances de l'algorithme après convergence. kroAB100, kroAB150, kroAB200 et d'autres cas d'échelle différents sont sélectionnés comme cas de test, MOACO/D-ACS, MOEA/D-ACO, MACS, MOAQ, PACO et ainsi de suite sont choisis comme algorithmes de test.

Comme le montre la figure (4.3), l'axe des x représente le nombre d'évaluations de la condition physique. Avec l'augmentation de la taille du problème, le nombre d'évaluations de la condition physique augmente également. L'axe y représente la valeur de l'indicateur H. Comme on peut le voir sur la figure, avec l'augmentation de l'échelle des cas de test, l'emplacement de convergence de chaque algorithme augmente également.

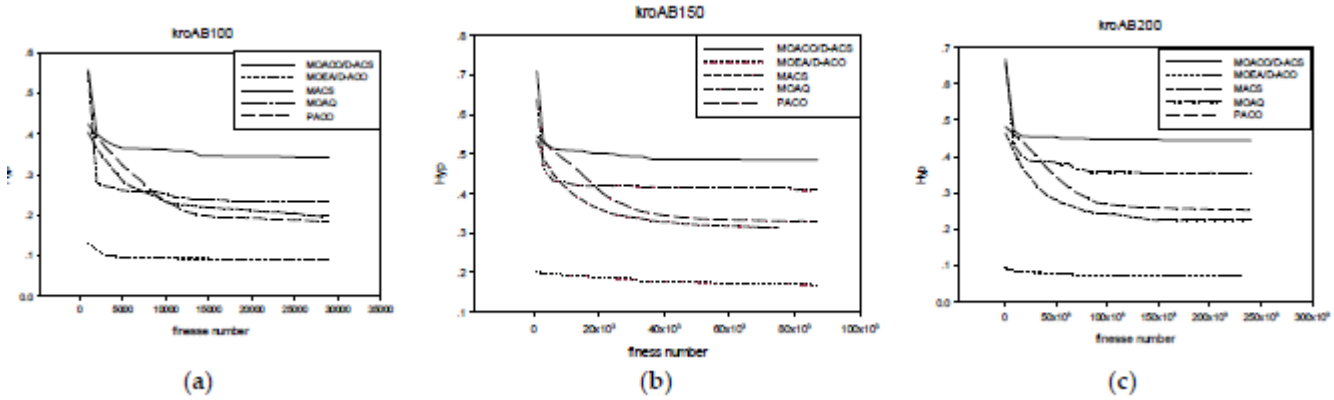

Figure 4.3 -Graphique d'analyse de convergence sur certaines instances de test indiquées comme (a) kroAB100, (b) kroAB150 et (c) kroAB200 [21]

4.3.3 Front de Pareto approximatif

Dans cette partie, nous sélectionnons trois exemples représentatifs kroAB100, kroAB150 et kroAB200. En exécutant ces algorithmes suivants 30 fois indépendamment, nous obtenons les

fronts de Pareto approximatifs pour les MOEA/D-ACO, MACS, MOACO, MOACO/D-ACS et PACO. À partir de la figure (4.4)

nous pouvons voir que l'ensemble de solutions de l'algorithme MOEA/D-ACO sur les deux premiers cas de test est meilleur que les autres algorithmes. Son ensemble de solutions est plus approché du PF. Cependant, MOEA/D-ACO ne se rapproche que d'une partie du front de Pareto, et l'ensemble de solutions d'autres algorithmes est plus complet. Dans le troisième graphique, les solutions des deux algorithmes basés sur la décomposition, MOEA/D-ACO et MOACO/D-ACS, sont significativement meilleures que celles des autres algorithmes. Cependant, l'ensemble de solutions de MOACO/D-ACS ne se rapproche que de la partie médiane du front de Pareto. algorithmes pour résoudre le même cas de test. Ceci est cohérent avec la conclusion que nous avons obtenue en analysant les box plots.

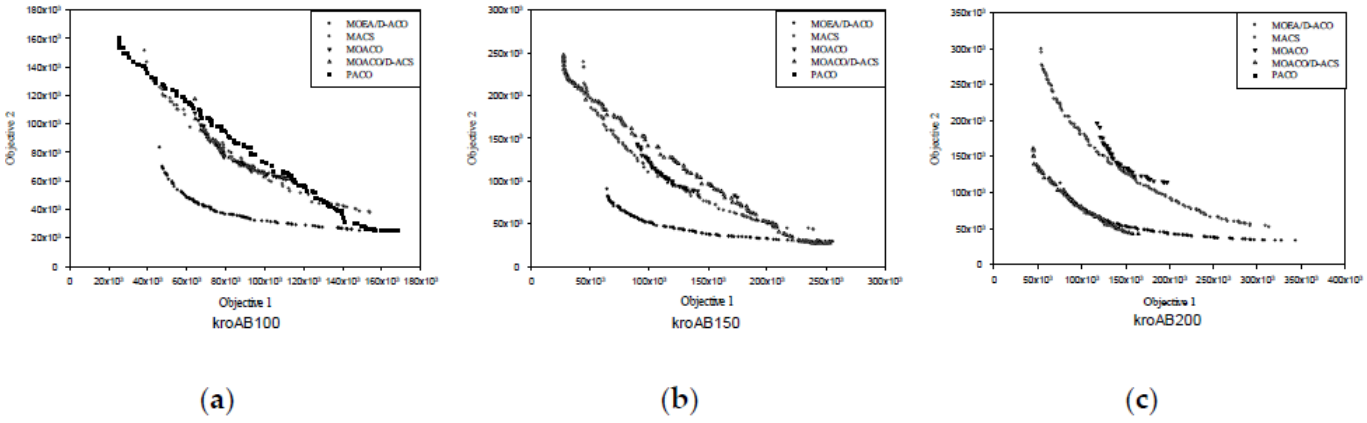

Figure 4.4 -Fronts de Pareto approximatifs sur certaines instances indiquées comme (a) pour kroAB100, (b) pour kroAB150 et (c) pour kroAB200. quad [21]

On peut voir que la solution obtenue par l'algorithme MOACO basé sur le tri non dominé peut obtenir plus de solutions et peut se rapprocher d'une partie spécifique du front de Pareto. La distribution de ces solutions n'est pas uniforme. En revanche, les solutions obtenues par le MOACO basées sur la décomposition peuvent se rapprocher du front de Pareto de manière plus uniforme.

Ce type d'algorithme n'enregistre que la meilleure solution dans chaque sous-direction. Ainsi, le nombre de solutions est plus faible. Il ressort des travaux récents sur le sujet que la recherche et l'application de MOACO basées sur la décomposition ont attiré beaucoup d'attention et sont devenues un sujet brûlant dans le domaine de l'optimisation multi-objectifs. Cependant, dans une situation spécifique, il est essentiel de considérer quel type d'algorithme de colonie de fourmis multi-objectifs choisir pour résoudre un problème et nous devons analyser les caractéristiques de l'espace de recherche qui lui est lié. Si la distribution du Front de Pareto est décentralisée, l'avantage d'utiliser une approche basée sur la décomposition est plus évident.

4.4 conclusion

Dans le quatrième chapitre, nous avons mené une étude comparative des algorithmes de colonies de fourmis (MACS, PACO, MOACO, MOACO/D-ACS, MOEA/D-ACO) pour l'optimisation multi-objectifs.

conclusion générale

Nous avons défini dans le première chapitre les problèmes d'optimisation combinatoire et quelques exemples de ces problèmes .

Dans le deuxième chapitre nous avons défini les problèmes d'optimisation multi-objectif et les éléments de base et Quelques méthodes de résolution.

Dans le troisième chapitre Nous avons présenté la classification des Algorithmes MOACO et plusieurs algorithmes MOACO proposés dans la littérature. Ces algorithmes proposent différentes stratégies pour la résolution des problèmes multi-objectifs.

Dans le quatrième chapitre, nous avons mené une étude comparative des algorithmes de colonies de fourmis (MACS, PACO, MOACO, MOACO/D-ACS, MOEA/D-ACO) pour l'optimisation multi-objectifs.

Avec le temps,la théorie et la recherche appliquée sur l'optimisation multi-objectifs des colonies de fourmis se développent continuellement.Il a été démontré que l'algorithme d'optimisation multi-objectifs des colonies de fourmis présente un avantage certain pour le problème d'optimisation de l'espace discret ,Les chercheurs ont également réalisé quelques réalisations lorsque essayant de l'appliquer pour résoudre le problème d'optimisation continue multi-objectifs. Bien que l'algorithme de fourmis multi-objectifs fonctionne bien pour résoudre le problème d'optimisation combinatoire discrète, il reste encore de nombreux problèmes à résoudre aujourd'hui, tels que comment empêcher les fourmis d'entrer dans l'optimal local, la meilleure approche pour éviter que l'algorithme ne converge prématurément

Les problèmes d'optimisation multi-objectifs sont déjà une voie de recherche future prometteuse.

Référence

[1] HAMDAOUI FAWZIA, contribution a la résolution du problème du sac a dos multidimensionnel a choix multiple, université des sciences et de la technologie d'Oran Mohamed Boudiaf, faculté des mathématique et informatique, spécialité d'informatique,2016 .

[2] KOUCHI Mohamed et Mili Kamel Le problem de voyageurs de commerce Bi-Objectifs,2012 – 2013, https ://wikimemoires.net/, 22 :26 18/04/2022.

[3] Hao, J. K., Galinier, P., & Habib, M. (1999). Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. Revue d'intelligence artificielle, 13(2), 283-324.

[4] Inès Alaya. Optimisation multi-objectif par colonies de fourmis : cas des problèmes de sac à dos.Ordinateur et société [cs.CY]. Université Claude Bernard-Lyon I ; Université de la Manouba (Tunisie),2009. Français. NNT : 2009LYO10060.

[5] Clarisse Dhaenens. Optimisation Combinatoire Multi-Objectif : Apport des méthodes coopératives et contribution à l'extraction de connaissances. Modélisation et simulation. Université des Sciences et Technologie de Lille-Lille I, 2005.

[6] Palpant M, Recherche exacte et approchée en optimisation combinatoire : schémas d'intégration et applications. Thèse de Doctorat, Université d'Avignon, 2005.

[7] Abdesslem LAYEB , Utilisation des Approches d'optimisation Combinatoire pour la Vérification des Applications Temps Réel , l'université Mentouri de Constantine, faculté des Sciences de l'ingénieur , Département d'informatique ,2010.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm : NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2002), no. 2, 182–197.

[9] ZOUBIR , RAMDANI. Cours d'optimisation multiobjectifs.Université Mohamed El-Bachir El-Ibrahimi de Bordj Bouararidj. Faculté des mathématiques et informatique.2020/2021.

[10] VINCENT, Thomas. Caractérisation des solutions efficaces et algorithmes d'énumération exacts pour l'optimisation multiobjectif en variables mixtes binaires.Thèse de doctorat. Nantes.2013.

[11] SAHA, Adel. Résolution des problèmes multi objectifs à base de colonies de fourmi, Thèse de Magister. Université de Batna 2,2010.

[12] CHERIET, Abdelhakim. Métaheuristique hybride pour l'optimisation multi-objectif, Thèse de doctorat, Université Mohamed Khider-Biskra,2016.

[13] BERRAH Garmia L'optimisation Quadratique Indéfinie ,2019/2020 ,https :/www.academia.edu/, 00 :50, 25/04/2022.

[14] THLEDJANE NOOR EL HOUDA ET SALIK NESRINE , Aspects théorique de l'optimisation multi-objectif , Université Mohamed El Bachir El Ibrahimi de Bordj Bou Arréridj Faculté de Mathématiques et de l'informatique Département de Mathématiques, 2016.

[15] TAHRAOUI Mohammed Amine,Colonie de fourmis pour le problème du transport mutlimodal , Université Mira Abderrahmene de Béjaia , faculté des sciences et science de l'ingénieur département d'informatique école doctorale d'informatique , Janvier 2009.

[16] Mochila fraccionable, Algoritmos voraces en JavaScript, 20/03/2020 https ://www.wextensible.com/temas/voraces, 25/05/2022 ,16 :29.

[17] BAYOU Abdelwahab ,BENSEFIA Amir, un algorithme hybride(ACO-2Opt) pour la résolution du problème de voyageur de commerce, université Mohamed El Bachir El Ibrahimi - Bordj Bou Arreridj- Faculté des Mathématiques et d'informatique 2021.

[18] Shahin Rostami ,ferrante Neri ,kiril Gyaurski ,On Algorithmic Descriptions and Software Implementations For Multi objective Optimisation : A Comparative Study, 04/08/2020, https ://www.researchgate.net ,25/05/2022,16 :46 .

[19] Kalyanmoy Deb, Kaisa Miettinen, and Shamik Chaudhuri,Toward an Estimation of Nadir Objective VectorUsing a Hybrid of Evolutionary andLocal Search Approaches.2010.https ://www.researchgate.net

[20] COSTANZO Andrea , LUONG Thé Van, MARILL Guillaume , Optimisation par colonies de fourmis , 19 mai 2006.

[21] Jiaxu Ning , Changsheng Zhang ,Peng Sun,and Yunfei Feng ,Comparative Study of Ant Colony Algorithms for Multi-Objective Optimization

[22] Benfriah Rafik ,Kouahi Mohammed Amine ,optimisation de la Chaine Logistique Pharmaceutique a base de colonie de fourmis artificielles cas de la societe cojipharm, 2013/2014 .