Université Mohamed El-Bachir El-Ibrahimi Bordj Bou Arreridj
Faculté des mathématiques et d'informatique
Département de la recherche opérationnelle

MEMOIRE

En vue de l'obtention du diplôme de

MASTER

FILIERE DES MATHEMATIQUE APPLIQUEE

Spécialité : Méthodes Et Outils pour La Recherche Opérationnelle

ALGORITHME HYBRIDE (ACO-APE) POUR LA RÉSOLUTION DU PROBLÈME DE VOYAGEUR DE COMMERCE

Réalisé par :
Kenza Bechtoula.
Hadjer Bouguerra.

Sous la direction de :
Dr. Adel Saha.

Soutenu le: 27/09/2021

Devant le jury composé de :

- *Dr. MAACHE SALAH.*
- *Dr. FARHAT FILALI.*

Année universitaire 2020-2021

Remerciements

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.

Nous remercions Dr : Saha Adel, notre encadreur, proposé ce sujet et son conseil dès le début.

Un grand merci au Dr : Mokhnache Abdelaziz pour son soutien et son accompagnement durant ce travail.

Nous remercions également tous enseignants durant notre parcours académique.

Nous remercions aussi tous les étudiants RO dans l'université de BBA, ceux de la promotion 2019-2021.

Ce travail n'aurait pu être réalisé sans le soutien et les encouragements de notre famille que nous remercie tout particulièrement.

Enfin, nous tenons à remercier tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Dédicaces

Grace a tout puissant, j'ai pu terminer ce modeste travail que je dédie :

A mes parents, ma mère pour ses encouragements et ses prières tout au long de mes études, mon père pour tous ce qu'il avait fait pour avoir ce résultat.

Que dieu les protègent.

A mon adorable sœur "Sahra", qui a toujours été là pour moi tout au long de mes études.

A ma chére cousin Nour, qui grâce à lui j'ai accompli mon travail

A tous mes amis, mes collègues et tous ceux qui me connaissent.

Kenza

Dédicaces

Je dédie ce travail :

A ma chére mère

A mon père qui nous a quitté le dans le mérite les sacrifices et les qualités humaines m'ont permis de vivre ce jour

A mes frères et ma sœur

A mon mari pour leur encouragement leurs soutien moral

Hadjer

TABLE DES MATIERES

Liste des Figures

Chapitre I

Chapitre II

Chapitre III

Liste des Tableaux

Liste des Algorithms

Liste des Abréviation

ACO : Ant Colony Optimization.

PSO : Particle swarm optimization.

APE : adjacent pairwaise axchange.

PVC : problème de voyageur de commerce.

TSP : travelling salesman problem.

POC : problème d'optimisation combinatoire.

AS : ant system.

ACS : Ant colony system.

OFC : optimisation par colonies de fourmies.

Introduction Générale

Introduction générale

Dans la vie courante, nous sommes souvent en contact avec des problèmes et des difficultés qui nécessitent des études, des décisions et des solutions. Parmi les domaines qui mettent en évidence ces problèmes figurent : les mathématiques, la physique, l'informatique et recherche opérationnelle.

Le domaine le plus important est la "recherche opérationnelle", cette science appartient aux mathématiques appliquées, elle repose sur le regroupement de toutes les méthodes et techniques rationnelles afin d'étudier et résoudre les problèmes, elle visant à trouver la meilleure décision et les meilleurs résultats possibles pour permettre aux décideurs de choisir des solutions plus efficaces. C'est ce que l'on appelle " optimisation".

Les problèmes d'optimisation que les on rencontre quotidiennement, ils sont modélisés par des spécialistes et mis sur la table d'étude pour chercher des solutions aux plusieurs domaines tels que : l'ingénierie, l'industrie, la gestion du marketing, l'armée, les transports...etc.

La résolution d'un problème d'optimisation consiste à trouver la ou les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur.

Suivant le problème d'optimisation, on distingue des méthodes dites exactes qui garantissent la résolution d'un problème, mais au coût de temps de calculs prohibitifs, en citant par exemple la méthode de simplexe.

Néanmoins, lorsqu'on veut résoudre un problème complexe, ces méthodes prennent un temps de calcul qui croit exponentiellent avec la taille des instances du problème, ceci conduit souvent à une explosion combinatoire.

Parmi les problèmes de l'optimisation combinatoire, nous trouvons le problème de voyageur de commerce, bien qu'il est facile à comprendre mais difficile à résoudre. Pour éviter ce type de problème, on fait appel aux méthodes approchées qui ne cherchent pas forcément l'optimum absolu mais donnent une solution très proche, montrant l'existence d'une solution sensiblement meilleure et largement acceptée.

En effet, les chercheurs du domaine ont trouvé des méthodes d'une inspiration biologique (exemple : colonie de fourmis, algorithme génétique).

Ces méthodes ont des limites c'est pour ça les chercheurs ont combiné ces derniers avec d'autres pour avoir des méthodes hybride comme le modèle proposé par Mahi & Al -2005- (ACO+PSO+3-OPT) pour résoudre le problème de voyageur de commerce et le modèle que nous allons présenter (ACO+PSO+APE) et qui sera étudié dans les chapitres suivants.

Notre étude dans ce mémoire est résumée suivant la question ci-après :

Est-ce que le modèle ACO+PSO+APE est meilleur que le modèle ACO+PSO+3-OPT ?

Afin de voir les performances de ce modèle, six instances du problème PVC de la base TSPLIB sont étudiés et à la fin nous avons comparé nos résultats avec ceux trouver par les modifications apportées sur model de Mahi & Al.

Ce mémoire est organisé de la façon suivante :

- Le premier chapitre est consacré à la présentation des problèmes d'optimisation combinatoire et la classification des méthodes de résolution existantes ;
- Dans le deuxième chapitre, nous focalisons plus particulièrement sur l'optimisation par Colonies de Fourmies avec l'explication de deux modelés hybrides ;
- Le troisième et le dernier chapitre, est une partie d'application et d'implémentation d'algorithmes, il abord aussi une analyse des résultats avec quelques changements de paramètres.

Chapitre I

OPTIMISATION COMBINATOIRE

Introduction

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 parte par la variété des applications pratiques est pouvant être formulée sous la forme d'un problème d'optimisation combinatoire.

Les problèmes d'optimisation combinatoire sont souvent faciles à définir généralement difficiles à résoudre, ils appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour des solutions algorithmiques efficace valable pour toutes les données.

L'optimisation combinatoire minimise (ou maximise) une fonction souvent appelée fonction de coût. Constituer d'une ou plusieurs variables et une ou plusieurs contraintes.
Le sujet de l'optimisation combinatoire ; traiter dans un domaine discret ; cherche toujours la possibilité optimale parmi toutes les choix. Ceci parait facile mais devient infaisable dès que la taille du problème est suffisamment grande, la taille pour laquelle la recherche d'un optimum devient faisable est petite.
En général, la difficulté d'un problème grandit très vite avec le nombre des variables, il n'est pas alors faisable d'examiner toutes les possibilités.

Dans ce chapitre nous présentons le problème combinatoire et les problèmes d'optimisation combinatoire.

I.1. L'optimisation

Un problème d'optimisation se définit comme la recherche du minimum ou du maximum (l'optimum) d'une fonction donnée. On peut aussi trouver des problèmes d'optimisation pour lesquels les variables de la fonction à optimiser sont contraintes d'évoluer dans une certaine partie de l'espace de recherche. [1]

I.2. Formulation mathématique des problèmes d'optimisation

Les problèmes d'optimisation combinatoire peuvent être formulés comme suit

I.2.1 Problèmes mono-objectifs [2]

Un problème d'optimisation est généralement formulé comme un problème de minimisation ou de maximisation, et s'écrit sous la forme suivante :

$$\begin{cases} \min f(x), \quad \text{telque} \\ g_i(x) \leq 0, \quad i = 1, \dots, m \\ h_j(x) = 0, j = 1, \dots, p \\ x \in S \subset R^n \end{cases} \qquad \text{I.1}$$

Où : *f* est la fonction à minimiser, appelée fonction coût ou fonction objectif ;
x représente le vecteur des variables d'optimisation ;
g_i sont les contraintes d'inégalité et h_j les contraintes d'égalité ;
S est l'espace des variables (appelé aussi espace de recherche).

A noté que l'espace des variables *S* indique quel type de variables, à savoir : réelles, entières, mixtes (réelles et entières dans un même problème), discrètes, bornées, ...etc.

Un point x_A est appelé un point admissible si $x_A \in S$ et si les contraintes d'optimisation sont satisfaites :

$$g_i(x_A) \leq 0,\ i = 1,\dots, m \quad \text{et} \quad h_j(x_A) = 0,\ j = 1,\dots, p. \qquad \text{I.2}$$

I.2.2 Problèmes multi-objectifs

Un problème d'optimisation combinatoire peut avoir plusieurs fonctions objectives, il s'agit alors d'un problème d'optimisation multicritère ou multi-objectif.

D'un point de vue mathématique, il est représenté, dans le cas où le vecteur F regroupe k fonctions objectif, de la façon suivante :

$$\begin{cases} \min F(x), \quad \text{telque} \\ g_i(x) \leq 0, \quad i = 1, \dots, m \\ h_j(x) = 0, \quad j = 1, \dots, p \\ x \in S \subset R^n \end{cases} \qquad \text{I.3}$$

De ce fait, résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points suivants :

- La définition de l'ensemble des solutions réalisables ;
- L'expression de l'objectif à optimiser ;
- Le choix de la méthode d'optimisation (exacte ou approchée) à utiliser.

A noté que les deux premiers points relèvent de la modélisation du problème et le troisième point de sa résolution. [3]

I.3. Optimisation combinatoire

L'étude des problèmes d'optimisation combinatoire représente un gain non négligeable sur la qualité de leur résolution.

Un problème d'optimisation combinatoire, consiste dans un espace discret (*i.e.* énumérable) de solutions réalisables et cherche à trouver la meilleure solution (ou un ensemble des meilleures solutions).

La notion de meilleur est donnée par un critère peut être défini par un ensemble discret des solutions réalisables du problème appelé Ω, on de qualité via une fonction objective.

Formellement, un problème d'optimisation combinatoire parle alors d'espace de recherche, et $f : \Omega \rightarrow \boldsymbol{R}$ la fonction objectif associée au critère de qualité.

Le but est de trouver $s^* \in \Omega$ tel que :

$$s^* = \arg\max_{s \in \Omega} \{f(s)\} \qquad \text{I.4}$$

On parle d'un problème de maximisation quand la qualité est donnée par une fonction objective à maximiser et d'un problème de minimisation quand la qualité est donnée par une fonction objective à minimiser. [4]

I.4. Terminologie

I.4.1. Voisinage [5]

Soit x une solution, on dit que x^* est une solution voisine de x, si on peut obtenir x^* en modifiant légèrement x.

Le voisinage V (x) de x est l'ensemble des solutions voisines de x.

I.4.2. Optimum local [5]

Une solution s $\in S$ est un optimum local si et seulement s'il n'existe pas de solution $s_0 \in v(s)$, dont l'évaluation de meilleure qualité que s, soit :

$$\forall s_0 \in v(s) \begin{cases} f(s) \leq f(s_0) & dans\ le\ cas\ d'un\ probléme\ de\ minimisation \\ f(s) \geq f(s_0) & dans\ le\ cas\ d'un\ probléme\ de\ maximisation \end{cases} \qquad \text{I.5}$$

Avec V(s) l'ensemble de solution voisines de s.

I.4.3. Optimum global [5]

Une solution est un optimum global à un problème d'optimisation s'il n'existe pas d'autre solution de meilleure qualité. La solution s* ∈ S est un optimum global si:

$$\forall s \in S \begin{cases} f(s^*) \leq f(S) & dans\ le\ cas\ d'un\ probléme\ de\ minimisation \\ f(s^*) \geq f(S) & dans\ le\ cas\ d'un\ probléme\ de\ maximisation \end{cases} \qquad \text{I.6}$$

I.4.4. Fonction objective

C'est le nom donné à la fonction *f (*on l'appelle encore fonction de coût ou critère d'optimisation). La fonction objective est la fonction que l'algorithme d'optimisation va devoir "optimiser" c'est-à-dire à trouver un optimum. [6]

I.4.5. Variables de décision

Elles sont regroupées dans le vecteur $\vec{x}$. Faisant varier ce vecteur pour trouver l'optimum de la fonction *f.*[5]

I.5. Classification des problèmes d'optimisation

On peut classer les différents problèmes d'optimisation que l'on rencontre dans la vie courante en fonction de leurs caractéristiques :

1 - Nombre de variables de décision, à savoir

_ Une →monovariable ;

_ Plusieurs→multivariable.

2 - Type de la variable de décision, à savoir

_ Nombre réel continu→continu ;

_ Nombre entier→entier ou discret ;

_ Permutation sur un ensemble fini de nombres→combinatoire.

3 - Type de la fonction objectif, à savoir

_ Fonction linéaire des variables de décision→ linéaire ;

_ Fonction quadratique des variables de décision→ quadratique ;

_ Fonction non linéaire des variables de décision→non linéaire.

4 - Formulation du problème, à savoir

_ Avec des contraintes→contraint ;

_ Sans contraintes→ non contraint.

5 - Nombre de fonctions objectifs, à savoir

_ Une→ mono-objectif ;

_ Plusieurs → multi-objectifs.[6]

I.6. La théorie de complexité

D'une manière générale, pour résoudre un problème, on est appelé à trouver l'algorithme le plus efficace. Cette notion d'efficacité induit normalement toutes les ressources de calcul nécessaire pour exécuter un algorithme. Or, le temps d'exécution est généralement le facteur dominant pour déterminer si un algorithme est assez efficace pour être utilisé dans la pratique, pour cela on se concentre principalement sur cette ressource.

On appelle complexité en temps d'un algorithme le nombre d'instructions élémentaires mises en œuvre dans cet algorithme afin de résoudre un problème donné. Une instruction élémentaire sera une affectation, une comparaison, une opération algébrique…etc.

La complexité d'un problème est la complexité du meilleur algorithme qui permet de le résoudre. Si cet algorithme est polynomial, le problème est dit facile, autrement le problème est difficile. [7]

I.6.1. La classe P (polynomial time)

Soit π un problème quelconque. Si pour toute instance I de π, une solution de I peut être déterminée algorithmiquement en un temps polynomial alors π est appelé problème polynomial et l'algorithme qui le résout algorithme polynomial (ou un algorithme efficace).
Les problèmes polynomiaux constituent la classe P. [8]

I.6.2. La classe NP (Not Deterministe Plynomial Time)

Soit un problème quelconque π et I ses instances pour lesquelles la réponse est oui.
Le problème π appartient à la classe NP s'il existe un algorithme polynomial qui permet de vérifier que cette réponse est bien oui pour ces instances I.
Notons que NP ne veut en aucun cas dire non Polynomial mais Not Deterministe Polynomial. [8]

Remarque :
Un problème de décision est un problème où la résolution se limite à la réponse par « oui » ou «non» à la question de savoir s'il existe une solution au problème.

I.6.3. Problème NP-Complet

Un problème de décision π est NP-complet s'il satisfait les deux conditions suivantes :

- $\pi \in$NP ;
- Tout problème NP se réduit à π en temps polynomial.

Les problèmes NP-Complet sont liés par une relation d'équivalence dans le sens où s'il existe un algorithme polynomial pour un des problèmes de cette classe alors tous les problèmes NP-Complet pourront être résolus en un temps polynomial.
Cette classe englobe les problèmes les plus étudiés de l'optimisation combinatoire. [8]

I.6.4. Problème NP-difficile

Un problème de NP est dit NP-difficile, si et seulement s'il existe un problème NP-Complet qu'est réductible à lui en temps polynomial.

De cette définition, on conclut que pour montrer qu'un problème d'optimisation est NP-difficile, il suffit de montrer que le problème de décision associé à lui est NP-complet.
Ceci explique pourquoi, lors de l'étude d'un nouveau problème, on commence par chercher à classer ce problème.
Si l'on parvient à montrer qu'il est polynomial, le problème sera résolu. [8]

I.7. Exemple de problèmes d'optimisation combinatoire

I.7.1. Le problème du sac à dos [9]

Considérons n objets, notés i = 1,…, n . Apportant chacun une utilité u et possédant un poids p_i. On veut ranger ces objets dans un « sac » de capacité c.

Le problème de sac-à-dos (knapsack) consiste à choisir les objets à prendre parmi les n objets de manière à avoir une utilité maximale et respecter la contrainte de la capacité c, à ne pas dépasser.
La formulation PLNE du problème de sac-à-dos est très simple. On utilise pour chaque objet $i \in 1,\ldots, n$, une variable binaire x_i correspond à 1 si l'objet i est pris 0 sinon.

Le problème du sac-à-dos est modélisé comme suit :

$$\begin{cases} Max \sum_{i=1}^{n} u_i \, x_i \\ p_i x_i \leq c \\ x_i \in \{0,1\} \; \forall i = \{1, \dots, n\} \end{cases} \qquad \text{I.8}$$

I.7.2. Problème d'ordonnancement

Le problème d'ordonnancement consiste à ordonnancer un ensemble de tâches indépendantes sur *m* machines non reliées, le critère à optimiser étant la durée totale t, de l'ordonnancement.

Dans le contexte non relié, chaque tâche *i* a une durée d'exécution dépendante de la machine *j* sur laquelle elle est effectuée. Ainsi la durée de la tâche *i* sur la machine *j* est p_{ij}. Ce problème est modélisé par le programme linéaire suivant:

$$(PL) \begin{cases} Min \; t \quad (1) \\ sc \sum_{j=1}^{m} x_{ij} = 1 \; \forall i \in \{1, \dots, n\} \quad (2) \\ \sum_{i=1}^{n} p_{ij} x_{ij} \leq t \; \forall j \in \{1, \dots, m\} \quad (3) \\ x_{ij} \in \{0,1\} \; \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\} \quad (4) \end{cases} \qquad \text{I.9}$$

La contrainte (2) assure que chacune des *n* tâches est affectée à l'une des *m* machines.
La contrainte (3) assure que pour chacune des machines *j*, la somme des durées d'exécution des tâches affectées à *j* est inférieure à la durée de l'ordonnancement qui est le critère à minimiser dans (1).
Pour la contrainte (4) : x_{ij} = 1 si et seulement si la tâche *i* est exécutée par la machine *j*. [10]

I.7.3. Problème d'emploi du temps

Dans les établissements scolaires, chaque année l'administration est face à un problème, elle essaye toujours de planifier un certain emploi de temps tout en respectant certaines contraintes suivantes, à savoir :

- Chaque enseignant doit être affecté à une seule salle pour une durée limitée (exemple : une heure) et le même enseignant ne peut être affecté à deux salles en même temps ;
- Chaque salle est occupée par un seul enseignant.

Le problème à résoudre consiste à concilier toutes ces contraintes pour proposer un emploi bien cerner à chaque enseignant. [11]

I.7.4. Le problème du voyageur de commerce (PVC)

Un voyageur de commerce ayant n villes à visiter souhaite établir une tournée qui lui permette de passer une et une seule fois dans chaque ville pour finalement revenir à son point de départ, ceci en minimisant la longueur du chemin parcouru.

Etant donné un graphe $G=(X,U)$ dans lequel :
-X : l'ensemble des sommets représente les villes à visiter, ainsi que la ville de départ de la tournée ;
-U : l'ensemble des arcs de G, représentent les parcours possibles entre les villes.

A tout arc $(i, j) \in U$, on associe la distance de parcours $d_{i,j}$ de la ville i à la ville j, la longueur d'un chemin dans G est la somme de distances associées aux arcs de ce chemin.

Le PVC se ramène alors à la recherche d'un circuit hamiltonien de longueur minimale dans G.
Au finale, Le PVC peut être modélisé comme suit :
En associant à chaque couple (i, j) de villes à visiter $(i=1,..,n ; j=1,...,n$ et $i \neq j$) une distance $\delta_{i,j}$ égale à $d_{i,j}$ s'il existe un moyen d'aller directement de i à j (c'est à dire , $(i, j) \in U$, fixé à ∞ sinon et une variable de succession, $x_{i,j}$, binaire, qui prend la valeur 1 si la ville j est visitée immédiatement après la ville i dans la tournée et qui prend la valeur 0 sinon.

Le PVC est alors modélisé par :

$$\begin{cases} Minimiser\ z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} & (1) \\ Sous\ les\ contraintes: & \\ \sum_{j=1}^{n} x_{ij} \leq 1\,, \quad i = \{1,\dots,m\} & (2) \\ \sum_{i=1}^{m} x_{ij} = 1\,, \quad j = \{1,\dots,n\} & (3) \\ x_{ij} \in \{0,1\} \quad \forall\, i \in \{1,\dots,n\}, \forall j \in \{1,\dots,m\} & (4) \end{cases} \qquad \text{I. 10}$$

La contrainte (1) - (2) traduisent le fait que chaque ville doit être visitée exactement une fois ;
La contrainte (3) interdit les solutions composées des sous-tours disjointes, elle est généralement appelée contrainte d'élimination des sous-tours. [12]

I.8. Résolution d'un problème d'optimisation combinatoire

Les méthodes de résolutions des problèmes d'optimisation sont généralement classifiées en deux grandes catégories : celles d'optimisation continue et celles d'optimisation combinatoire.

Les méthodes d'optimisation continue sont classifiées en linéaires et non-linéaires, celles de l'optimisation combinatoire sont classifiées en exactes et approchées (ci-joint La figure I.1).

A noté que plusieurs approches ont été utilisées pour résoudre le PVC, ils sont généralement classés en deux grandes catégories :
-les méthodes exactes (complètes) qui garantissent la complétude de la résolution ;
-les méthodes approchées (incomplètes) qui perdent la complétude pour gagner en efficacité. [13]

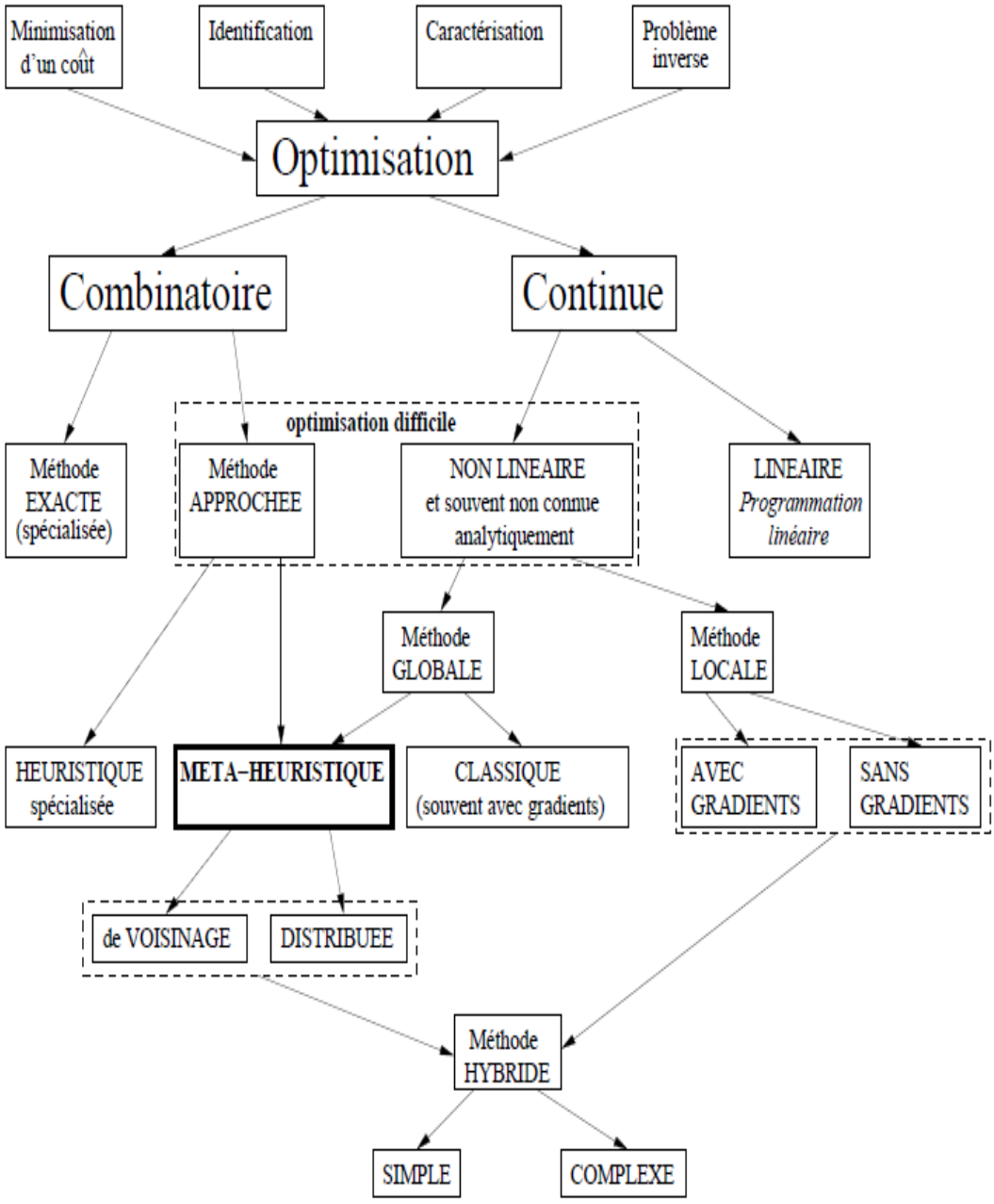

Figure. I.1 : Classification générale des méthodes d'optimisation monobjectif. [6]

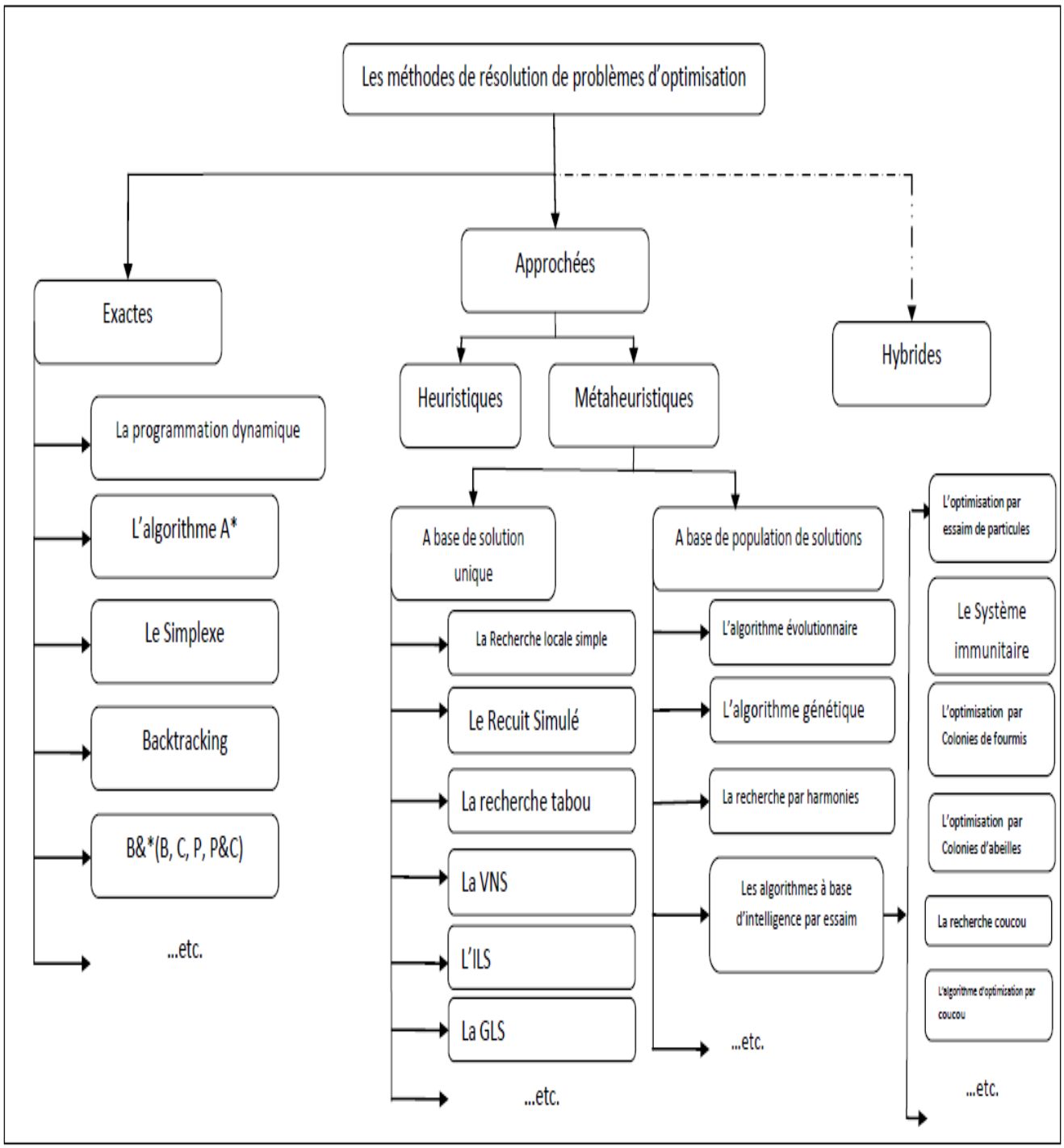

Figure.I.2 : Classification de méthode de résolution de problème d'optimisation. [14]

I.8.1. Les méthodes exactes

Les méthodes (ou algorithmes) exactes garantissent de trouver la solution optimale et de prouver son optimalité pour toutes les instances d'un *POC* permettent aussi de fournir des informations sur les bornes inférieures/supérieures de la solution optimale d'un *POC* dans le cas où l'algorithme est arrêté avant la fin. Par contre, le temps nécessaire pour trouver la solution optimale d'un *POC* augmente en général fortement avec la taille du problème.

D'une façon pratique, seuls les problèmes de petites et moyennes tailles peuvent être résolus de façon optimale par des algorithmes exacts. De plus, pour certains problèmes, la consommation de mémoire de ces algorithmes peut être très grande et peut parfois entraîner l'arrêt prématuré de l'application informatique.

Parmi les algorithmes exacts, on retrouve l'algorithme du simplexe, le *branch-and-bound (B&B),* le *branch-and-cut (B&C),* le *branch-and-price (B&P),* le *branch-and-cut-and-price (B&C&P),* la programmation dynamique, les méthodes basées sur la relaxation lagrangienne et la programmation par contraintes. [14]

I.8.2. Les méthodes approchées (heuristiques)

Les méthodes approchées représentent une alternative pour résoudre les problèmes d'optimisation de grande taille lorsque les méthodes exactes échouent. Elles permettent aussi d'obtenir des solutions de bonne qualité en temps de calcul réduit, mais sans garantie d'optimalité.

Les méthodes approchées sont fondées principalement sur des heuristiques spécifiques à un type de problème, elles sont englobent deux classes :

- Méthodes constructives qui génèrent des solutions à partir d'une solution initiale en essayant d'ajouter petit à petit des éléments jusqu'à ce qu'a l'obtention d'une solution complète ;
- Méthodes d'amélioration qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière répétitive essaient d'améliorer cette solution en explorant son voisinage (MST, 3-OPT, APE). [15]

I.8.3. Méta-heuristiques

Face aux difficultés rencontrées par les heuristiques pour avoir une solution réalisable de bonne qualité pour des problèmes d'optimisation difficiles, issus des domaines de la recherche opérationnelle dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand à la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage, les méta-heuristiques ont fait leur apparition.

Le rapport entre le temps d'exécution et la qualité de la solution trouvée d'une méta-heuristique reste alors, dans la majorité des cas, très intéressant par rapport aux différents types d'approches de résolution.

Une méta-heuristique peut être adaptée pour différents types de problèmes, tandis qu'une heuristique est utilisée à un problème donné (i.e. : une méta-heuristique combine plusieurs approches heuristiques), et parmi les méta-heuristique, on trouve l'algorithme de colonie de fourmis qui a joué un rôle important et efficace dans la résolution de nombreux problèmes d'optimisation. [14]

I.9. Méthodes hybrides

Une méthode hybride est une méthode de recherche constituée d'au moins de deux méthodes de recherche distinctes.
Elle consiste à exploiter les avantages respectifs de plusieurs méthodes en combinant leurs algorithmes suivant une approche synergétique.

Une méthode hybride peut être mauvaise ou bonne selon le choix et les rôles de ses composants. Pour définir une méthode hybride efficace, il faut savoir caractériser les avantages et les limites de chaque méthode.

A titre d'exemple des meilleurs résultats sont obtenus en hybridant une méthode évolutionnaire avec une méthode de recherche locale pour la résolution d'un problème de coloration des graphes.

Actuellement, les approches hybrides gagnent en popularité car ce type d'algorithme produit généralement les meilleurs résultats pour plusieurs problèmes d'optimisation combinatoire. [16]

Conclusion

Dans ce chapitre nous avons présenté les problèmes d'optimisation combinatoire.

Dans un premier temps nous avons défini le problème d'optimisation combinatoire puis nous avons exposé la formulation mathématique des problème (mono et multi objectifs) et la complexité d'un problème d'optimisation, ensuit nous avons détaillé les différentes classes de complexité, et nous avons présente quelques exemples des problèmes. Et à la fin nous avons présenté plusieurs méthodes de résolution heuristiques.

Pour permettant de trouver de bonnes solution approchée, il est nécessaire de faire appel à des heuristiques.

Chapitre II

COLONIES DE FOURMIS

Introduction

Une méthode hybride est une méthode de recherche constituée d'au moins de deux méthodes de recherche distinctes. Elle consiste à exploiter les avantages respectifs de plusieurs méthodes en combinant leurs algorithmes suivant une approche synergétique.

Une méthode hybride peut être mauvaise ou bonne selon le choix et les rôles de ses composants. Pour définir une méthode hybride efficace, il faut savoir caractériser les avantages et les limites de chaque méthode.

Actuellement, les approches hybrides ont permis d'obtenir de bons résultats dans une grande variété de problèmes théoriques d'optimisation combinatoire tel que le problème du voyageur de commerce, le problème de coloration de graphe, le problème d'affectation quadratique, le problème de tournée de véhicules, le séquençage d'ADN ou encore le calcul des trajectoires des satellites.
Les premières approches hybrides proposées ont combiné des méta-heuristiques à base de populations (algorithmes génétiques) avec des méta-heuristiques à solution unique (APE ou recherche tabou).

A noté que les méthodes approchées ne procurent pas forcément une solution optimale, mais seulement une bonne solution (de qualité raisonnable) en un temps de calcul aussi faible que possible.

Par conséquent, l'hybridation de méta-heuristiques avec les méthodes approchées peut devenir une alternative très intéressant.

II.1. Les fourmis réel et l'inspiration biologique

Les différentes méta-heuristiques ont été inspirées par les travaux des biologistes qui ont longuement été intrigués par le comportement des insectes sociaux en générale (abeilles, termites…) et les fourmis en particulier (1983 Deneubourg Jean Louis Biologiste, étudie le comportement des fourmis). Ces derniers sont capables de résoudre collectivement des problèmes complexes, comme trouver le plus court chemin entre une source de nourriture et leur nid par des moyens très simples dans un environnement accidenté. Pour cela, elles communiquent entre elles de façon locale et indirecte, grâce à une hormone volatile, appelée Phéromone.

La fourmi ; Au cours de sa progression ; laisse derrière elle une trace de phéromone (une sorte de marquage du chemin, elle fournit une information sur la qualité des chemins empruntés afin d'attirer et guider les fourmis) [17]. Ce procédé basé sur le mécanisme de rétroaction positive, assure que pendant le fourragement pour la nourriture, les fourmis utilisent la voie d'accès la plus courte car elle sera le plus imprégnée par la phéromone. [18]

Ce phénomène est appelé autocatalytique. Le temps nécessaire à la stabilisation du système est imputable au fait qu'aucune phéromone n'est présente sur aucune route au départ, les fourmis ont alors autant de chances de choisir le plus court que le plus long chemin. [16]

On peut citer plusieurs raisons à cette inspiration:

a) l'influence des fourmis sur leur environnement naturel est extrêmement importante. Il a par exemple été montré (qu'elles déplacent plus de terre en forêt tropicale que les vers de terre, ou encore que le poids total des fourmis sur terre est du même ordre de grandeur que le poids des humains de plus, la domination des fourmis est une preuve de leur adaptation à des environnements très variés) ;

b) l'étude des fourmis se fait assez facilement en laboratoire car elles s'adaptent sans trop de difficultés à des environnements différents de leur habitat d'origine ;

c) les fourmis possèdent une gamme de comportements très variés, collectifs ou individuels. [19]

II.2. L'intelligence collective des fourmis

Malgré que certaines espèces des fourmis aient des capacités individuelles étonnantes telle que des capacités visuelles inhabituelles et des capacités d'apprentissage, mais la plupart des caractéristiques qui nous intéressent sont cependant collectives.

On parle d'intelligence collective quand un groupe social peut résoudre un problème dans un cas où un agent isolé en serait incapable. Cette intelligence est basée sur les processus d'auto-organisation.

L'auto-organisation se parée bien à l'étude des insectes sociaux montrent des comportements collectifs complexes issus de comportements individuels simples. On peut regrouper les processus d'auto-organisation chez les insectes sociaux en quatre groupes tant leur diversité est importante. [20]

– la division du travail et l'organisation des rôles sociaux : à l'intérieur d'une même société, on peut observer différentes catégories spécialisées dans un certain nombre de tâches (la recherche de nourriture, la défense du nid, …) ;

– l'organisation de l'environnement : la construction du nid est un symbole de l'organisation distribuée des insectes. Le nid est construit sans que les insectes soient dirigés, ils répondent à un certain nombre de stimuli provenant de leur environnement ;

– la reconnaissance inter-individuelle : chaque fourmi est capable d'identifier ses congénères tout en participant elle-même à l'identité de sa colonie (par exemple l'échange d'aliments entre les individus d'une même colonie '*trophalaxie'*) ;

– le recrutement et l'exploitation collective des sources de nourriture : le fourragement met à jour des stratégies qui permettent aux insectes une grande adaptation à leur milieu.

Les capacités des fourmis en matière de coopération, de communication, de compétition et d'apprentissage, entre autres, peuvent être mises à profit pour la conception d'algorithmes de résolution des problèmes d'optimisation. [21]

II.3. Les fourmis artificielles

Le terme « fourmi » est un mot qui se profile plusieurs domaines : celui de la biologie ou plus précisément de la myrmécologie qui est l'étude du comportement naturel des fourmis, celui de la robotique qui utilise leur comportement pour concevoir de nouvelles machines, et celui de l'informatique où ces créatures sont modélisées pour la simulation ou la création d'algorithme.

Les différentes applications informatiques qui découlent des capacités de communication des fourmis se retrouvent par exemple en optimisation combinatoire où la coopération stigmergétique s'applique parfaitement à la recherche du plus court chemin dans un graphe. [21]

II.4. L'algorithme de colonies de fourmis pour résoudre le PVC [22]

L'idée d'imiter le comportement des fourmis pour trouver de bonnes solutions à des problèmes d'optimisation combinatoire a été proposée par Colorni & Al.

Le principe de cette méta-heuristiques est basé sur la manière dont les fourmis cherchent leur nourriture et retrouvent leur chemin pour retourner dans la fourmilière.

Initialement, les fourmis explorent les environs de leur nid de manière aléatoire. Sitôt qu'une source de nourriture est repérée par une fourmi, son intérêt est évalué le chemin en matière de quantité et de qualité pour ramène un peu de nourriture au nid et pour retrouver son chemin, elle prend soin de laisser derrière elle une phéromone, trace chimique qu'elle arrive à détecter.

Durant le chemin du retour, elle dépose également une quantité de phéromone dépendant de l'intérêt de la source de nourriture. Comme toutes les fourmis font de même, les traces laissées augmentent plus rapidement pour les sources de nourritures proches de la fourmilière, et lorsque plusieurs traces mènent à la même source, les traces correspondant aux chemins les plus courts sont renforcées à un rythme plus élevé.

Après un certain temps, les chemins les plus rapides menant à la source de nourriture sont marqués par des traces plus importantes (Figure.II.1).

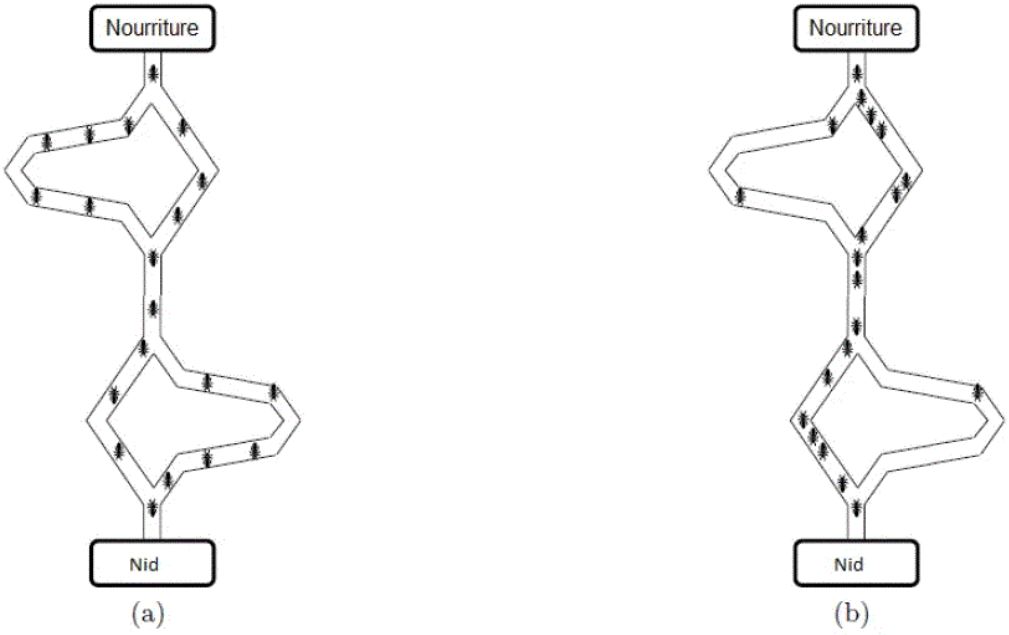

Figure.II.1 : Expérience de sélection des branches les plus courtes par une colonie de fourmis:

(a) Au début de l'expérience, (b) à la fin de l'expérience. [22]

Pour transposer ce comportement à un algorithme général d'optimisation combinatoire, on fait une analogie entre la zone dans laquelle les fourmis cherchent de la nourriture et l'ensemble des solutions admissibles du problème, entre la quantité ou la qualité de la nourriture et la fonction objectif à optimiser, et enfin entre les traces de phéromones et une mémoire adaptative.

Le problème du voyageur de commerce (PVC ou TSP : Travelling Salesman Problem) a fait l'objet de la première implémentation d'un algorithme de colonies de fourmis.

Le passage de la métaphore à l'algorithme est ici relativement facile et le problème du voyageur de commerce est bien connu et étudié.

Le problème du voyageur de commerce consiste à trouver le trajet le plus court (tournée / tour) reliant n villes données et chaque ville ne devant être visitée qu'une seule fois. Ce problème est plus généralement défini comme un graphe complètement connecté (N, A) où les villes sont les nœuds N et les trajets entre ces villes, les arêtes A.

Dans l'algorithme Ant System, à chaque itération ($1 \leq t \leq t_{max}$),chaque fourmi k ($k = 1,\ldots, m$) parcourt le graphe et construit un trajet complet de $n = |N|$étapes (on note $|N|$le cardinal de l'ensemble N)

Pour chaque fourmi, le trajet entre une ville i et une ville j dépend de:

- la liste des villes déjà visitées, qui définit les mouvements possible à chaque pas, quand la fourmi k est sur la ville i: J_i^k ;
- L'inverse de la distance entre les villes: $n_{ij} = 1/d_{ij}$, appelé visibilité. Cette information statique est utilisée pour diriger le choix des fourmis vers des villes proches et éviter les villes trop lointaines ;
- La quantité de phéromone déposée sur l'arête reliant les deux villes, appelée l'intensité de la piste. Ce paramètre définit l'attractivité d'une partie du trajet global et change à chaque passage d'une fourmi. La matrice de phéromones où chaque entrée correspond à l'intensité de chaque arête constitue la mémoire adaptative de l'algorithme.

Par conséquent, la règle de déplacement d'une fourmi est la suivante:

$$p_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^a\,(n_{ij})^\beta}{\sum_{l \in J_i^k}(\tau_{ij}(t))^a\,(n_{ij})^\beta} & si\ j \in J_i^k \\ 0 & sinon \end{cases} \qquad \text{II. 1}$$

Où α et β sont deux paramètres contrôlant l'importance relative de l'intensité de la piste $\tau_{ij}(t)$ et de la visibilité n_{ij}.

Avec $\alpha = 0$, seule la visibilité de la ville est prise en compte; la ville la plus proche est donc choisie à chaque pas.

Au contraire, Avec $\beta = 0$, seules les pistes de phéromone sont prises en compte.

Pour éviter une sélection trop rapide d'un trajet, un compromis entre ces deux paramètres jouant sur les comportements de diversification et d'intensification est nécessaire.

Après un tour complet, chaque fourmi laisse une certaine quantité de phéromone $\Delta\tau_{ij}^{k}(t)$ sur l'ensemble de son parcours, quantité qui dépend de la qualité de la solution trouvée :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L^{k}(t)} & si(i,j) \in T^{k}(t) \\ 0 & sinon \end{cases} \qquad \text{II. 2}$$

Où:

$T^{k}(t)$ est le trajet effectué par la fourmi k à l'itération t ;

$L^{k}(t)$ la longueur de la tournée ;

Q un paramètre fixé.

L'algorithme ne serait pas complet sans le processus d'évaporation des pistes de phéromone.

En effet, pour éviter d'être piégé dans des solutions sous-optimales, il est nécessaire de permettre au système d'oublier les mauvaises solutions en contrebalançant l'additivité des pistes par une décroissance constante des valeurs des arêtes à chaque itération.

La règle de mise à jour des pistes est donc comme suit :

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \qquad \text{II.3}$$

Où :

- m est le nombre de fourmis ;

- ρ le taux d'évaporation.

La quantité initiale de phéromone sur les arêtes est une distribution uniforme d'une petite quantité $\tau_0 \geq 0$.

Le pseudo-code de l'algorithme ACS est présenté dans l'algorithme II.1.

Algorithme II.1. Algorithme de colonies de fourmis (Ant System) [22]

Pour $t = 1, \dots, t_{\max}$ **faire**

Pour chaque fourmi $k = 1, \dots, m$ **faire**

– Choisir une ville au hasard.

Pour chaque ville non visitée i **faire**

–Choisir une ville j, dans la liste j_i^k des villes restantes, selon la formule II. 1

Fin Pour

– Déposer une piste $\Delta\tau_{ij}^k(t)$ sur le trajet $T^k(t)$ conformément à l'équation II. 2

Fin Pour

– Evaporer les pistes selon la formule II.3

Fin Pour

L'algorithme Ant Colony System (ACS) a été introduit plus tard pour améliorer les performances du premier algorithme (AS).

Par conséquent ACS est fondé sur des modifications du AS, à savoir :

1- ACS introduit une règle de transition dépendant d'un paramètre q_0 ($0 \leq q0 \leq 1$), qui définit une balance diversification/intensification.

Une fourmi k sur une ville i choisira une ville j par la règle:

$$j = \begin{cases} argmax_{u \in j_i^k}\left[(\tau_{iu}(t))(n_{ij})^{\beta}\right] & siq \leq q_0 \\ J & siq > q_0 \end{cases} \qquad \text{II.4}$$

Où q est une variable aléatoire uniformément distribuée sur $[0.1]$ et $J \in J_i^k$ une ville sélectionnée aléatoirement selon la probabilité:

$$p_{ij}^k(t) = \frac{\left(\tau_{iJ}(t)\right)\left(n_{iJ}\right)^{\beta}}{\sum_{l \in J_i^k}\left(\tau_{iJ}(t)\right)\left(n_{iJ}\right)^{\beta}} \qquad \text{II.5}$$

En fonction du paramètre q_0, il y a donc deux comportements possibles:

- si $q > q_0$ le choix se fait de la même façon que pour l'algorithme AS, et le système tend à effectuer une diversification;
- si $q \leq q_0$, le système tend au contraire vers une intensification.

  En effet pour $q \leq q_0$, l'algorithme exploite davantage l'information récoltée par le système, il ne peut pas choisir un trajet non exploré.

2- La gestion des pistes est séparée en deux niveaux: une mise à jour locale et une mise à jour globale. Chaque fourmi dépose une piste lors de la mise à jour locale:

$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\,\tau_0$ II.6 Où τ_0 est la valeur initiale de la piste.

A chaque passage, les arêtes visitées voient leur quantité de phéromone diminuer, ce qui favorise la diversification par la prise en compte des trajets non explorés.

A chaque itération, la mise à jour globale s'effectue comme ceci:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\,\Delta\tau_{ij}(t) \qquad \text{II.7}$$

Où les arêtes (i, j) appartiennent au meilleur tour T^+ de longueur L^+et $\Delta\tau_{ij}(t) = 1/L^+$.

Ici, seule la meilleure piste est donc mise à jour, ce qui participe à une intensification par sélection de la meilleure solution.

3- Le système utilise une liste de candidats. Pour chaque ville, cette liste stocke les plus proches villes voisines, classées par distances croissantes.
   Une fourmi ne prendra en compte une arête vers une ville en dehors de la liste que si celle-ci déjà été explorée.

Concrètement, si toutes les arêtes sont déjà été visitées dans la liste de candidats, le choix se fera en fonction de la règle II. 5, sinon c'est la plus proche des villes non visitées qui sera choisie.

Du point de vue de la PMA, la mémoire adaptative est mise en place par la matrice de phéromones (pistes de phéromones), cette mémoire s'adapte aux nouvelles connaissances acquises en cours de la recherche.

En ce concerne l'auto-organisation, les rétroactions positives correspondent à l'attractivité des pistes de phéromones. Cette attractivité est maintenue sous contrôle par le mécanisme d'évaporation qui constitue la boucle de rétroaction négative.

Notons que : NC est un cycle.

Algorithme II.2. Algorithme OCF [23]

```
Début
    Entrée m : nombre de fourmis par colonies ;
    n : nombre de villes ;
    N_c ← 0 ;
    Initialiser liste Taboue // avec la ville de départ de chaque fourmi ;
    Initialiser la matrice τ_ij
    Tant que (N_c< N_max) et (convergence non atteinte) Faire
        pour i 1 à n faire
            pour j 1 à m faire
                Sélectionner la ville j à être ajoutée ou tour en cour selon la
                  formule p_ij^k(t);
                Effectuer la mise à jour locale de la trace selon la paire de
                  Villes (i, j) ;
            fin pour
        fin pour
    Fin tant que
    pour chaque fourmi k faire
        Evaluer la solution k pour chacune des étapes ;
        Insérer la solution k dans la liste TABOU ;
        Effectuer la mise à jour Globale de la trace selon la meilleure solution du
          cycle ;
    fin pour
    NC ← NC + 1;
Fin
```

II.5. L'algorithme adjacent pairwaise exchange APE [24]

APE est une heuristique de recherche locale, permettant d'obtenir des résultats faisables dans un délai raisonnable, elle est utilisée pour améliorer la qualité d'une solution donnée.

L'algorithme APE commence avec une tournée admissible donnée. Ensuite, elle cherche dans le voisinage de la solution courante toute tournée pour améliorer la configuration actuelle.

À chaque étape de la procédure d'amélioration, l'algorithme examine si l'échange de deux sommés adjacentes (villes) produit une tournée plus courte pour améliorer la tournée et l'algorithme continue ainsi jusqu'à ce qu'aucune amélioration ne soit plus possible.

Les étapes de l'algorithme se résume comme suit :

Étape 1 : prendre deux sommés adjacentes de la tournée : A et B ;

Étape 2 : vérifier si la distance ...B, A, C, D... est inférieur à la distance ...A, B, C, D... ;

Étape 3 : si vérifiée, alors échanger A avec B ;

Étape 4 : si amélioration de la tournée, alors aller à 1, sinon s'arrêter.

Exemple :

Soit la tournée suivante : 5-1-4-3-2 avec la longueur L=30.

Selon l'algorithme, les modifications apportées sont les suivantes :

Premier modifications de l'algorithme : 1-5-4-3-2, avec une longueur de : L=38 ;

Deuxième modifications de l'algorithme : 5-4-1-3-2, avec une longueur de : L=35 ;

Troisième modifications de l'algorithme : 5-1-3-4-2, avec une longueur de : L=29 ;

Quatrième modifications de l'algorithme : 5-1-5-2-3, avec une longueur de : L=31.

Parmi les propositions de l'algorithme la meilleur est celle de la longueur 29 et comme la longueur 29 est inferieur a la longueur 30, la meilleure tourner est celle du "5-1-3-4-2".

En conclusion, l'algorithme s'arrêt, si et seulement si la longueur 29 est la meilleure solution, sinon il contenu la recherche.

II.6. Présentation des modèles

Certaines méthodes ont été développées pour résoudre le PVC et l'une d'entre elles est la méthode hybride proposée par Mahi et al. (2015), basée sur l'algorithme Ant Colony Optimisation (ACO) (Dorigo, 1996, deneubourget Goss, 1989), Particle Swarm Optimisation (PSO) (Kennedy, 1995, Reynold, 1987, Heppner et Grenander, 1990) et l'algorithme 3-Opt.

Tout d'abord, nous présentons le modèle de Mahi avec les deux modifications proposées dans l'article. [25]

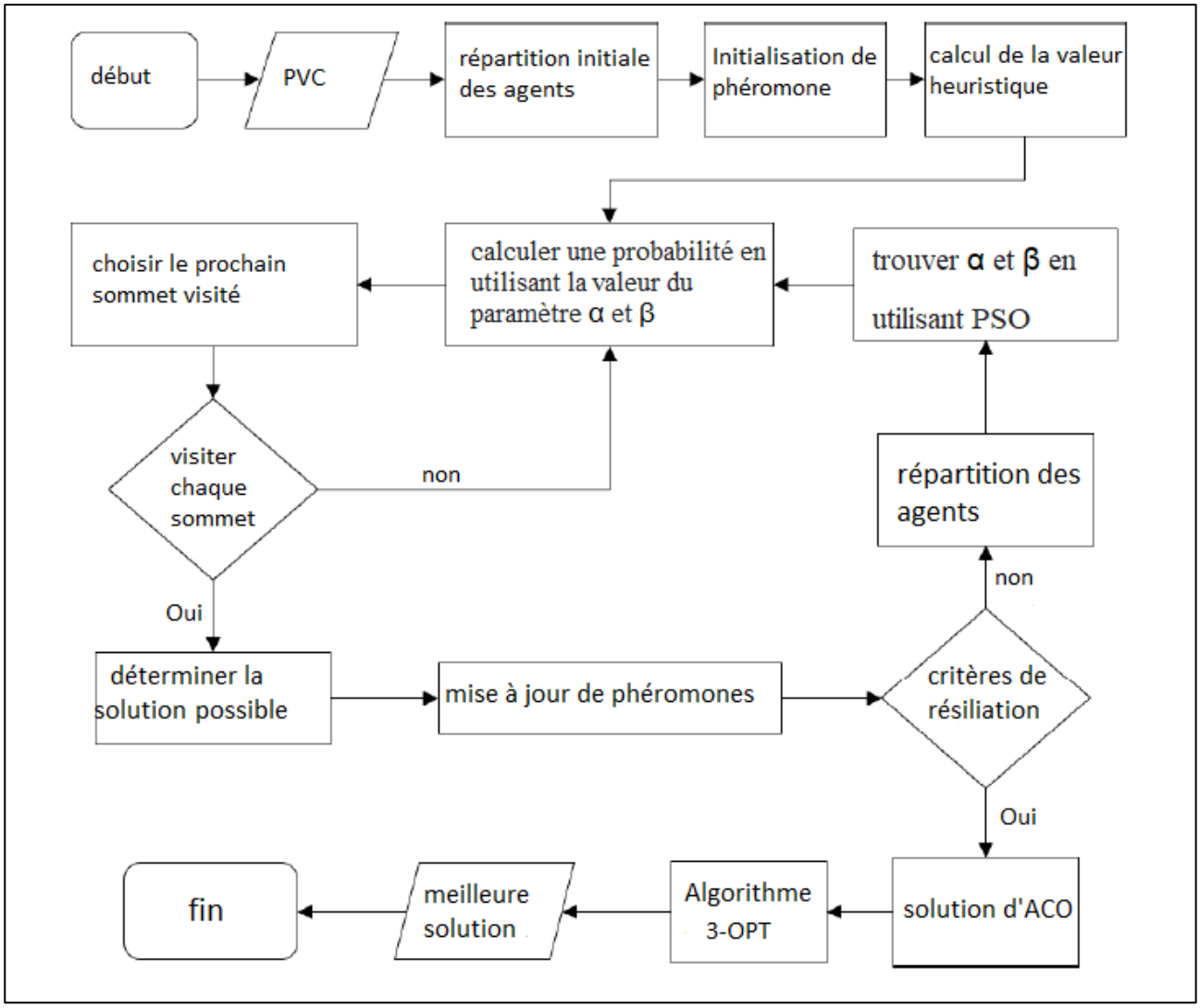

Figure.II.2 : La méthode hybride da MAHI & AL [25]

La différence entre ces modifications réside dans l'utilisation de l'algorithme 3-Opt pendant le processus. Dans la première méthode de modification, l'algorithme 3-Opt est utilisé pour réduire la distance totale de la solution réalisable qui est obtenue à partir de chaque itération. Pendant ce temps, dans la deuxième modification, l'algorithme 3-Opt est utilisé pour réduire la distance totale de tous solutions obtenues à chaque itération.

II.6.1. Description de la méthode proposée par Mahi & AL [25]

La méthode hybride commence par mettre en œuvre les étapes ACO, qui se composent de:

- ✓ Processus initial de distribution des agents ;
- ✓ Initialisation de la phéromone ;
- ✓ Calcul de la valeur heuristique.

Afin de créer des solutions, la deuxième étape de l'ACO calcule une probabilité en utilisant les valeurs des paramètres α et β. Un agent visitera le prochain sommet en fonction de cette valeur de probabilité. Cette étape sera répétée jusqu'à ce que chaque agent ait visité chaque sommet.

La solution réalisable ; qui est l'itinéraire avec la distance totale minimale ; sera déterminée à partir de l'itinéraire de tous les agents.

La troisième étape est utilisée pour mettre à jour la phéromone de la solution réalisable.

Le processus ACO sera terminé par des critères de résiliation. Si les critères ne remplissent pas la condition de résiliation, le processus sera poursuivi en redistribuant les agents aux sommets.

Dans cette phase, le PSO sera utilisé pour déterminer les nouvelles valeurs de paramètre α et β qui seront utilisés dans les prochains processus ACO.

L'ensemble du processus sera répété jusqu'à ce que les critères de tertiarisations soient remplis et que la solution candidate puisse être obtenue à partir de toutes les solutions réalisables.

Ces solutions candidates seront optimisées par l'algorithme 3-Opt pour obtenir la meilleure solution.

Dans notre, étude nous représentons le même modèle que le précédent avec une légère modification et nous remplacerons l'algorithme 3-Opt avec l'algorithme APE pour construire un nouveau modèle hybride (ACO+PSO+APE), si joint la figure.II.3.

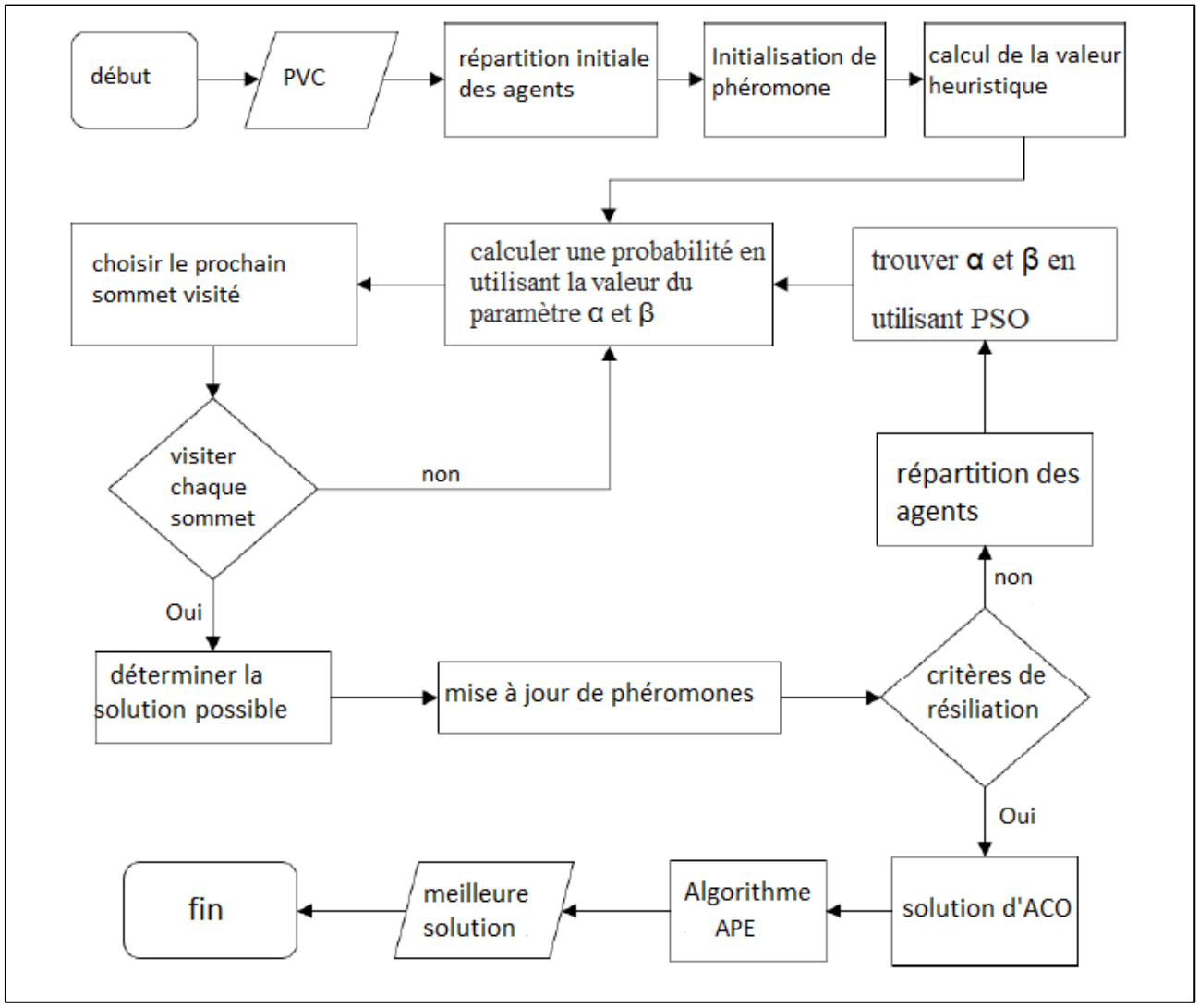

Figure.II.3 : Notre modèle hybride.

A la fin de l'étude, nous comparons les deux modèles pour déterminer le meilleur.

Conclusion

Dans ce chapitre, nous avons présenté une introduction à les méthodes hybrides. Ensuite, nous avons donné une explication de l'inspiration biologique. Puis, on a discuté sur les fourmis réelles et les fourmis artificielles. Nous avons aussi parler en détail sur les deux algorithme (l'algorithme de colonies de fourmies ACO et l'algorithme adjacent pairwaise exchenge APE). Enfin, nous avons entré un peu plus dans le vif du sujet de cette mémoire, à savoir les algorithmes hybrides pour résoudre le problème de voyageur de commerce.

Après avoir présenté les deux modèles hybrides dans ce chapitre, le chapitre suivant étudient la comparaison de ces algorithmes appliqués au problème de voyageur de commence.

Chapitre III

IMPLEMENTATIONS ET RESULTATS

Introductions

Ce chapitre est consacré à la contribution de notre mémoire, consistant à comparer les résultats des algorithmes ceux des stratégies appliquées sur le modelé de MAHI & AL et ceux de notre modèle.

A rappeler que, l'algorithme de MAHI & AL est composé de ACO, PSO et 3-OPT que nous allons modifier en remplaçant l'algorithme 3-opt par APE.

L'objectif de cette partie est de comparer les deux algorithmes par des tests pour sélectionner le meilleur modèle.

Durant cette étude, nous avons utilisé la version MATLAB R2017a pour effectuer les expériences.

III.1. Langage de programmation (MatLab) [26]

MATLAB est une abréviation de Matrix LABoratory. La version actuelle ; écrite en C par MathWorks Inc ; existe en version professionnelle et en version étudiant, sa disponibilité est assurée sur plusieurs plateformes.

MATLAB est un environnement puissant, complet et facile à utiliser, destiné au calcul scientifique. Il permet de réaliser des simulations numériques basées sur des algorithmes d'analyse numérique ou des algorithmes génériques.

MATLAB est considéré comme un des meilleurs langages de programmation (C ou Fortran), il possède les particularités suivantes par rapport à ces langages :

- Programmation facile ;
- Continuité parmi les valeurs entières réelles et complexes ;
- Gamme étendue des nombres et leurs précisions ;
- Bibliothèque mathématique très compréhensive ;
- Outil graphique qui inclut les fonctions d'interface graphique et les utilitaires ;
- Possibilité de liaison avec les autres langages classiques de programmations (C ou Fortran).

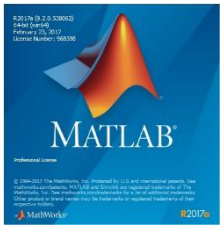

Figure.III.1 : MATLAB R2017a

III.2. Résultats et discutions

III.2.1. Paramètres utilisés

L'implémentation utilise six problèmes de benchmarks de TSPLIB (Reinelt, 1991) : Eil51, Berlin52, St70, Eil76, Rat99 et kroA200 et chaque problème sera testé en 10 exécutions avec 50 itérations.

Pour trouver la solution moyenne (SM) on utilise l'équation suivante :

$$SM = \frac{\sum_{i=1}^{n} D_i}{n} \qquad \text{III. 1}$$

tel que D_i est la solution du problème i, i = 1,2,…,n.

Le pourcentage d'erreur relative (ER) est utilisé pour déterminer la qualité de la méthode de résolution du PVC. Ce pourcentage est calculé par l'équation ci-après :

$$ER = \frac{SM - BKS}{BKS} \times 100 \qquad \text{III. 2}$$

Où BKS est la solution la plus connue.

Dans l'implémentation, nous utilisons également la valeur des paramètres suggérés dans les articles précédents, comme suit : $\rho = 0{,}1$ et $Q = 100$ comme meilleure valeur constante pour la méthode ACO (Dorigo, 1996).

Nous utilisons également $0 \leq \alpha \leq 2$ et $0 \leq \beta \leq 2$, suggéré par MAHI & AL en 2015 pour obtenir le meilleur résultat pour la méthode ACO.

Aussi, MAHI & AL ; pour la même année ; suggère également d'utiliser dix agents lorsque nous utilisons la méthode hybride composer des trois l'algorithme ACO, PSO et 3-Opt.

III.2.2. Résultats

III.2.2.1. Description de la première stratégie

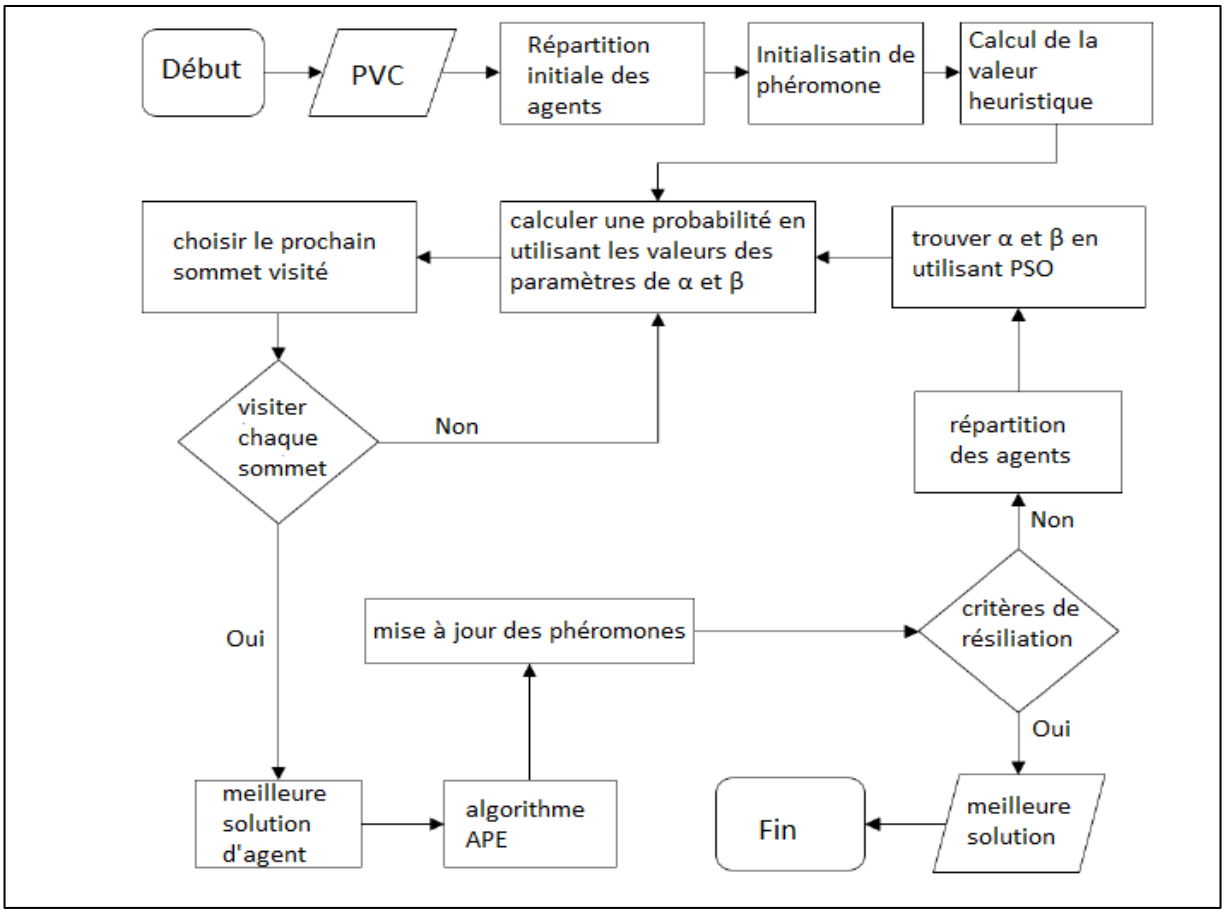

Figure.III.2 : La première méthode de modification hybride.

Selon la première stratégie, l'algorithme 3-Opt est appliqué à la solution candidate.

En parallèle dans notre première stratégie, on a utilisé l'algorithme APE qui sera appliqué à la solution réalisable ; comme décrit dans la figure.III.2.

Les différences peuvent être observées après que chaque agent a visité chaque sommet (visitation terminée) et la solution réalisable sera choisie parmi toutes les solutions générées en appliquant l'algorithme APE.

Une fois les solutions réalisables sont sélectionnées, on applique le troisième processus ACO qui met à jour la phéromone. Et enfin la suite de notre stratégie suit des mêmes étapes de MAHI & AL jusqu'à ce que le processus satisfasse aux critères terminaux.

Lorsque les critères sont remplis, la meilleure des solutions réalisables sera la meilleure solution de la méthode hybride de première modification.

```
Adjacent Pairwaise Exchange
     fixed_indexes = [];
     iteration = 0;
     while iteration < length(BS.Tour)
         iteration = iteration + 1;
         for i = 1:length(BS.Tour) - 1
             best_swap = [];
             can_swap = ~ismember(i, fixed_indexes) && ~ismember(i + 1,
                        fixed_indexes);
             if can_swap
                 swapped_tour = BS.Tour;
                 current_city = swapped_tour(i);
                 swapped_tour(i) = swapped_tour(i + 1);
                 swapped_tour(i + 1) = current_city;
                 swapped_tour_cost = CostFunction(swapped_tour);
                 if swapped_tour_cost < BS.Cost
                     BS.Tour = swapped_tour;
                     BS.Cost = swapped_tour_cost
                     best_swap(end + 1) = i;
                     best_swap(end + 1) = i + 1;
                 end
             end
             fixed_indexes = horzcat(fixed_indexes, best_swap);
         end
     end
```

Algorithme III.1 Implémentation de l'algorithme APE de la première stratégie en utilisant MATLAB.

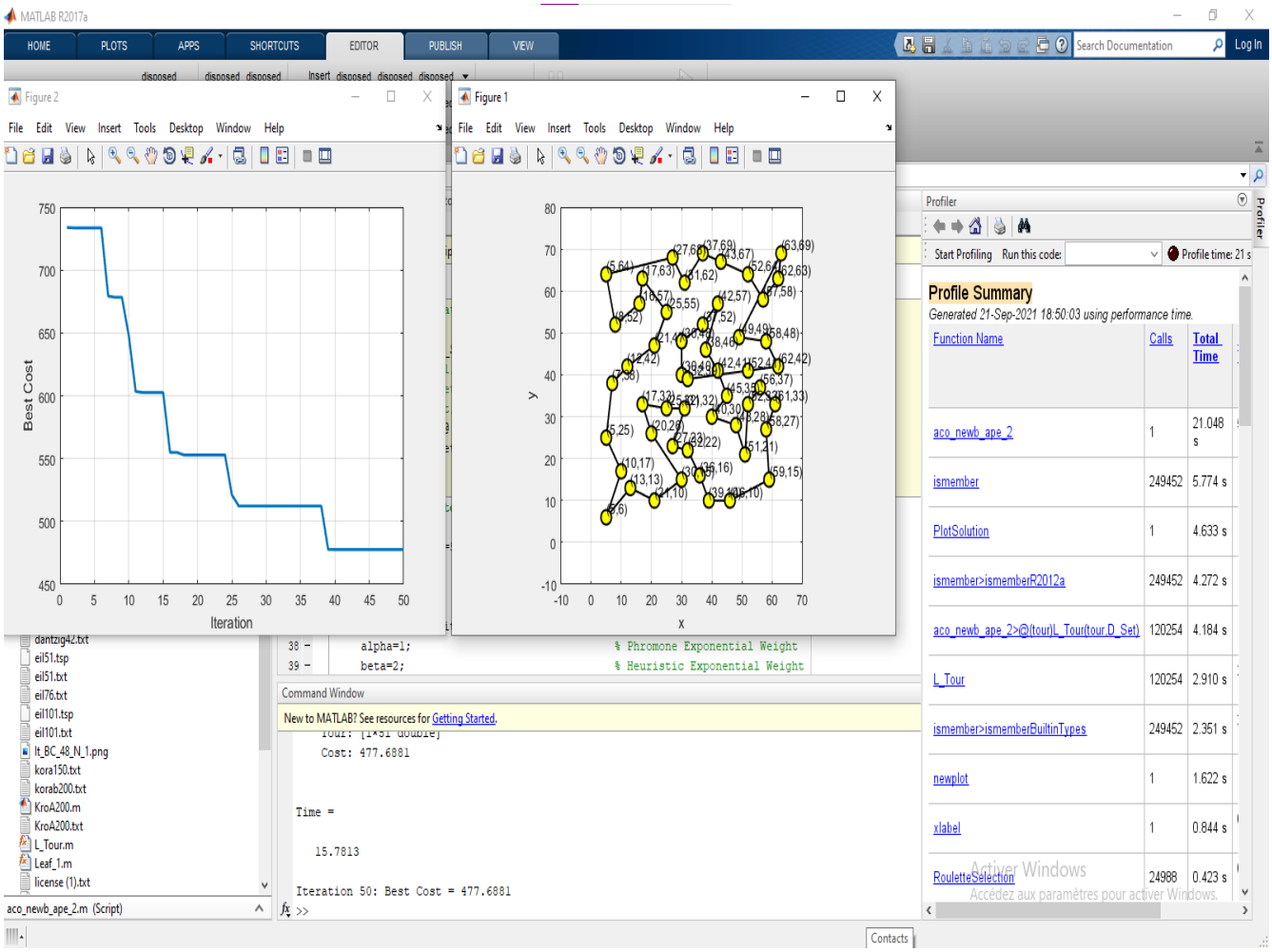

Figure.III.3 : L'interface des résultats

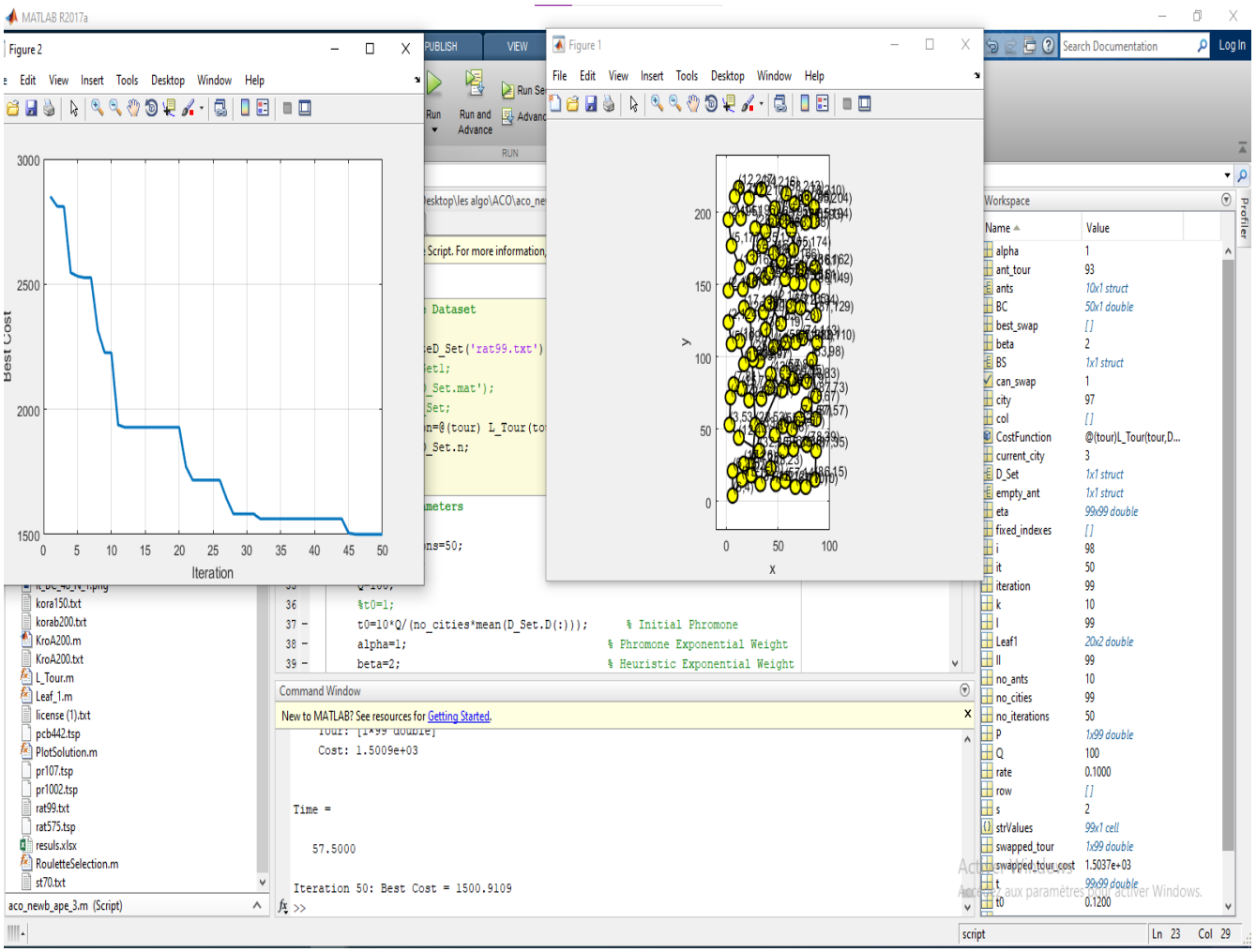

Figure.III.4 : L'interface des résultats

Tableau.III.1 : Les résultats de la première stratégie du premier modèle.

Problèmes	BKS	Meilleur Solution	Mauvaise Solution	La solution moyenne	ER (%)	Temps (seconde)
Eil51	426	437	574	480.8	12.86	325.55
Berlin52	7542	7930	8832	8351.6	10.73	328.27
St70	675	683	777	732.8	8.56	427.28
Eil76	538	605	663	634.4	17.92	488.64
Rat99	1211	1311	1499	1439.6	18.88	818.51
KroA200	29368	29957	36761	34039.4	15.91	1042.13

Tableau.III.2 : Les résultats de la première stratégie de notre modèle.

Problèmes	BKS	Meilleur Solution	Mauvaise Solution	La solution moyenne	ER (%)	Temps (seconde)
Eil51	426	475.3056	512.4759	489.6684	14.9456338	15.83438
Berlin52	7542	7952.5747	8708.3662	8326.47373	10.4014019	16.25938
St70	675	738.8907	796.5697	782.29809	15.8960133	28.05157
Eil76	538	612.6328	649.7246	627.38494	16.6143011	34.45469
Rat99	1211	1407.315	1495.9729	1451.22496	19.8369083	62.27501
KroA200	29368	36141.2749	39010.6978	37703.1073	28.3815967	285.31406

Selon l'expérience de la première stratégie, les résultats affichés sous le tableau numéro 1, la meilleure solution est plus proche de BKS que la nôtre, ainsi que l'écart entre la meilleur et la mauvaise solution est plus petit ce qui donne une erreur relative inferieure à l'erreur relative de notre expérience.

A noté que l'erreur pour les deux expériences est supérieure à la norme 8%.

En conclusion et en matière de résultats affichés, l'expérience de la première stratégie dans l'étude précédente donne de meilleures solutions plus proches de la BKS en temps élevé alors que notre méthode prend moins de temps pour afficher les résultats.

III.2.2.2. Description de la deuxième stratégie

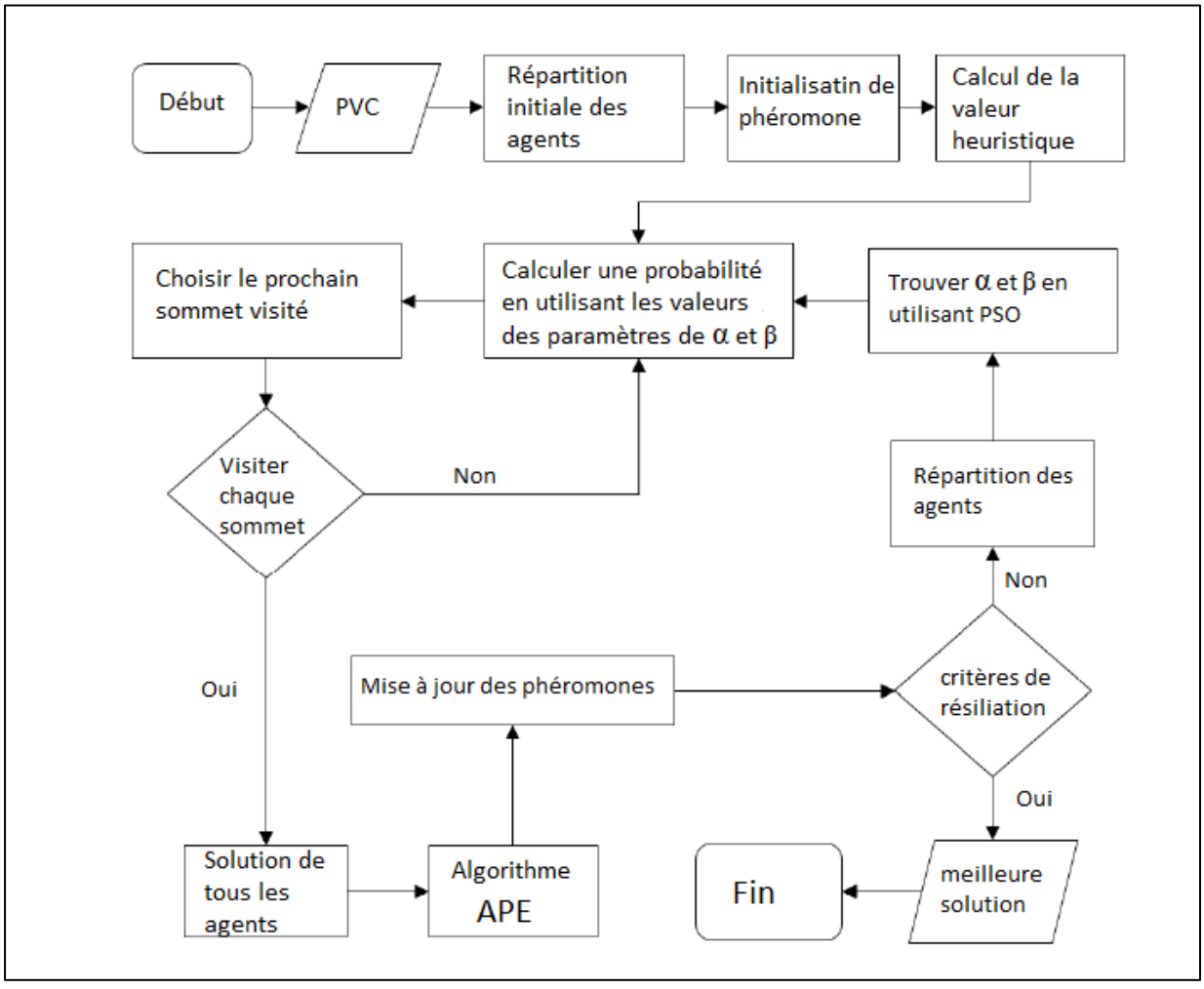

Figure.III.5 : La deuxième méthode de modification hybride.

Dans la deuxième méthode de modification hybride, nous implémentons l'algorithme APE à la solution de tous les agents à chaque itération, comme le montre la figure.III.3.

Nous rappelons que dans la première méthode de modification, l'algorithme APE n'est appliqué qu'à la solution réalisable dans chaque itération ACO, tandis que dans la deuxième méthode de modification, l'algorithme APE est appliqué à la solution de tous les agents à chaque itération.

Une fois que toutes les solutions ont été optimisées par APE, la solution avec la distance totale minimale est sélectionnée comme solution réalisable.

La prochaine étape est la troisième étape de l'ACO qui est la mise à jour de la phéromone.

Le processus se poursuivra jusqu'à ce qu'il satisfasse aux critères terminaux et que la meilleure solution soit obtenue.

```
% Adjacent Pairwaise Exchange
    all_solutions = ants
    for solution_index = 1:length(all_solutions)
        solution = all_solutions(solution_index)
        fixed_indexes = [];
        iteration = 0;
        while iteration < length(solution.Tour)
            iteration = iteration + 1;
            for i = 1:length(solution.Tour) - 1
                best_swap = [];
                can_swap = ~ismember(i, fixed_indexes) && ~ismember(i + 1,
                           fixed_indexes);
                if can_swap
                    swapped_tour = solution.Tour;
                    current_city = swapped_tour(i);
                    swapped_tour(i) = swapped_tour(i + 1);
                    swapped_tour(i + 1) = current_city;
                    swapped_tour_cost = CostFunction(swapped_tour);
                    if swapped_tour_cost < BS.Cost
                        BS.Tour = swapped_tour;
                        BS.Cost = swapped_tour_cost
                        best_swap(end + 1) = i;
                        best_swap(end + 1) = i + 1;
                    end
                end
                fixed_indexes = horzcat(fixed_indexes, best_swap);
            end
        end
    end
```

Algorithme III.2 Implémentation de l'algorithme APE de la deuxième stratégie en utilisant MATLAB.

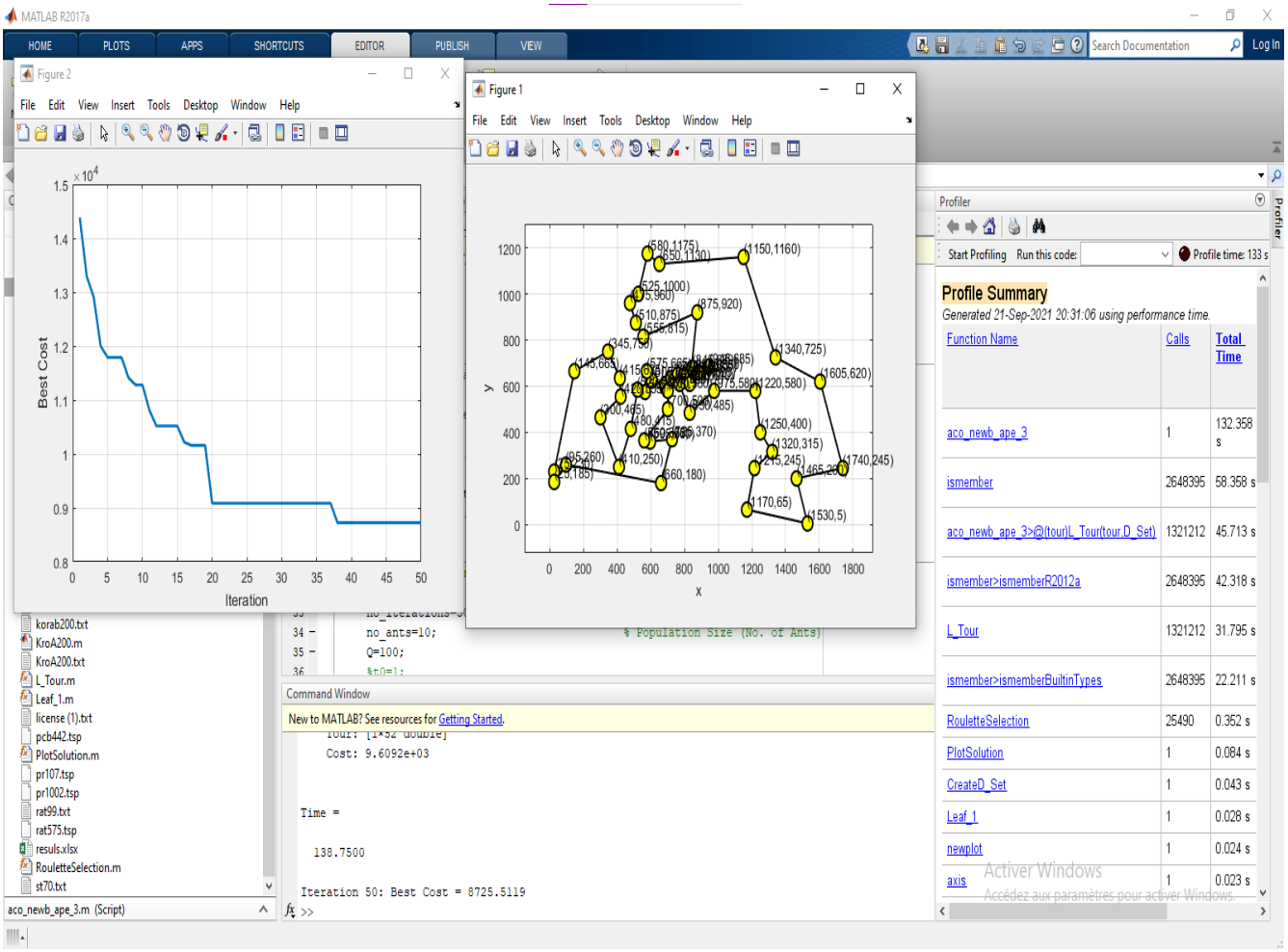

Figure.III.6 : L'interface des résultats

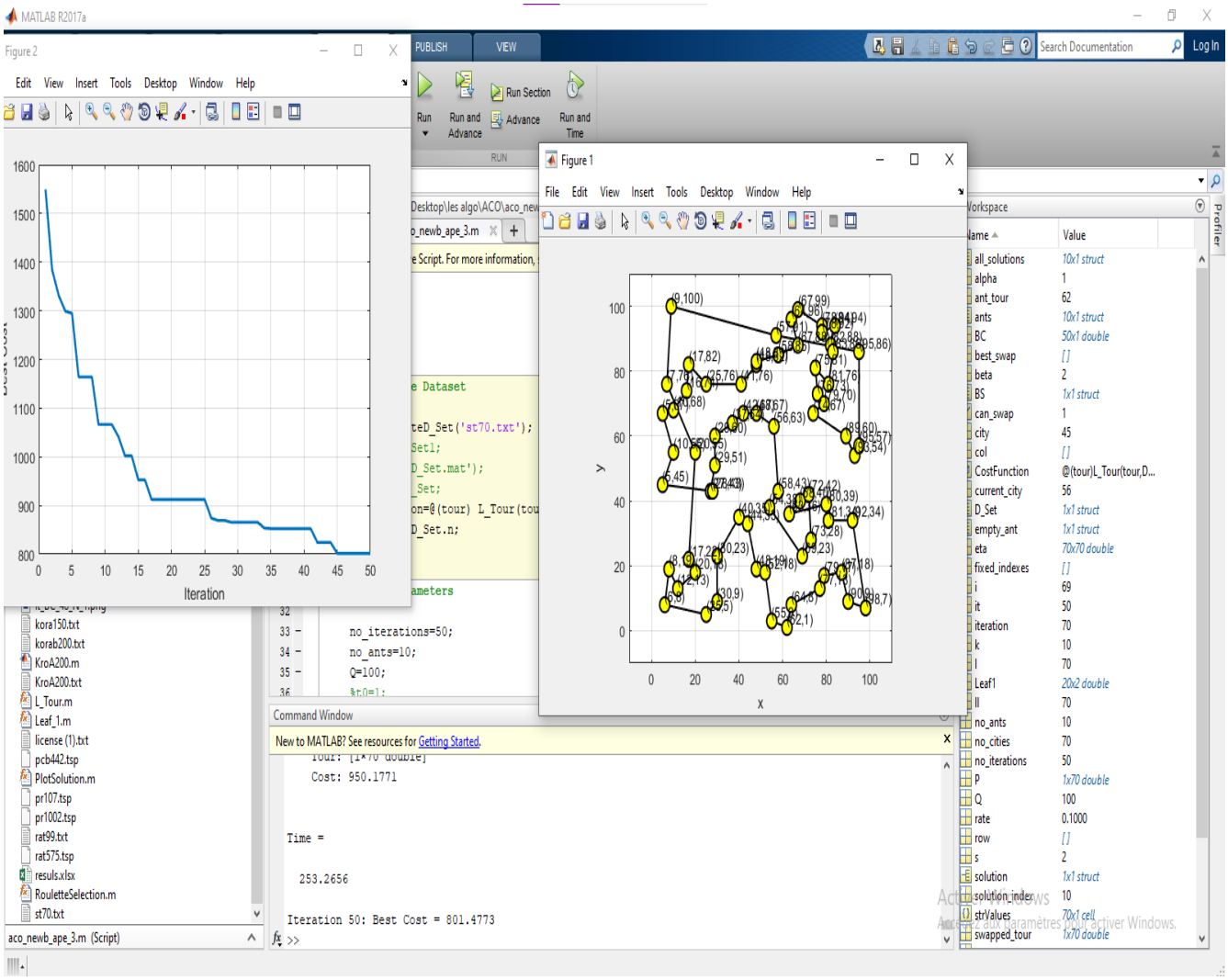

Figure.III.7 : L'interface des résultats

Tableau.III.3 : Les résultats de la deuxième stratégie du premier modèle.

Problèmes	BKS	Meilleur Solution	Mauvaise Solution	La solution moyenne	ER (%)	Temps (seconde)
Eil51	426	426	426	426	0.00	658.43
Berlin52	7542	7542	7542	7542	0.00	547.11
St70	675	675	675	675	0.00	1138.9
Eil76	538	538	538	538	0.00	1413.8
Rat99	1211	1211	1211	1211	0.00	3941.3
KroA200	29368	29368	29368	29368	0.00	7261.4

Tableau.III.4 : Les résultats de la deuxième stratégie de notre modèle.

Problèmes	BKS	Meilleur Solution	Mauvaise Solution	La solution moyenne	ER (%)	Temps (seconde)
Eil51	426	469.2153	510.9889	490.489322	19.950446	1328.1721
Berlin52	7542	7873.516	8822.5663	8377.08604	11.0724747	1326.5001
St70	675	771.0458	824.6336	796.38676	17.9832237	2375.2033
Eil76	538	602.722	648.0988	628.444	16.8111524	2874.6097
Rat99	1211	1432.2743	1516.9172	1471.30228	21.4948208	5137.1251
KroA200	29368	36467.4911	40799.032	39357.2393	34.0140263	2.65E+04

Comparant les résultats des deux expériences, l'approche de l'expérience de la deuxième stratégie donne des résultats adéquats a la BKS avec une erreur relative nulle.

Notre approche affiche une meilleure solution supérieure à la BKS avec une erreur relative fortement supérieure à la norme.

Nous remarquons aussi que le temps d'exécution ; qui représente le temps moyen nécessaire pour résoudre le problème dans un essai ; est trop élevé dans notre approche par rapport à l'approche de la deuxième stratégie.

En conclusion, la proche de la deuxième stratégie est la meilleur en matière de meilleure solution, mauvaise solution, erreur relative et temps.

III.3. Discussion globale

A la fin d'étude, nous disons que le travail présenté dans l'article (sur lequel on a travaillé) est meilleur en terme de résultats obtenus dans le cadre de ce que nous avons accompli ci-dessus, a va savoir : valeur des résultats affichés par le programme, erreur et temps.

Et par conséquence on propose de changer l'algorithme APE par un autre pour espérer une amélioration dans les futurs travaux de fin d'étude de recherche.

Conclusion

Nous avons d'abord présenté l'outil de programmation pour cette étude (MATLAB R2017a). Ensuite, nous avons présenté des stratégies appliquées au modèle ACO plus PSO plus APE.

Au finale, nous testons l'application sur notre problème avec des différentes bases de test (villes) pour comparer les résultats obtenus avec les résultats des stratégies de l'article pour définir le meilleur modèle.

Le résultat tiré à la fin de ce chapitre confirme que la combinaison ACO plus PSO plus 3-OPT est la meilleur selon MAHI & AL.

Jouant sur la troisième combinaison en la remplacent par un autre algorithme d'amélioration citant comme exemple GPE, qu'elle sera la meilleure solution... Celle d'ACO plus PSO plus 3-OPT ou bien celle de la nouvelle combinaison ACO plus PSO plus GPE ?!

Conclusion Générale

Conclusion générale

Les problèmes d'optimisation combinatoire sont des problématiques de choix d'une meilleure solution qui s'appelle : la solution optimale. Parmi ces problèmes, on cite : le problème de coloration, d'affectation, de sac-à-dos, d'ordonnancement, le problème de voyageur de commerce... etc.

Pour résoudre tous ces problèmes, il existe deux méthodes de résolution : les méthodes exactes et les méthodes approchées.

Les algorithmes de colonies de fourmis sont une de ces approches utilisées pour traiter ce genre de problème. Ils sont des algorithmes inspirés du comportement des fourmis ou d'autres espèces formant un super organisme et qui constituent une famille de méta-heuristiques d'optimisation.

Cet algorithme est basé sur l'algorithme Ant System, qu'il a été créé au départ pour la résolution du problème de voyageur de commerce.

Dans notre mémoire, nous avons pris le chemin emprunté par MAHI & AL en modifiant l'algorithme utilisé par un autre, où nous avons utilisé deux stratégies différentes.

Dans un premier temps, nous avons appliqué L'algorithme APE uniquement pour la meilleure solution. Dans un deuxième temps, nous avons appliqué le programme sur tous les résultats obtenus et à la fin d'étude, nous avons comparé les résultats obtenus avec les résultats précédents pour choisir la meilleure application.

De là, on peut dire que le modèle présenté à travers cette étude ne donne pas de bons résultats par rapport à son prédécesseur.

Dans le futur et comme perspective, il sera intéressant d'ajouter des nouvelles combinaisons pour trouver des résultats plus efficaces que le modèle ACO plus PSO plus 3-OPT.

Dans notre étude, on a testé la combinaison selon deux stratégie et toujours la combinaison ACO plus PSO plus 3-OPT reste la meilleure, mais ça n'empêche de tester d'autres algorithmes pour valider l'efficacité du modèle président. Ceci ouvre d'autres problématiques qui doivent être études, à titre d'exemple si on remplace 3-OPT par un autre algorithme que doit être la meilleure combinaison ?!

Bibliographie

[1]. V. Th. Paschos, Optimisation combinatoire, concepts fondamentaux, hermès science publication, pp. 17-60, lavoisier, 2005.

[2]. Melle Labed Kaouther, Expérimentation des algorithmes génétiques multiobjectif dans un processus décisionnel multicritère en aménagement du territoire, université d'Oran Essenia.

[3]. Y. Collette, P. Siarry, Optimisation multiobjectif, éditions eyrolles, paris, 2002.

[4]. C.H.Papadimitrion and K. Steiglitz, Combinatorial optimization: algorithms and complesity, prentce-hall, Inc, upper saddle river, nj, usa, 1982.

[5]. Vincent Barichard, Approches hybrides pour les problemes multiobjectifs: application à Angers, Thèse de doctorat Informatique, 24 Novembre 2003, 162P.

[6]. Michel Nakhla, Jean-Claude Moisdon, Recherche opérationnelle méthodes d'optimisation en gestion, transvalor presses des mines, 2010.

[7]. D. E. Knuth, The art of computer programming, Addison-Wesley, Reading, MA, 1st edition, 1973.

[8]. Albin. Morelle, Introduction à l'algorithmique, esiee, paris, mars 2010.

[9]. GREGOIRE ALLAIRE livre - Analyse numérique et optimisation - Une introduction a la modélisation mathématique et a la simulation numérique french, octobre 2005.

[10]. S. Bourazza, Variantes d'algorithmes génétiques appliquées aux problèmes d'ordonnancement, thèse de doctorat, université du Havre, france, 2006.

[11]. A Hertz, « Tabu search for large scale timetabling problems », European journal of operational research, vol 54, page 39, 1991.

[12]. Fischetti (M.) et Toth (P.), An additive approach for the optimal solution of the prize collecting traveling salesman problem. In: Vehicle routing: methods and studies, _ed. par Golden (B.) et A. (A.). pp. 319 343. {Elsevier, Amsterdam}.

[13]. Y. Harrath, Contribution à l'ordonnancement conjoint de la production et de la maintenance: application au cas d'un job shop, thèse de doctorat à L'UFR des sciences et techniques de l'université de franche-comté, besançon, 2003.

[14]. Amira Gherboudj, Méthodes de résolution de problèmes difficiles académiques, université de 2, thèse doctorat d'informatique, 2013, P216.

[15]. Sébastien.Noël, Métaheuristiques hybrides pour la résolution du problème d'ordonnancement de voitures dans une chaîne d'assemblage automobile, a université du Québec à Montréal, magister informatique, Octobre 2007, 127p.

[16]. Labed.Said, Méthodes bio-inpirées hybrides pour la résolution de problèmes complexes, université constantine 2, thèse de doctorat en Sciences en Informatique, 2013, p135.

[17]. Dreo.Johann, Pétrowski A, Siarry P. & Eric T, Métheuristiques pour l'optimisation difficile, editions eyrolles, 2003.

[18]. Ouadfel.S, Contributions à la segmentation d'images basées sur la résolution collective par colonies de fourmis artificielles, thèse de doctorat de l'université de Batna, discipline : Informatique, 2006.

[19]. L.Slimani, Contribution à l'application de l'optimisation par des méthodes métaheuristiques à l'écoulement de puissance optimal dans un environnement de l'électricité dérégulé : université de Batna, thèse de doctorat: Electrotechnique, 2009, 186p.

[20]. Hölldobler.B and Wilson.E, The ants springer verlag, Berlin, Germany, 1990.

[21]. Bonabeau.E and Theraulaz.G, Intelligence collective, Hermes, 1994.

[22]. Mostapha Redouane. Khouadjia, Résolution de problemes d'optimisation combinatoire par systèmes artificiels auto-organises : université mentouri de constantine, magister en informatique, 2008, p87.

[23]. Mathurin.Soh, Baudoin Nguimeya.Tsofack, Clémentin Tayou.Djamegni: Approche heuristique multi colonie des fourmis pour la résolution du problème de voyageur de commerce, Proceedings of CARI 2020, October 2020.

[24]. Lec-27 Heuristics for TSP (Contd), Prof.G. Srinivan, voir le lien: https://www.youtube.com/watch?v=hjZDDz3r1es

[25]. Hertono. G.F, Ubadah, Handari. B.D, The modification of hybrid method of ant colony optimization, particle swarm optimization and 3-OPT algorithm in traveling salesman problem, Journal of Physics: Conf. Series 974 2018.

[26]. Cleve Model, MATLAB, récupéré à avril 2019, voir le lien : https://fr.wikipedia.org/wiki/MATLAB.

Résumé

Dans ce mémoire, on a proposé un algorithme hybride (ACO plus APE) pour résoudre le problème de voyageur de commerce et on a comparé les résultats avec le modèle de MAHI & AL pour tirer le meilleur modèle au coût de temps, résultats et erreur.

Mots clés : OCF, APE, PVC, Algorithme hybride.

ملخص

في هذه المذكرة، اقترحنا نموذجا هجينا (خوارزمية مستعمرة النمل إضافة إلى خوارزمية التبادل الزوجي المجاور) لحل مشكلة البائع المتجول وقارنّا النتائج المتحصّل عليها مع نتائج نموذج ماهي وآل، للحصول على أفضل نموذج بتكلفة أقل تمس الوقت والنتائج والخطأ.

الكلمات المفتاحية: خوارزمية مستعمرة النمل، خوارزمية التبادل الزوجي المجاور، مشكلة البائع المتجول، الخوارزميات الهجينة.

Abstract

In this thesis, we proposed a hybrid algorithm (ACO plus APE) to solve the traveling salesman problem and we compared the results with the model of MAHI & AL to get the best model at the cost of time, results and error.

Key words: ACO, APE, TSP, hybrid Algorithm.