République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed El Bachir El Ibrahimi - Bordj Bou Arreridj- Faculté des Mathématiques et d'informatique

Département mathématique

MEMOIRE

Présenté en vue de l'obtention du diplôme

Master en mathématique

Spécialité : **Recherche opérationnelle**

THEME

Un algorithme hybride (Aco-2opt) pour la résolution du problème de voyageur de commerce

Présenté par :

BAYOU Abdelwahab

BENSEFIA Amir

Devant le jury composé de :

Président : S.MAACHE

Examinateur : H.TOUATI

Encadreur Dr : SAHA Adel MCB à L'U. El Bachir El Ibrahimi- BBA

Promotion : 2020/2021

Remerciements

Je tiens à remercier DR. SAHA Adel pour son encadrement, son soutien, sa compréhension et la ferme conviction dont il a fait preuve dans ce projet et les membres du jury. Je tiens également à remercier ma famille, qui a été une inspiration unique pour ce projet.

Abstract

Today, nature-inspired hybrid methods are widely used to solve optimization problems, due to their ability to provide innovative solutions to complex problems and become more efficient if we use hybrid methods. We therefore proposed a new model to solve the traveling salesman problem, we applied the model to real problems and then we compared the results with the results of another model.

Keywords: ant colony optimization, 2-opt, hybrid methods, traveling salesman problem

Résume

Aujourd'hui, les méthodes hybrides inspirées de la nature sont largement utilisées pour résoudre des problèmes d'optimisation, en raison de leur capacité à apporter des solutions innovantes à des problèmes complexes et deviennent plus efficaces si nous utilisons des méthodes hybrides. Nous avons donc proposé un nouveau modèle pour résoudre le problème du voyageur de commerce, nous avons appliqué le modèle à des problèmes réels puis nous avons comparé les résultats avec les résultats d'un autre modèle.

Mots-clés : optimisation par colonies de fourmis, 2-opt, méthodes hybrides, problème du voyageur de commerce

الملخص

في يومنا هذا، تستخدم طرق التهجين المستوحاة من الطبيعة على نطاق واسع لحل مشكلات التحسين، ذلك لقدرتها على تقديم حلول مبتكرة للمشكلات المعقدة وتصبح أكثر فاعلية إذا استخدمنا طرق هجينة. لذلك اقترحنا نموذجا جديدا لحل مشكلة بائع متجول، قمنا بتطبيق النموذج على مشاكل حقيقية ثم قمنا بمقارنة النتائج بنتائج نموذج اخر.

الكلمات المفتاحية: تحسين مستعمرة النمل، الطرق الهجينة، مشكلة بائع متجول

Table des matières

Introduction Générale

Dans de nombreux domaines d'activités humaines (industrie, commerce, économie) les dirigeants d'entreprises sont souvent confrontés à de multitudes de choix lors de la prise de décision concernant une action donnée (organisation d'une production, réseau du transport,...etc.).

Ils sont alors amenés à prendre des décisions qui répondent le mieux à leurs intérêts et ce en appliquant des méthodes qui optimisent des critères qu'ils ont choisis auparavant comme exemple la maximisation d'un profit ou la minimisation d'une dépense .

La Recherche Opérationnelle (RO), est la discipline des outils et méthodes scientifiques utilisables pour élaborer de meilleures décisions. C'est un ensemble de méthodes et techniques visant à résoudre des problèmes d'optimisation (programmes mathématiques minimisant ou maximisant un ou plusieurs critères en respectant certaines conditions dites « contraintes ») modélisant des problèmes réels dans différents domaines (économie, finance, gestion, transport, logistique, communication, etc.)[1].

L'optimisation est un outil pertinent en sciences appliquées et pour l'analyse des systèmes physiques et mathématiques. Pour utiliser cet outil, on doit passer par l'identification des objectifs. Cet objectif peut être le profit, le temps, l'´énergie potentielle, ou n'importe quelle quantité ou combinaison de qualité qui peut être représentée par une valeur algébrique. L'objectif dépend de quelques caractéristiques du système, appelées variables ou inconnus. Notre but est de déterminer les valeurs des variables qui optimisent l'objectif. Après l'identification des objectifs des variables pour les problèmes donnés (modélisation) et obtention d'un modèle formulé, un algorithme d'optimisation peut être utilisé pour la résolution du problème.

Les problèmes d'optimisation consister par exemple à chercher le maximum du rendement d'une unité de production industrielle, la distance maximale parcourue par un véhicule pour une quantité de carburant donnée, le nombre maximal de clients servis à un guichet dans un intervalle de temps fixé, la résistance mécanique maximale d'une pièce produite, etc. Un problème d'optimisation peut également consister à chercher le minimum des coûts de production d'une usine, à minimiser les pertes thermiques d'un procédé industriel, à identifier le temps de fabrication minimal, etc. Dans tous ces exemples, il est nécessaire d'avoir modélisé (c'est à dire mis en équations) la quantité qu'on cherche à optimiser (maximiser ou minimiser). En fait, on retrouve des problèmes d'optimisation dans de nombreuses applications telles que les problèmes de conception, d'ordonnancement, de planification de production, de localisation, et de transport.

Parmi les problèmes mentionnés ci-dessus, les problèmes de transport occupé une place très importante dans la vie économique de notre société. Ce problème est l'un des sujets les plus importants de la recherche opérationnelle car il consiste à étudier les méthodes pour optimaliser le transport des produits entre des sources et leurs destinations afin de maximiser les bénéfices et minimiser les pertes, tels que le problème du voyageur de commerce (PVC).

La résolution des problèmes d'optimisation est devenue un sujet central en recherche opérationnelle, le nombre de problèmes d'aide à la décision pouvant être formalisés sous la forme d'un problème d'optimisation étant en forte croissance. Il existe plusieurs méthodes

pour résoudre ces problèmes. Il y a des méthodes exactes qui fournissent une solution optimale mais dans un temps pouvant être trop long, d'autre part il y a des méthodes approchée qui fournissent rapidement de bonnes solutions, sans pouvoir prouver l'optimalité.

La bio-inspiration (inspiration biologique) est une branche des sciences de l'ingénieur qui consiste à s'inspirer de la nature pour améliorer ou développer des produits respectueux de l'environnement tout en étant très performante. Les applications de l'ingénierie bio-inspirée se retrouvent aujourd'hui dans un grand nombre des domaines industriels tell que aéronautique, matériaux, informatique, robotique, intelligence artificielle, l'énergie, économie, etc.

Une évolution croissante du nombre d'études menées sur les animaux vivants en groupe ou en société et plus particulièrement les insectes sociaux. Ces études dans la théorie de l'auto-organisation ont inspiré un grand nombre des chercheurs pour développer une nouvelle approche appelé les métaheuristiques. Ces derniers sont des algorithmes généraux d'optimisation applicables à une grande variété des problèmes. Elles sont apparues dans le but de résoudre au mieux des problèmes d'optimisation, elles sont généralement inspirées de la nature : l'éthologie (colonie des abeilles, communauté des loups, colonie des fourmis, etc.).

Il existe des algorithmes qui permettent de résoudre les problèmes d'optimisation appelé les algorithmes d'hybridation, L'utilisation des méthodes hybrides permettent de combiner les avantages des deux types des méthodes (exacte et approchée). De sorte que l'avantage des méthodes exacte est la précision de donner la solution optimale et l'avantage des méthodes approchée est de donner une solution dans un temps minimal.

Parmi les méthodes développées pour résoudre le problème de voyageur de commerce (PVC) est la méthode hybride proposée par Mahi et al [2], qui basée sur l'optimisation des colonies des fourmis ou (ant colony optimization (ACO)), et algorithme de 2- OPT.

Dans ce mémoire, nous étudions un nouveau modèle d'optimisation (ACO, 2-OPT) et voir s'il est meilleur que l'ancien modèle (ACO, 3-OPT) ou non.

Nous souhaitons implémenter un nouveau modèle d'hybridation (ACO, 2-OPT) pour résoudre un problème du voyageur de commerce (PVC) et comparer ces résultats avec celles trouver dans l'ancien modèle (ACO, 3-OPT), et déterminer le meilleur modèle.

Notre mémoire est organisé comme suit

Le chapitre I : Dresse un état de l'art des méthodes d'optimisation en accordant une importance particulière aux méthodes utilisées dans nos travaux de recherche. Il aborde aussi les définitions générales des méthodes d'optimisation qui se divisent en deux volets exacte et approchée.et nous donnons aussi brièvement les formulations des problèmes traités dans ce mémoire

Le chapitre II : Il présente de manière détaillée les métaheuristiques que nous avons étudiées dans le cadre de cette thèse et est consacré à la description des méthodes hybrides proposées pour résoudre des problèmes du voyageur de commerce (PVC).

Le chapitre III : contient les outils et les langages que nous allons utiliser pour résoudre notre problème (PVC) et discuter les résultats obtenus.

Finalement, on conclue notre mémoire avec une conclusion générale

CHAPITRE I

L’optimisation combinatoire

I.1-Introduction

Dans ce chapitre, nous introduisons les bases et quelques concepts sur l'optimisation, l'optimisation harmonique, et quelques problèmes d'optimisation et les moyens de les résoudre.

I.2-Optimisation

L'optimisation est un processus qui cherche à analyser et résoudre des problèmes analytiques ou numériques qui consistent à déterminer le meilleur élément de l'ensemble, c'est-à-dire un certain critère quantitatif. Il existe plusieurs types d'optimisation, par exemple l'optimisation combinatoire qui fait l'objet de notre recherche.

I.2.1-Optimisation combinatoire

L'optimisation combinatoire occupe une place très importante dans la recherche opérationnelle, les mathématiques discrètes et l'informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par les nombreuses applications pratiques qui peuvent être formulées sous la forme d'un problème d'optimisation combinatoire. Bien que les problèmes d'optimisation globale soient faciles à identifier, ils sont généralement difficiles à résoudre. En fait, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles, et il n'existe donc actuellement aucune solution algorithmique valide pour toutes les données. [3]

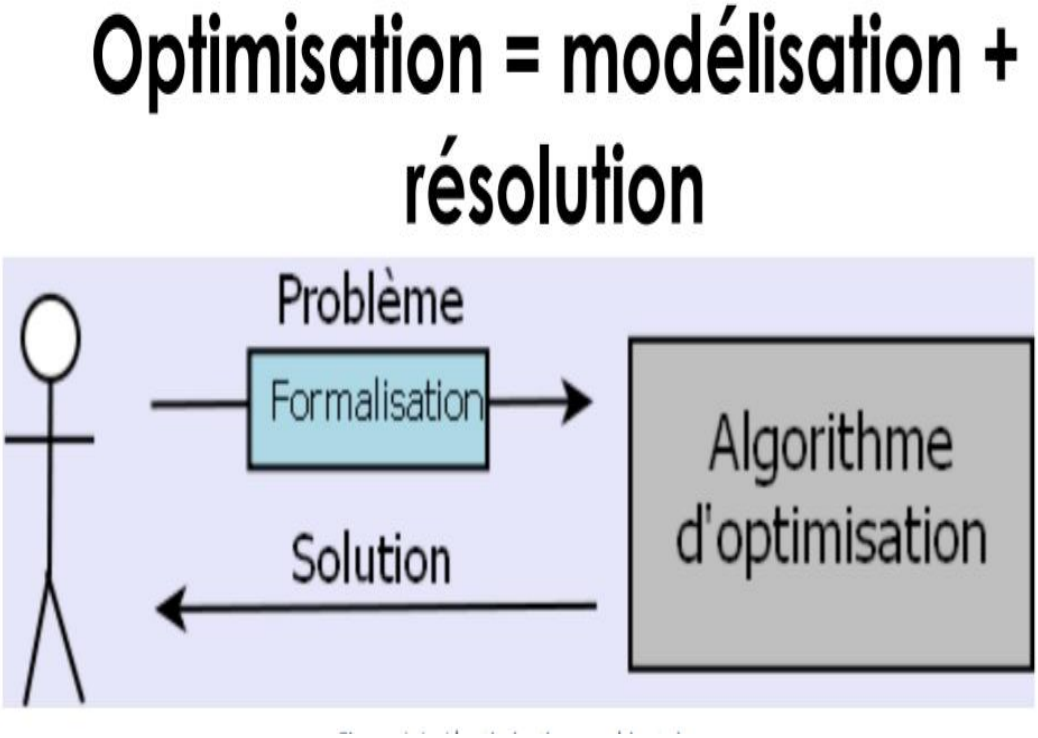

Figure I.1- L’optimisation combinatoire

I.2.2-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 :

$$(PO) = \begin{cases} minimiser\ f(x) \\ g_i(x) \leq 0 \quad i = 1 \ldots . n\ (contraintes\ d'inégalités) \ \ldots\ldots\ldots . (\mathrm{I}.1) \\ h_j(x) = 0 \quad i = 1 \ldots . n\ (contraintes\ d'égalités) \ \ldots\ldots\ldots\ldots . . (\mathrm{I}.2) \end{cases} \quad [4]$$

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

$$\max_{s \in S} \mathbf{f(s)} \cong \min_{s \in S} (-\mathbf{f(s)}) \quad \ldots\ldots\ldots\ldots\ldots \textbf{(I.3)} \qquad [4]$$

Généralement, une solution s ϵ S est un vecteur d'un espace à n dimensions.

Définition 2 : problème d'optimisation continue

Dans le cas de variables réelles, on a :

$$\mathbf{S \subseteq R^n} \ldots\ldots\ldots\ldots\ldots\ldots \textbf{(I.4)} \quad [4]$$

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

$$\mathbf{(PO)} \ \min_{s \in R^N} \mathbf{f(s)} \quad \ldots\ldots\ldots\ldots\ldots \textbf{(I.5)} \ [4]$$

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 :

$$\mathbf{(POC)} \ \min_{s \epsilon Z^N} \mathbf{f(s)} \ldots\ldots\ldots\ldots \textbf{(I.6)} \quad [4]$$

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

$$\mathbf{S \subseteq Z^n} \ldots\ldots\ldots\ldots \textbf{(I.7)} \quad [4]$$

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 ϵ 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. [4]

I.3-Exemple de problèmes d'Optimisation Combinatoire

Comme exemple de problème d'optimisation combinatoire on peut citer le problème du sac à dos, le problème d'affectation, la planification, le VRP (véhicule routing problem) problème de tournées de véhicules et le problème de voyageur de commerce TSP (Traveling salesman problem) qui est l'exemple le plus connu de problèmes d'optimisation combinatoires.

I.3.1-Le problème du voyageur de commerce

Le problème du voyageur de commerce (TSP) a été étudié au 18 ème siècle par un mathématicien d'Irlande nommé William Rowam Hamilton et par le mathématicien britannique nommé Thomas Penyngton Kirkman. Discussion détaillée sur le travail de Hamilton et Kirkman peut être vu à partir du livre intitulé Graph Theory [5]. C'est un problème relevant de la classe NP, il est même NP complet. Sous sa forme la plus classique, son énoncé est le suivant " Un voyageur de commerce doit visiter une et une seule fois un nombre fini de villes et revenir à son point d'origine. Trouvez l'ordre de visite des villes qui minimise la distance totale parcourue par le voyageur"[6] (voir figure **I.2**).

Soit n villes et C_{ij} le coût (ou la distance) correspondant au trajet i – j. Le problème consiste à déterminer un tour ou circuit hamiltonien, c'est-à-dire ne passant qu'une et une seule fois par les n villes, et qui soit de coût minimum. Soit la variable X_{ij} qui vaut 1 si le tour contient le trajet i – j, et 0 autrement. Le problème s'écrit [4] :

$$\begin{cases} \text{Min } Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\ \sum_{j=1}^{n} x_{ij} = 1 & \forall i \quad (I.8) \\ \sum_{i=1}^{n} x_{ij} = 1 & \forall j \quad (I.9) \\ \sum_{i\epsilon Q}\sum_{j\epsilon Q} x_{ij} \geq 1 & \forall Q \quad (I.10) \\ x_{ij} \in \{0,1\} & \forall i, \forall j \end{cases}$$

Où Q représente un sous-ensemble de {1,…, n}. [4]

Les deux premières contraintes traduisent le fait que chaque ville doit être visitée exactement une fois ; la troisième contrainte interdit les solutions composées de sous-tours disjoints, elle est généralement appelée contrainte d'élimination des sous-tours [7]

Figure I.2- le problème de voyageur de commerce (PVC) [8]

I.4-Les méthodes de résolution de problème d'optimisation

Elles sont schématisées comme suit :

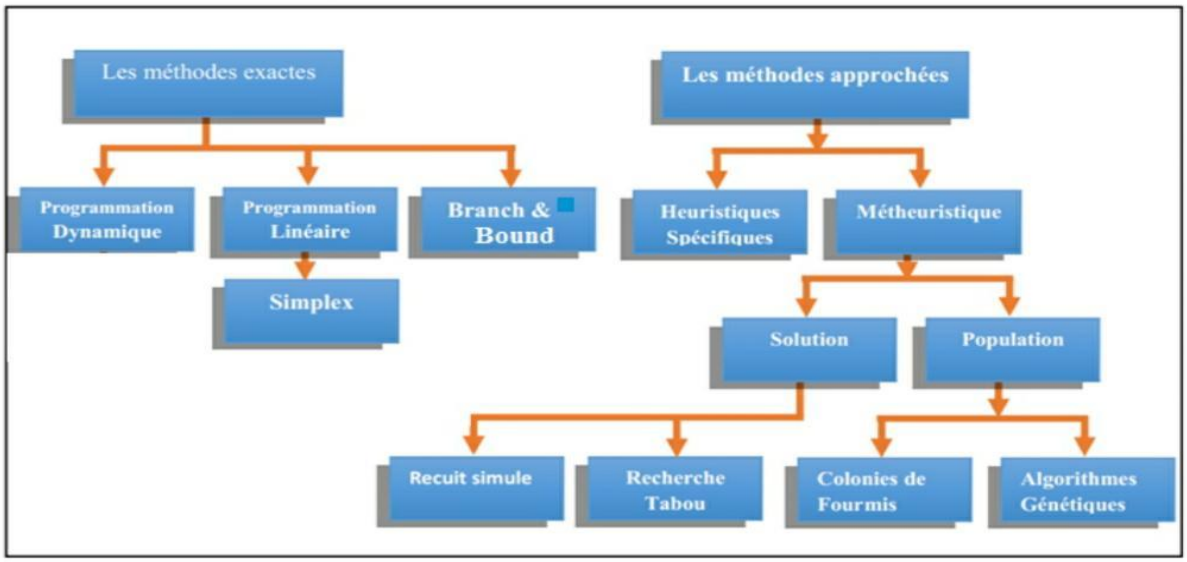

Figure I.3- classification des méthodes de résolution des problèmes d'optimisation [9]

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 difficile à résoudre. Les méthodes approchées, encore appelées heuristiques, permettent quant à elles d'obtenir une solution approchée rapidement, mais qui n'est donc pas toujours optimale. [10]

Pour les problèmes ayant un ensemble fini de solutions admissibles, un simple algorithme exact consiste simplement à énumérer toutes les solutions de l'ensemble des solutions possibles. Un tel algorithme n'est pas pratique dans le cas des problèmes d'optimisation combinatoire en raison de la grande taille de l'ensemble des solutions admissibles.

Pour augmenter leur efficacité, toutes les méthodes exactes modernes utilisent des règles d'élimination de parties de l'espace de recherche dans lesquelles la solution (optimale) ne peut pas exister.

Ces approches font une énumération implicite de l'espace de recherche. Les méthodes exactes les plus connues sont Branch and Bound, Branch and cut, Branch and Bound and Cut, Programmation Dynamique développée par Richard Bellman. [6]

Les méthodes (ou algorithmes) exactes garantissent de trouver la solution optimale d'un problème d'optimisation combinatoire donné mais le temps nécessaire pour son obtention augmente de manière exponentielle en augmentant la taille du problème. [6]

En 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. [6]

En optimisation combinatoire, une heuristique est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale. L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte. Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre sans offrir aucune garantie quant à la qualité de la solution calculée. Les heuristiques peuvent être classées en deux catégories :

– Méthodes constructives qui génèrent des solutions à partir d'une solution initiale en essayant d'en ajouter petit à petit des éléments jusqu'à ce qu'une solution complète soit obtenue [11].

–Méthodes d'amélioration qui démarrent avec une solution initialement complète, et de manière répétitive essaie d'améliorer cette solution comme c'est le cas de 2-opt.

Les "métaheuristiques" d'optimisation sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Elles sont apparues à partir des années 80, dans le but de résoudre au mieux des problèmes d'optimisation. Dans la pratique, trois types de problèmes d'optimisation sont souvent rencontrés : les problèmes combinatoires (discrets), les problèmes continus (à variables continues), et les problèmes mixtes [12].

L'exemple le plus répandu en optimisation combinatoire est celui du voyageur de commerce. En optimisation continue, un exemple simple est celui de la recherche des paramètres d'un modèle numérique pour approcher au mieux des données réelles [13]. Le dernier type est celui des problèmes mixtes, qui comportent à la fois des variables discrètes et des variables continues. Les métaheuristiques s'efforcent de résoudre tout type de problème d'optimisation. Elles sont caractérisées par leur caractère stochastique, ainsi que par leur origine discrète. Elles sont inspirées par des analogies avec la physique (recuit simulé, recuit micro canonique), avec la biologie (algorithmes évolutionnaires) ou encore l'éthologie (colonies de fourmis, essaim particuliers). .

Cependant, elles ont l’inconvénient d’avoir plusieurs paramètres à régler. Il est à souligner que les métaheuristiques se prêtent à toutes sortes d’extensions, notamment en optimisation mono objectif et multi objectif. De nos jours les gestionnaires et les décideurs sont confrontés quotidiennement à des problèmes de complexité grandissante, qui surgissent dans des secteurs très divers. Le problème à résoudre peut souvent s’exprimer sous la forme générale d’un problème d’optimisation, dans lequel on définit une ou plusieurs fonctions objectif que l’on cherche à minimiser ou à maximiser par rapport à tous les paramètres concernés [14]. La résolution d’un tel problème a conduit les chercheurs à proposer des méthodes de plus en plus performantes, parmi lesquelles on trouve les métaheuristiques qui sont des méthodes générales de recherche dédiées aux problèmes d’optimisation difficile. [15]

Les métaheuristiques hybrides sont apparues en même temps que le paradigme lui-même, mais la plupart des chercheurs n'y accordaient que peu d'intérêt (Cotta, Talbi et al) [16]. Elles gagnent maintenant en popularité, car les meilleurs résultats trouvés pour plusieurs POC ont étés obtenus avec des algorithmes hybrides. Les méthodes hybrides peuvent être divisées en deux groupes : les métaheuristiques hybrides qui impliquent une combinaison de plusieurs métaheuristiques et les méthodes hybrides impliquant une combinaison d'une méthode exacte et d'une méthaheuristique. [17]

I.5-Conclusion

Dans ce chapitre, nous avons vu la définition de l'optimisation combinatoire avec quelques exemples, en particulier le problème du voyageur de commerce et quelques méthodes de résolution.

CHAPITRE II

(Aco-2opt) pour la résolution du problème de voyageur de commerce

II.1-Introduction

Dans ce chapitre on va définir les métaheuristiques (ACO) et les heuristiques (2-opt) Pour résoudre le problème du voyageur de commerce.

II.2-Optimisation des colonies de fourmis (ACO)

Les algorithmes basés sur l'optimisation des colonies de fourmis (ACO) sont destinés à l'étude des systèmes informatiques dans lesquels le calcul est effectué par des fourmis artificielles pour imiter le comportement des fourmis normales. Comme le montre la **Figure II.1**, **Figure II.2**, leur principe fondamental repose sur la manière dont les fourmis cherchent leur nourriture et retournent au nid. Les vraies fourmis sont utilisées pour trouver une source de nourriture sans aucune information visuelle et cette capacité des vraies fourmis est également utilisée pour trouver le chemin le plus court entre les nids et la source de nourriture. Au départ, les fourmis recherchent la nourriture de manière aléatoire et explorent les environs du nid. Chaque fois qu'une fourmi trouve une source de nourriture, elle évalue la quantité et la qualité de la nourriture et apporte la nourriture nécessaire au nid. Pendant le voyage de retour, la fourmi laisse une substance chimique connue sous le nom de traînée de phéromone sur le sol. Les autres fourmis suivent cette piste de phéromones pour trouver une source de nourriture pour elles. Après un certain temps, l'intensité des phéromones augmentera sur un chemin particulier et toutes les fourmis commenceront à suivre ce chemin uniquement. Par conséquent, les fourmis exploitent toutes ces pistes de phéromones comme moyen de trouver leur chemin de la source à la destination et vice-versa. [18]

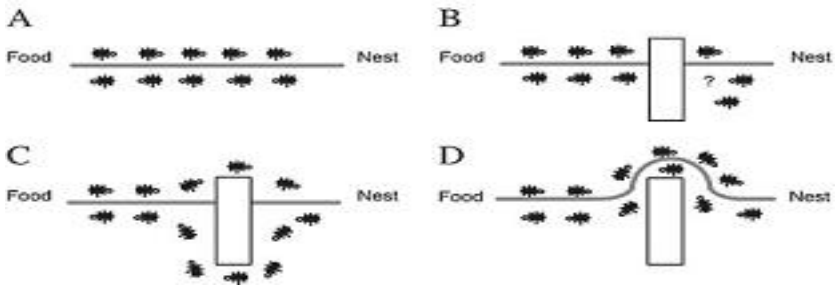

Figure II.1-colonie de fourmis

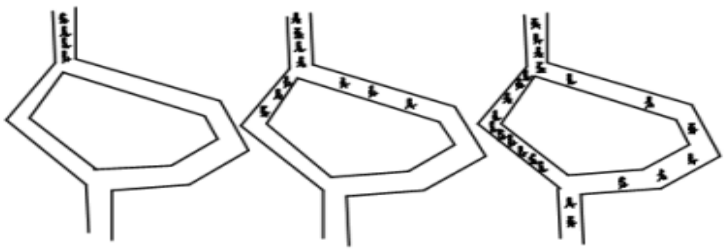

Figure II.2- Comportement social d'une vraie fourmi

Comme le montre la **Figure II.3**, ces comportements biologiques des fourmis sont convertis en un cadre de calcul pour de nombreux problèmes de nature combinatoire. [18]

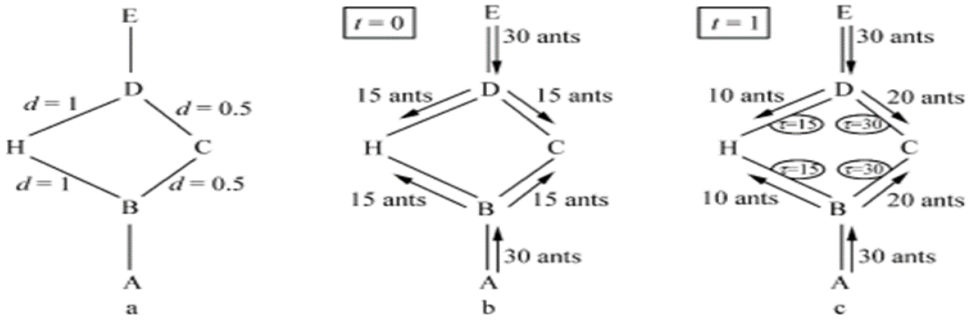

Figure II.3- Comportement informatique de la fourmi artificielle

L'optimisation des colonies de fourmis est une méthaheuristique pour les problèmes d'optimisation combinatoire difficiles modélisés d'après la communication des fourmis trouvant les chemins les plus courts vers les sources de nourriture. Le premier algorithme ACO était Ant System (AS), introduit par Dorigo en 1992. Il l'a ensuite généralisé dans la méthaheuristique ACO. [18]

Une méthaheuristique est un ensemble de concepts algorithmiques qui peuvent être utilisés pour définir l'application de méthodes heuristiques à un large ensemble de problèmes différents. L'utilisation de la méthaheuristique a augmenté la capacité à trouver une solution de très haute qualité à des problèmes d'optimisation combinatoire difficiles et pertinents dans la pratique dans un délai raisonnable. La méthaheuristique ACO a été proposée comme cadre commun pour l'application existante et les variantes algorithmiques de variétés d'algorithmes de fourmi. C'est une méthaheuristique où une colonie de fourmis artificielles coopère pour trouver de bonnes solutions à des problèmes d'optimisation discrets difficiles. [18]

En cartographiant ce comportement de fourmis biologiques avec de vraies fourmis, un concept méthaheuristique connu sous le nom d'optimisation des colonies de fourmis (ACO) est introduit par Marco Dorigo, qui est discuté dans la section suivante. L'algorithme de fourmis étudie des modèles dérivés de l'observation du comportement de fourmis réelles et utilise ces modèles comme source d'inspiration pour la conception et la mise en œuvre de nouveaux algorithmes pour la résolution de problèmes d'optimisation et de contrôle distribué. [18]

II.2.1-Système de fourmis

Ant System (système de fourmis) sera par la suite appelé AS. Les variables que nous devons définir pour comprendre la suite sont les suivantes [Dorigo 02] :

– $b_i(t)$ ou $i \in X$ le nombre de fourmis dans la ville i à l'instant t, (X : l'ensemble de ville).

– $m = \sum_{i \in X} b_i$ leur nombre total, invariant dans le temps

– $\tau_{ij}(t)$ la valeur de $\tau_{ij}(t)$ à l'instant t ($\tau_{ij}(t)$: quantité de phéromone)

– n = |X|, le nombre de villes

– $\eta_{ij} = \frac{1}{d_{ij}}$, la visibilité d'une ville j quand on est placé sur la ville i, invariante dans le temps (d_{ij} : la distance entre la ville i et la ville j).

Précisons maintenant le comportement de l'ensemble de la colonie. A tout instant t, chaque fourmi choisit une ville de destination selon un choix défini. Toutes les fourmis se placent à l'instant (t+1) dans une ville de leur choix. On appelle une itération de l'algorithme AS, l'ensemble de déplacements de l'ensemble de la colonie entre l'instant t et l'instant(t +1). Ainsi après n itérations, l'ensemble de la colonie aura effectué un circuit hamiltonien sur le graphe. De cette manière toutes les fourmis commenceront et finiront leur tour en même temps. [19]

II.2.1.1- Choix d'implémentation

On précise que chaque fourmi a une mémoire implémentée par une liste de villes déjà visitées. Cela permet de garantir qu'aucune fourmi ne visitera deux fois une même ville au cour de sa recherche.

La mémoire de chaque fourmi est vidée lorsqu'elles ont terminé leur cycle. [19]

Choix des transitions [Dorigo 02] :

Une fourmi k placée sur la ville i à l'instant t va choisir sa ville j de destination en fonction de la visibilité η_{ij} de cette ville et de la quantité de phéromones τ_{ij} (t) déposée sur l'arc reliant ces deux villes. Ce choix sera réalisé de manière aléatoire, avec une probabilité de choisir la ville j donnée par : [18]

$$P_{ij}{}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}.\eta_{ij}^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}.\eta_{il}^{\beta}} & \text{si } j \in N_i^k \\ 0 \quad \text{autrement} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots \text{(II.1)}$$

Où l'on définit l'ensemble N_i^kcomme étant l'ensemble des villes que la fourmi k, placée sur la ville i, n'a pas encore visité à l'instant ''t'' dans le cycle courant. α et β sont deux paramètres qui contrôlent l'importance relative entre phéromones et visibilité. Ainsi si α est égal à 0, le choix se fera uniquement en fonction de la visibilité (si β est différent de 0). [19]

Mise à jour des phéromones [Dorigo 02]

A la fin de chaque cycle (chaque fourmi a parcouru les n sommets qui composent le graphe), les variables des phéromones sont mises à jour selon la formule : [19]

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \ldots\ldots\ldots\ldots \text{(II.2)}$$

Où ρ ∈ [0, 1[est un coefficient qui définira la vitesse d'évaporation des phéromones sur les arcs entre l'instant t et l'instant (t + n), et où $\Delta\tau_{ij}(t)$ représente la quantité de phéromone déposée par les fourmis dans ce même intervalle de temps sur l'arc (i, j). Le choix de ρ est important, en effet si ρ se rapproche trop de 1, on observe un effet de stagnation des

phéromones sur les arcs, ce qui implique des inconvénients tel que le fait de voir les mauvaises solutions persister. De même, choisir ρ ≈ 0 implique une évaporation trop rapide des phéromones, donc amène la fourmi à un choix dépendant uniquement de la visibilité des nœuds. [19]

Quantités de phéromones déposées [Dorigo 02]

Appelons$T_k(t)$ = (u_{k_1} , . . . , u_{k_q}) le tour réalisé par la k-ème fourmi dans l'intervalle de temps [t, t+n], et $L_k(t)$ sa longueur. $T_k(t)$ (et donc $L_k(t)$) s'obtient en analysant la mémoire de la fourmi.

Soit $\Delta\tau_{ij}^k$ (t), la quantité de phéromones déposée par cette fourmi sur l'arc (i, j) dans ce même intervalle de temps. On le définit ainsi : [18]

$$\Delta\tau_{ij}{}^{\mathrm{k}} = \begin{cases} \frac{\mathrm{Q}}{L_k(t)} & \text{"si } (\mathrm{u} \in T_k(\mathrm{t}) \wedge \mathrm{u} = (\mathrm{i},\mathrm{j}))" \\ 0 & \text{autrement} \end{cases} \quad \ldots\ldots \text{(II.3)}$$

Où Q est une constante. On voit bien ici que les phéromones sont régulées en fonction de la qualité de la solution obtenue car plus L_k(t) est faible plus l'arc sera mis à jour en phéromones. On peut maintenant définir le $\Delta\tau_{ij}$(t) de la formule de mise à jour des phéromones ainsi : [19]

$$\Delta\tau_{ij}\ (\mathrm{t}) = \sum_{\mathrm{k}=1}^{\mathrm{m}} \Delta\tau_{ij}^k\ (\mathrm{t}) \ldots\ldots\ldots\ldots \text{(II.4)}$$

II.3-Algorithme de 2-opt

Nous présentons un algorithme de recherche locale parallèle évolutif basé sur le parallélisme des données. Le concept de structures de voisinage distribuées est introduit et appliqué au problème du voyageur de commerce (TSP). L'algorithme de recherche locale parallèle que nous utilisons trouve les mêmes solutions de qualité que l'algorithme classique à 2-opt et à une bonne accélération. [20]

La technique 2-OPT a été proposée pour la première fois par Croes en 1958. Dans son article, il définit la technique comme une solution optimisée pour le problème du voyageur de commerce pour les instances symétriques et asymétriques (cette dernière nécessitant plus de travail). Il n'y a aucune garantie que cette technique trouvera une réponse optimale globale, à la place, la réponse renvoyée est généralement dite 2-optimale, ce qui en fait une heuristique . En fait, il améliore progressivement une réponse réalisable, initialement donnée (recherche locale) jusqu'à ce qu'elle atteigne un optimum local et qu'aucune autre amélioration ne puisse être apportée. Les améliorations se font à l'aide de ce qu'il appelle des « 'inversions' ». [21]

2-Opt est probablement l'heuristique de recherche locale la plus basique et la plus largement utilisée pour le TSP. Cette heuristique obtient des résultats étonnamment bons sur des instances euclidiennes « du monde réel » à la fois en ce qui concerne le temps d'exécution et le rapport d'approximation. Il existe de nombreuses études expérimentales sur les performances de 2-Opt. Cependant, les connaissances théoriques sur cette heuristique sont

encore très limitées. Même son pire temps d'exécution sur les instances euclidiennes n'était pas connu jusqu'à présent. [22]

II.3.1-Principe de fonctionnement

On peut donner une description plus formelle de l'heuristique. Soit un graphe G=(V,E) et H un cycle hamiltonien dans G muni d'une fonction coût renvoyant la somme des poids des arêtes composant le cycle On définit une 2-opt dans le cycle hamiltonien H comme le remplacement de deux arêtes a_1,a_2 ∈ H par deux arêtes a_3,a_4 ∈ H tel que le tour résultant est toujours hamiltonien dans G.(voir **figure II**.4)

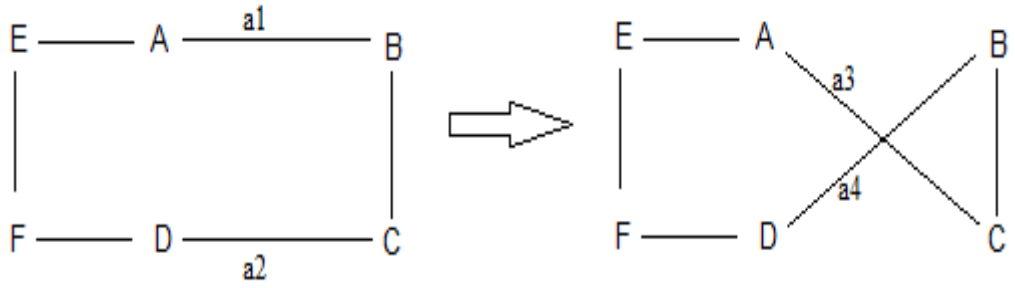

Figure II.4-Exemple 1 de 2-opt

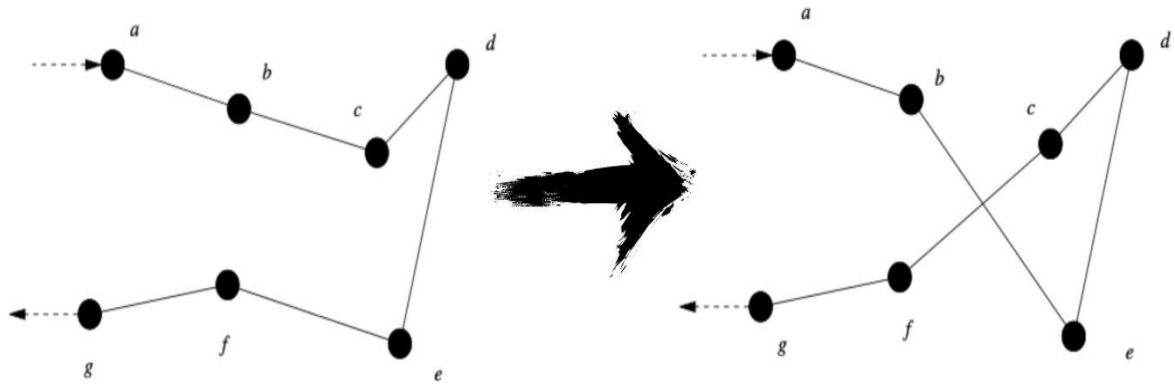

Figure II.5-Exemple 2 de 2-opt

2-opt est un algorithme itératif : à chaque étape, on supprime deux arêtes de la solution courante et on reconnecte les deux tours ainsi formés. Cette méthode permet, entre autres, d'améliorer le coût des solutions en supprimant les arêtes sécantes lorsque l'inégalité triangulaire est respectée (voir figure II.5). Sur le schéma de droite, la route <A; B; C; D; E; F; G> est changée en <A; B; E; D; C; F; G> en inversant l'ordre de visite des villes e et c. Plus généralement, lorsqu'on inverse l'ordre de parcours de deux villes, il faut aussi inverser l'ordre de parcours de toutes les villes entre ces deux villes. [23]

II.3.2-Exemple générale

Voice une matrice de distance de 5 villes

	1	2	3	4	5
1	0	9	17	13	7
2	9	0	19	8	3
3	17	19	0	7	14
4	13	8	7	0	9
5	7	3	14	9	0

Tab II.2-matrice de distance

Et voici une solution à ce problème avec une taille de L=49, (figure **II**.6)

Quand on applique 2-opt à ce cycle hamiltonien on va trouver une solution meilleure que L=49, figure (**II**.7)

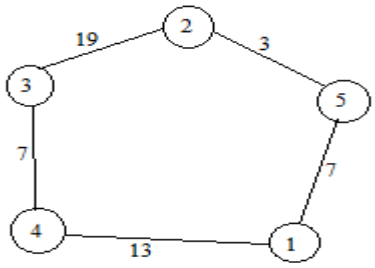

Figure II.6-tour de notre problème

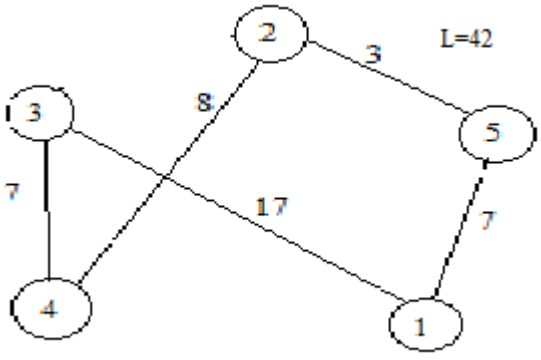

Figure II.7-tour a 2-opt de notre problème

II.4-Conclusion

Dans ce chapitre on a vu les méthodes principales pour résoudre notre problème de voyageur du commerce, chaque méthode est basée sur une idée différente, mais elles partagent le même handicap si l'espace de recherche est très grand la solution trouvée peut être loin de la solution optimale, donc il est intéressent de proposer une approche coopérative et parallèle utilise les points fortes d'ensemble des méthodes avec un parallélisme pour améliorer les résultat donnes. .

CHAPITRE III

Implémentation et résultats

III.1-Introduction

Dans ce chapitre on faire une résolution a quelques exemples de problèmes de PVC avec une méthode hybride basée sur l'algorithme de fourmis et l'algorithme de 2-opt et faire une discussion pour les résultats obtenu.

III.2-Langage de Matlab

Matlab est un logiciel de calcul numérique produit par Math Works (voir le site web http : //www.mathworks.com/).Il est disponible sur plusieurs plateformes. Matlab est un langage simple et très efficace, optimisé pour le traitement des matrices, d'où son nom. Le nom dérive de cette représentation : MATLAB = MATrix LABoratory. MATLAB n'est pas le seul environnement de calcul scientifique existant car il existe d'autres concurrents dont les plus importants sont : MAPLE et MATHEMATICA. Il existe même des logiciels libres qui sont des clones de MATLAB comme SCILAB et OCTAVE. Pour le calcul numérique, Matlab est beaucoup plus concis que les vieux langages (C, Pascal, Fortran, Basic). Un exemple : plus besoin de programmer des boucles modifier pour un à un les éléments d'une matrice. On peut traiter la matrice comme une simple variable. Matlab contient également une interface graphique puissante, ainsi qu'une grande variété d'algorithmes scientifiques. On peut enrichir Matlab en ajoutant des boîtes à outils (toolbox) qui sont des ensembles de fonctions supplémentaires, profilées pour des applications particulières. [24]

III.2.1-Interface Matlab

Au lancement de Matlab, l'interface suivante apparait :

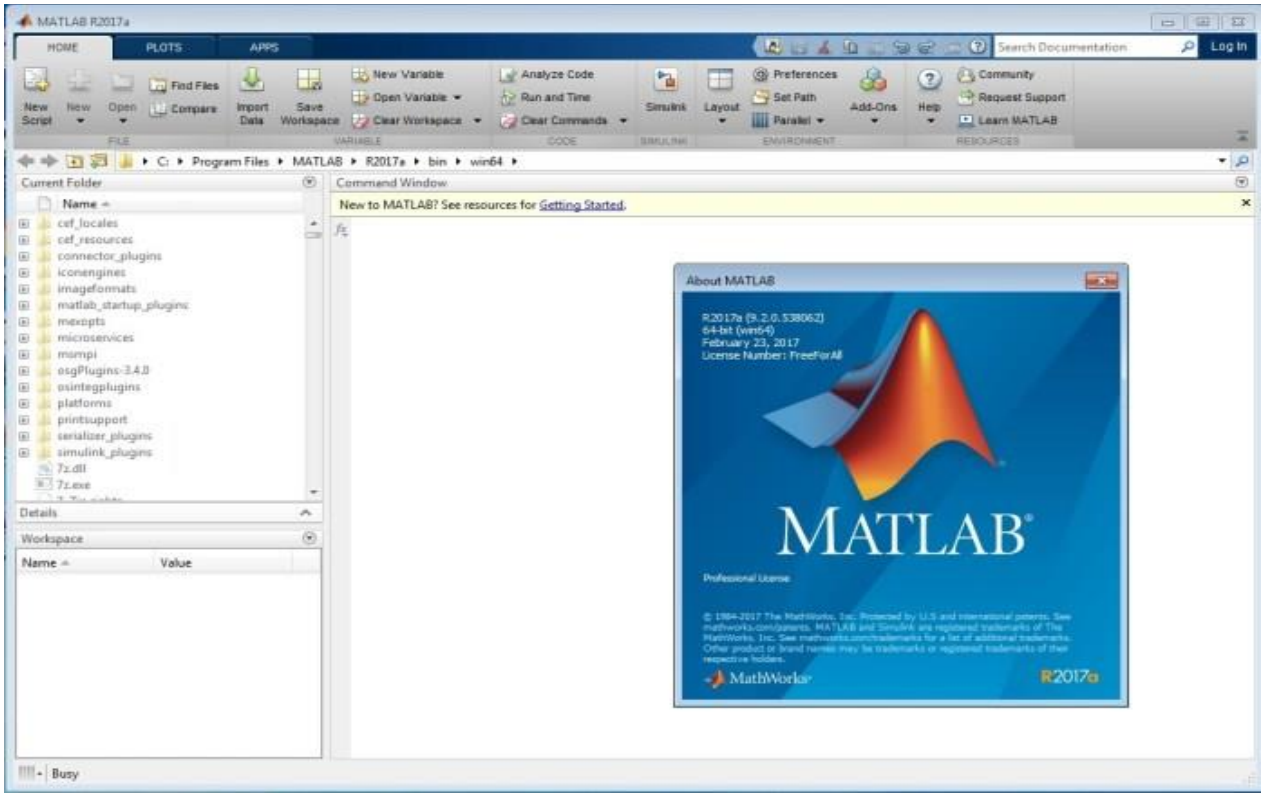

Figure III.1-L'interface de Matlab

- Command window (console d'exécution) : à l'invite de commande « >> », l'utilisateur peut entrer les instructions à exécuter. Il s'agit de la fenêtre principale de l'interface. [25]

- Current directory (répertoire courant) : permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Les programmes de l'utilisateur doivent être situés dans ce répertoire pour être visible et donc exécutable. [25]

- Workspace (espace de travail) : permet de visualiser les variables définies, leur type, la taille occupée en mémoire… [25]

- Command history : historique des commandes que l'utilisateur a exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande. [25]

III.3-Objectif

On va utiliser Matlab pour résoudre notre problème PVC. L'implémentation utilise 6 problèmes de benchmarks de TSPLIB (Reinelt, 1991): Eil51 (51-city problem (Christofides/Eilon)), Berlin52 (52 locations in Berlin (Groetschel)), St70 (70-city problem (Smith/Thompson)), Eil76 (76-city problem (Christofides/Eilon)), Rat99 (Rattled grid (Pulleyblank)) et kroA200 (200-city problem A (Krolak/Felts/Nelson)). Chaque problème est testé en 10 exécutions avec 50 itérations. La solution moyenne (MS) est calculée à l'aide de l'équation suivante [2] :

$$MS = \frac{\sum_{i=1}^{n} Di}{n} \quad \text{(III.1)}$$

Où Di représenté les solutions de chaque exécution, n nombre d'exécution

Le pourcentage d'erreur relative (ER) est utilisé pour déterminer la qualité de la méthode de résolution du TSP et calculé par l'équation [2] :

$$ER = \frac{MS-BKS}{BKS} * 100 \quad \text{(III.2)}$$

Où BKS est la solution la plus connue. Dans la mise en œuvre, nous utilisons également la valeur des paramètres suggérée dans les articles précédents, comme suit : ρ = 0,1, (Mahi)[26] et Q = 100 comme la meilleure valeur constante pour la méthode ACO (Dorigo, 1996). Nous utilisons également $0 \leq \alpha \leq 2$ et $0 \leq \beta \leq 2$ comme suggéré par Mahi [26] pour obtenir le meilleur résultat pour la méthode ACO. Mahi [26] suggère également d'utiliser 10 agents lorsque nous utilisons la méthode hybride utilisant l'ACO et l'algorithme 2-Opt [2].

III.4-Les paramètres utilisés dans l'algorithme d'ACO

Nombre d'itérations	50
Nombre des fourmis	10
Q	100
α, β	∈ [0,2]
ρ	0.1

Tab III.1-parametre utilisé dans ACO

III.5-Principe de l'algorithme hybride

La méthode hybride commence par la mise en œuvre des étapes ACO par les étapes suivants :
• le processus initial de distribution de l'agent.
• l'initialisation de la phéromone et le calcul de la valeur heuristique.
• calculera une probabilité en utilisant les valeurs des paramètres de 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 terminaison, le processus sera poursuivi en redistribuant les agents aux sommets.
• L'ensemble du processus sera répété jusqu'à ce que les critères de terminaison 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 2-Opt pour obtenir la meilleure solution.

III.5.1- 1er Model hybride proposé

Dans Mahi [26], l'algorithme 3-Opt est appliqué à la solution candidate. Dans le 1er model hybride proposé, l'algorithme 2-Opt est appliqué à la solution réalisable, comme décrit dans la **Figure III.1**. Les différences peuvent être observées après que chaque agent a visité chaque sommet (visitation terminée). La solution réalisable sera choisie parmi toutes les solutions générées par tous les agents. L'algorithme 3-Opt sera appliqué à ces solutions réalisables. L'étape suivante est le troisième processus ACO qui met à jour la phéromone et suivi des mêmes étapes que Mahi 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. [2]

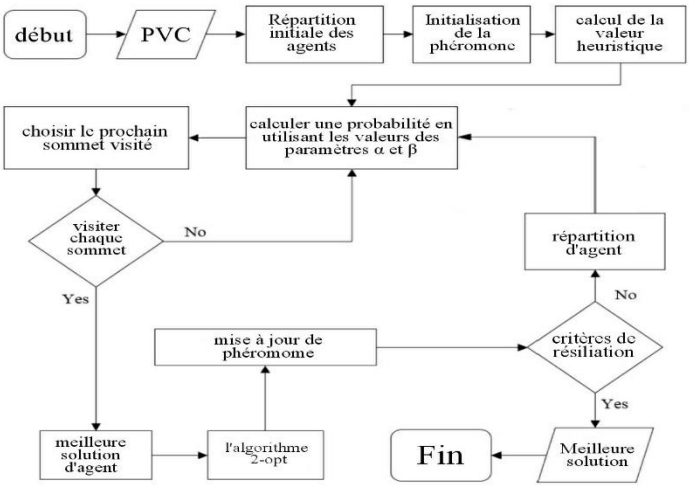

Figure III.1 – Le 1er Model hybride proposé

Problem	BKS	Best sol	Wrost Sol	Mean Sol	ER(%)	TIME
Eil51	426	435.4657	489.3243	456.5230	7.16	115.37
Berlin52	7542	7544.3659	8351.0331	7994.42858	5.99	139.58
St70	675	686.6472	732.5344	715.85685	6.05	161.34
Eil76	538	565.064	601.0226	579.79426	7.76	181.48
Rat99	1211	1282.0711	1350.158	1315.3566	8.61	204.54
KroA200	29368	30507.1658	33054.2573	31862.9376	8.49	234.02

Tab III.2- les résultats de 1er Model hybride proposé

Discussion de tab III.2

Le tableau **Tab III.2** montre que le 1er model hybride proposé ne donne pas de bons résultats pour tous les problèmes. Il montre que la meilleure solution correspondante est plus grande que la meilleure solution connue (BKS). De plus, le ‘’ER’’ correspondant est également supérieur à 5.99 %. Les résultats montrent également un temps de fonctionnement élevé

Et voici les résultats obtenu graphiquement :

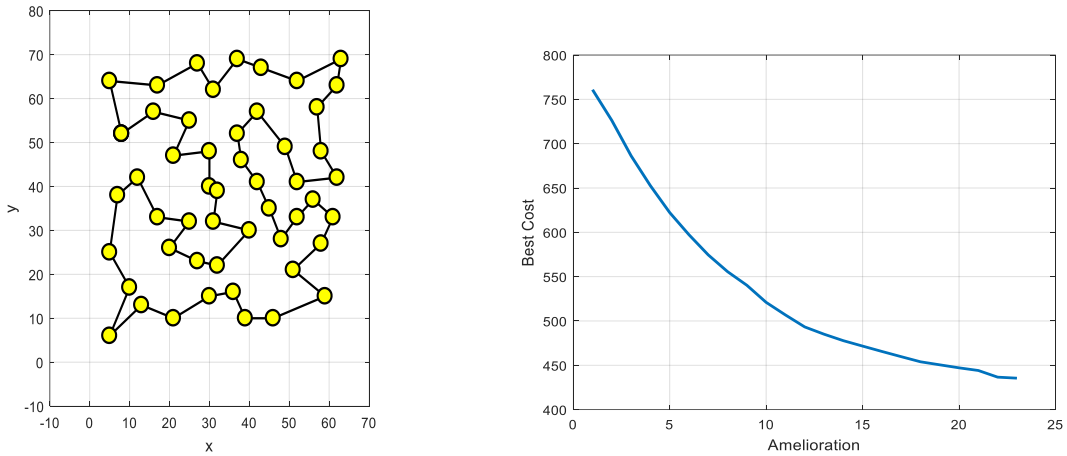

Figure III.2-la meilleure solution de problème (Eil51)

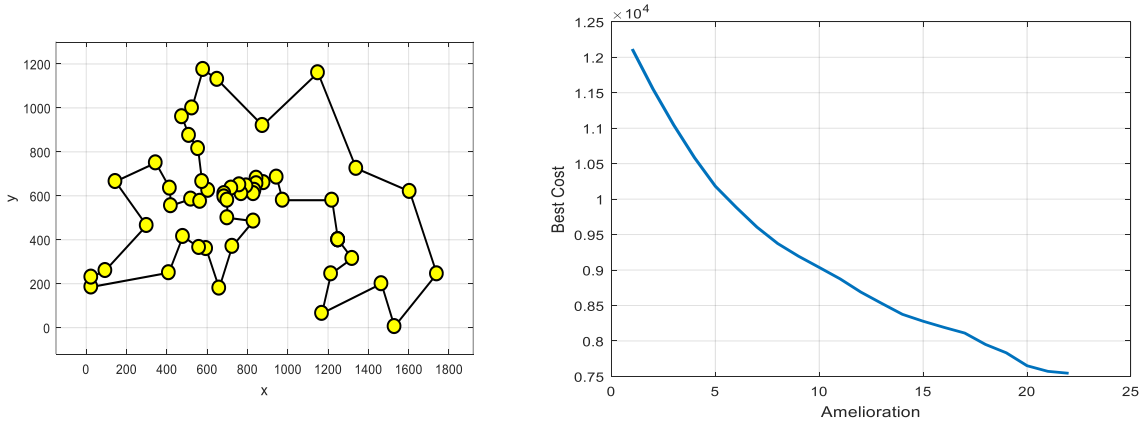

Figure III.3-la meilleure solution de problème (Berlin52)

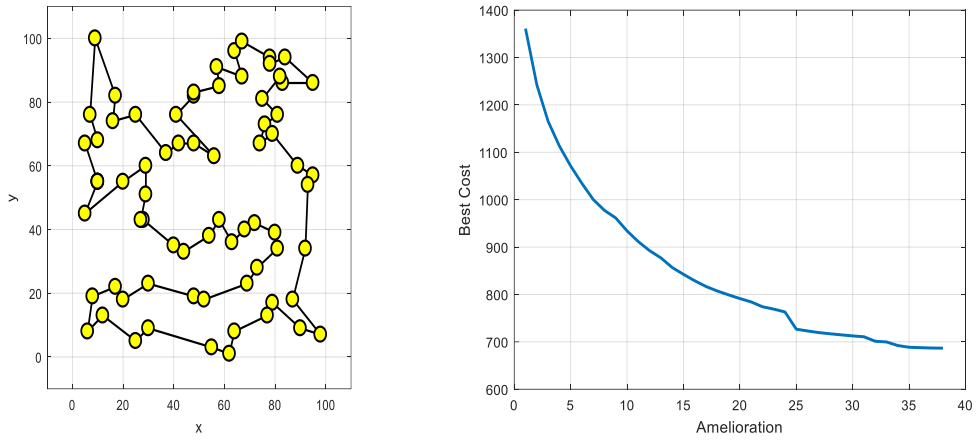

Figure III.4-la meilleure solution de problème (St70)

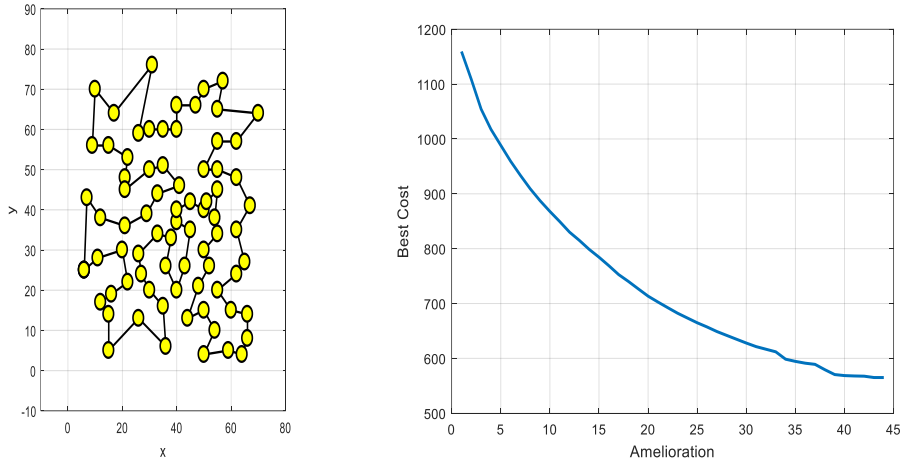

Figure III.5-la meilleure solution de problème (Eil76)

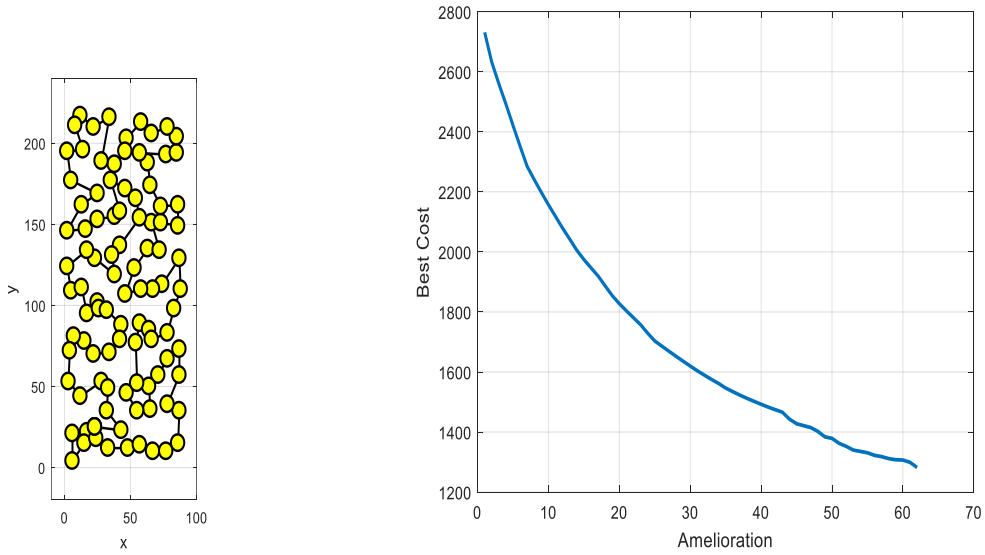

Figure III.6-la meilleure solution de problème (Rat99)

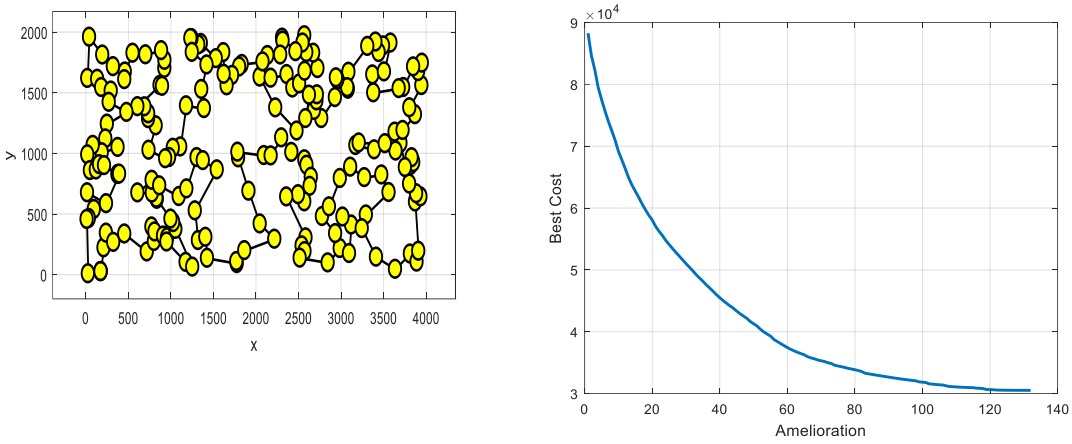

Figure III.7-la meilleure solution de problème (KroA200)

	Problem	BKS	Best sol	Wrost Sol	Mean Sol	ER(%)	Time (sec)
Aco 2opt	Eil51	426	435.4657	489.3243	456.5230	7.16	115.37
	Berlin52	7542	7544.3659	8351.0331	7994.42858	5.99	139.58
	St70	675	686.6472	732.5344	715.85685	6.05	161.34
	Eil76	538	565.064	601.0226	579.79426	7.76	181.48
	Rat99	1211	1282.0711	1350.158	1315.3566	8.61	204.54
	KroA200	29368	30507.1658	33054.2573	31862.9376	8.49	234.02
Aco 3opt	Eil51	426	437	514	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

Tab III.3- la comparaison entre le modèle original et le 1er Model hybride proposé

Discussion de tab III.3

Le tableau **Tab III.3** montre clairement que 2-opt est meilleur que 3-opt dans la première méthode de modification car le ‘’ER’’ de 2-opt est inférieur à celui de 3-opt et le temps d’exécution elle est plus élevé dans 3-opt pour les 6 problèmes.

III.5.2- 2éme model hybride proposé

Dans la 2éme model hybride proposé, nous implémentons l'algorithme 2-Opt à la solution de tous les agents à chaque itération, comme le montre la **Figure III.8**.Dans la première méthode de modification, l'algorithme 2-Opt n'est appliqué qu'à la solution réalisable dans chaque itération ACO, tandis que dans la deuxième méthode de modification, l'algorithme 2-Opt est appliqué à la solution de tous les agents à chaque itération. Une fois que toutes les solutions ont été optimisées

par 2-Opt, la solution avec la distance totale minimale est sélectionnée comme solution réalisable. La prochaine étape est la troisième étape de l'ACO (mise à jour de la phéromone) et des critères de test. Le processus se poursuivra jusqu'à ce que le processus satisfasse aux critères terminaux et que la meilleure solution soit obtenue.

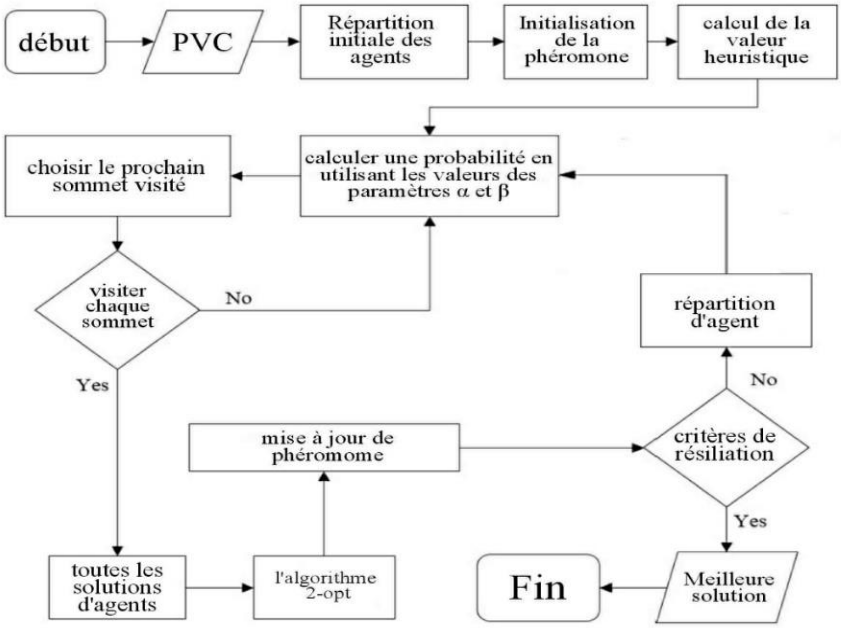

Figure III.8-la 2éme model hybride proposé

Problem	BKS	Best sol	Wrost Sol	Mean Sol	ER(%)	TIME
Eil51	426	430.4455	433.4536	431.87891	1.38	137.07
Berlin52	7542	7544.3659	7544.3659	7544.3659	0.03	161.97
St70	675	677.1096	680.4998	678.16207	0.46	195.37
Eil76	538	551.5871	562.1849	557.35369	3.59	235.91
Rat99	1211	1223.1084	1253.4284	1242.01794	2.56	294.13
KroA200	29368	30026.264	30373.5552	30204.7971	2.84	478.19

Tab III.4- les résultats du 2éme model hybride proposé

Discussion de tab III.4

La 2éme model hybride proposé donne de meilleurs résultats pour chaque problème utilisé. Le tableau **Tab III.4** montre que les pires solutions approchent le BKS de chaque problème et que le ‘’ER’’ approche à 0. Cependant, le temps d'exécution représentant le temps moyen requis pour résoudre le problème dans l'expérience est supérieur à n'importe lequel des problèmes correspondants dans le premier model hybride proposé. Parce qu'il implémente un grand nombre d'implémentations de l'algorithme à deux choix dans la méthode. L'algorithme 2-Opt fonctionne avec chaque solution générée par chaque facteur à chaque itération. Comme il y a 10 agents (fourmis) et que chacun s'exécute en 50 itérations, il existe 500 implémentations de l'algorithme 2-Opt.

Et voici les résultats trouve dans **tab III.4** graphiquement :

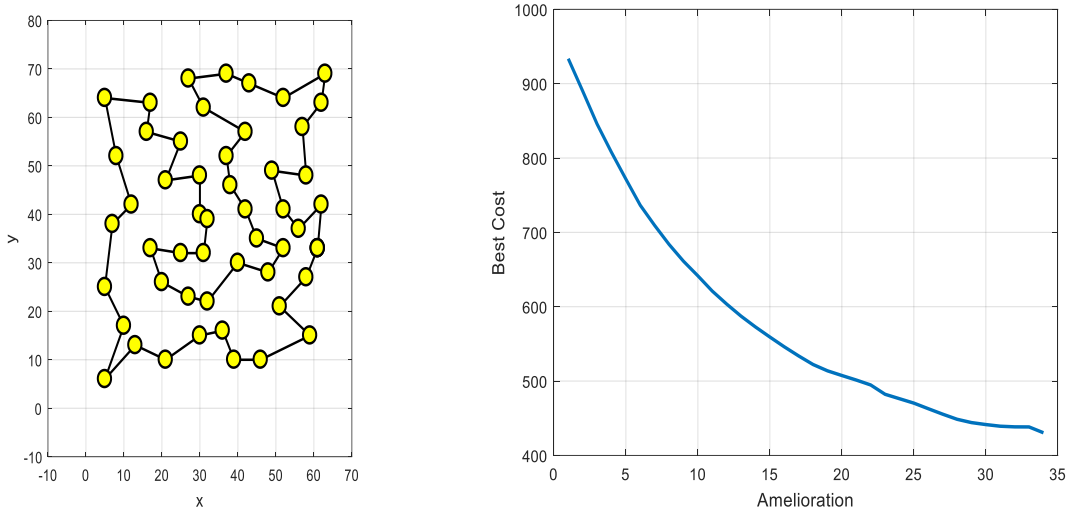

Figure III.9-la meilleure solution de problème (Eil51)

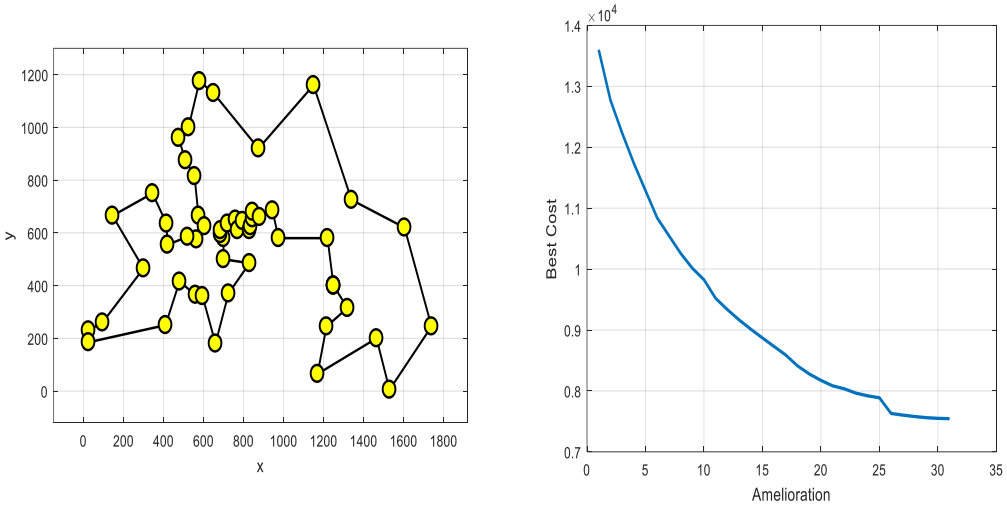

Figure III.10-la meilleure solution de problème (Berlin52)

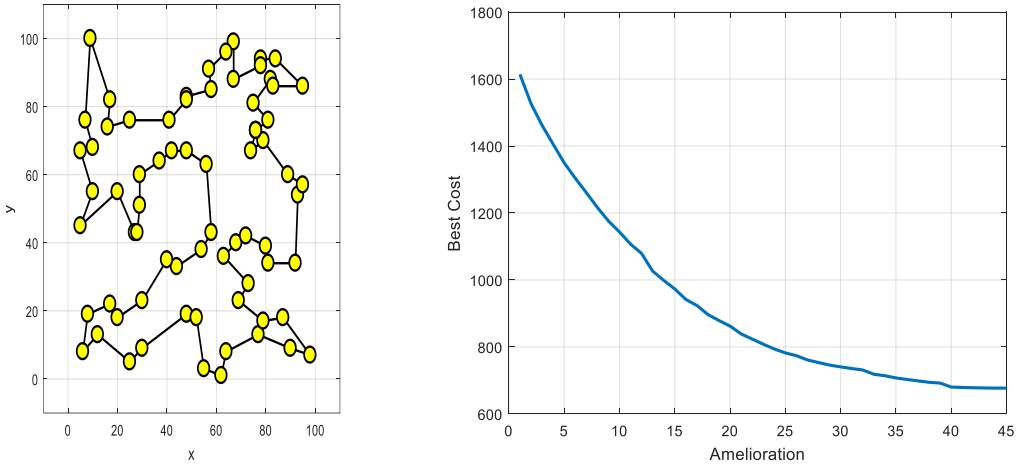

Figure III.11-la meilleure solution de problème (St70)

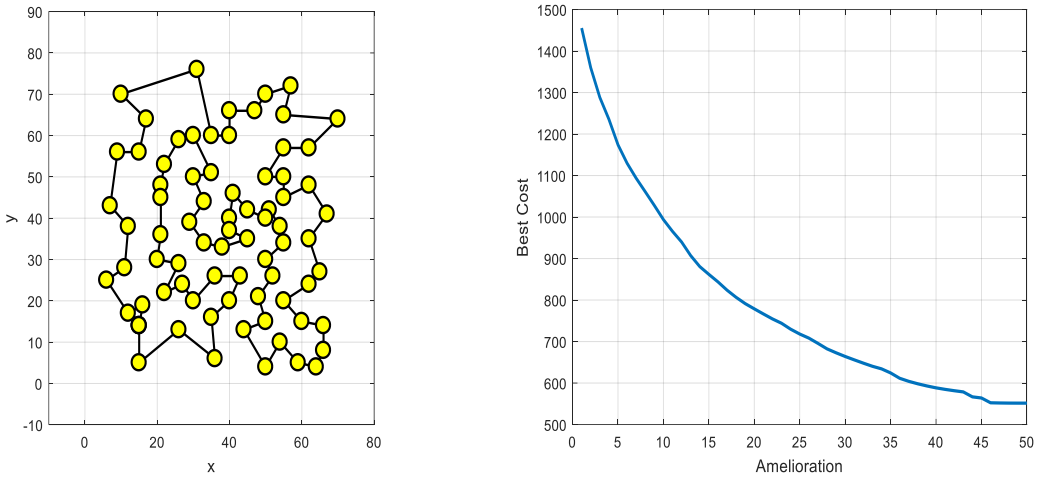

Figure III.12-la meilleure solution de problème (Eil76)

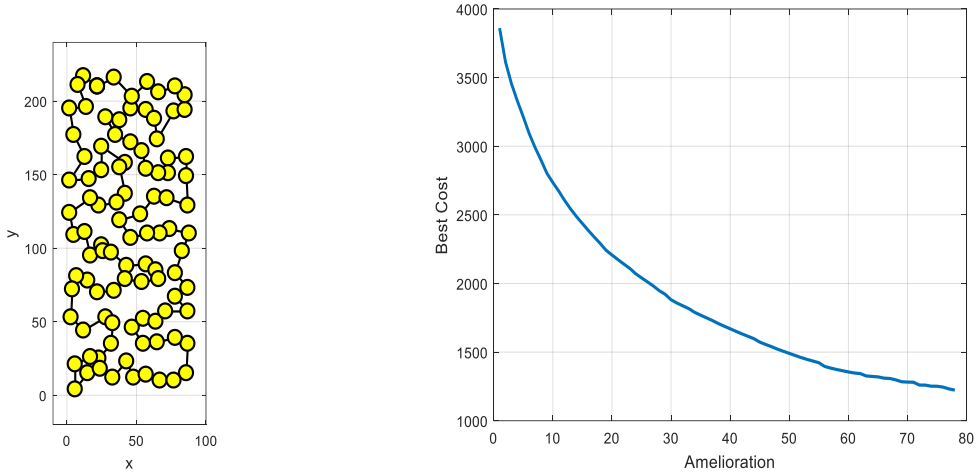

Figure III.13-la meilleure solution de problème (Rat99)

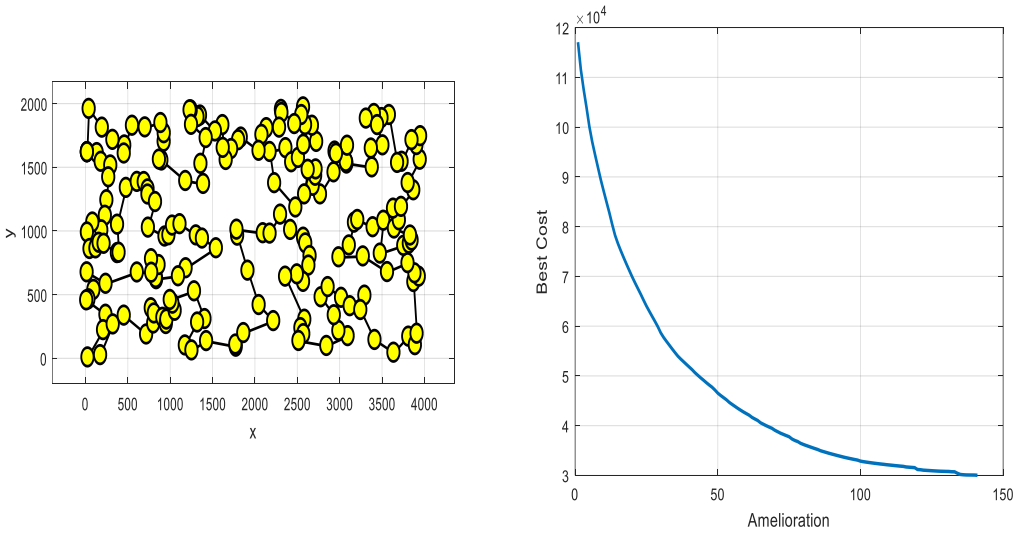

Figure III.14-la meilleure solution de problème (KroA200)

	Problem	BKS	Best sol	Wrost Sol	Mean Sol	ER(%)	TIME(S)
Aco-2opt	Eil51	426	430.4455	433.4536	431.87891	1.38	137.07
	Berlin52	7542	7544.3659	7544.3659	7544.3659	0.03	161.97
	St70	675	677.1096	680.4998	678.16207	0.46	195.37
	Eil76	538	551.5871	562.1849	557.35369	3.59	235.91
	Rat99	1211	1223.1084	1253.4284	1242.0179	2.56	294.13
	KroA200	29368	30026.264	30373.555	30204.797	2.84	478.19
Aco-3opt	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

Tab III.5-la comparaison entre 2éme model hybride proposé et model original

Discussion de tab III.5

Le tableau **Tab III.5** montre clairement que 3-opt est meilleur que 2-opt dans la 2éme méthode de modification car le ER de 3-opt est inférieur (0%) à celui de 2-opt pour les six problèmes, mais quand observé le temps d'exécution on va remarquer que 2opt est meilleure avec un ER(%) très inférieur.

Dans le tableau **Tab III.6**, nous essayons d'utiliser plus d'itérations avec deux problèmes TSPLIB, Ei51 (problème de 51 sommets) et KroA200 (problème de 200 sommets), sur les deux méthodes hybrides Avec 5 exécutions, et 50, 100 itérations à chacune.

	Problème	**Itération**	**ER(%**		**TIME (sec)**	
			1	**2**	**1**	**2**
2opt	**Eil51**	50	6.40	1.33	105.72	130.22
		100	5.71	1.14	136.89	213.91
	kroA200	50	8.92	2.84	225.24	406.08
		100	4.95	2.40	282.78	524.26
3-opt	**Eil51**	50	12.28	0.00	304.04	655.01
		100	7.04	0.00	533.54	1004.98
	kroA200	50	20.17	0.00	1338.13	7371.02
		100	12.01	0.00	1985.14	13292.01

Tab III.6- Les résultats de comparaison de deux méthodes hybride sur Eil51 et KroA200 avec 50 et 100 itérations (2-opt et 3-opt)

Discussion de tab III.6

La première et la 2éme modification de la méthode hybride à une réduction significative du pourcentage d'erreur relative en augmentant le nombre d'itérations, et en augmentant le temps d'exécution Dans 2-opt et 3-opt.

III.6- Conclusion

Sur la base de six problèmes TSPLIB utilisés avec 10 agents, avec 10 opérations en 50 itérations chacune, nous pouvons voir que la 2^{éme} model hybride proposé donne le meilleure résultat en termes d'erreur relative mais de temps d'exécution élevé. La première modification donne une erreur relative importante mais nécessite peu de temps d'exécution.

Conclusion Générale

Dans ce mémoire, nous étudions un problème d'optimisation combinatoire : le problème de voyageur de commerce (PVC). Notre travail est principalement basé sur la proposition des méthodes approchées qui nous permettent de résoudre efficacement notre problème (ACO et 2-OPT) et la comparer la qualité de solution avec un autres algorithme (ACO et 3-OPT).

Après avoir étudié le problème et l'avoir résolu en utilisant l'algorithme (ACO et 2-OPT) et en comparant les résultats avec l'algorithme (ACO et 3-OPT) dans le troisième chapitre, nous avons trouvé que (ACO + 2-OPT) est meilleur que (ACO + 3-OPT) en considérant la qualité des solutions par rapport au temps d'exécution.

Pour les travaux futurs, nous suggérons aux nouveaux étudiants d'essayer de trouver des nouveaux modèles pour résoudre le problème du voyageur de commerce, afin qu'ils soient faciles et rapides et donnent des meilleurs résultats que les modèles mentionnés dans ce mémoire.

Références et bibliographie

[1] Adnan Yassine, La logistique (2014), pages 91 à 110.

[2] G.F. Hertono et al 2018 J. Phys.: Conf. Ser. 974 012032, the modification of hybrid method of ant colony optimization, particle swarm optimization and 3-OPT algorithm in traveling salesman problem.

[3] Garey, Johnson, Computers and intractability: a guide to the theory of NP-completeness,

W.H. Freeman and Company, New York, 1979.

[4] Le problem de voyageurs de commerce Bi-Objectifs, https://wikimemoires.net/,

10:30 09/06/2021.

[5] Norman Biggs, E Keith Lloyd, and Robin J Wilson. Graph Theory, 1736-1936. Oxford

University Press, 1986.

[6] Ahmia Ibtissam Thèse de doctorat en sciences présentée pour l'obtention du grade de docteur en : mathématiques spécialité : recherche opérationnelle : mathématiques de gestion université des sciences et de la technologie Houari Boumediene 2019

[7] Catherine Mancel, Modélisation et résolution de problèmes d'optimisation

Combinatoire issus d'applications spatiales, 2005, page 19.

[8] Problème du voyageur de commerce, https://fr.m.wikipedia.org/, 14:20,15/10/2021.

[9] Medjber Kahina et Salhi Souhila, Optimisation du plan de stockage de la chambre froide au niveau de l'entreprise CEVITAL, université Abderrahmane Mira de Bejaia, faculté des sciences exactes, département de la recherche opérationnelle 2016.

[10] Master option réseaux et multimédia la résolution du problème de voyageurs de commerce bi-objectifs par la métaheuristiques d'optimisation par colonie de fourmis artificielles par kouchi mohamed & mili kamel 2012 – 2013.

[11] Introduction aux algorithmes de recherche, https://www.cloudschool.org/, 14.00,12/07/2021

[12] Z. Huang, x. Miao, and p. Wang. A revised cut-peak function method for box constrained continuous global optimization. Applied mathematics and computation, 194:224–233, 2007.

[13] M. Bruyneel, p. Duysinx, and c. Fleury. A family of mma approximations for structural optimization. Struct multidisc optim, 24:263– 276, 2002.

[14] C. Hamzacebi and F. Kutay. A heuristic approach for finding the global minimum: Adaptive random search technique. Applied Mathematics and Computation, 173:1323-1333, 2006.

[15] Hanaa Hachimi, Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications, 2013.

[16] Cotta, Talbi et Alba, ParaUel Hybrid Metaheuristics, in ParaUel Metaheuristics: A New Class of Algorithms, E. Alba (Ed.) 2005, Wiley Interscience. pp 347-370.

[17] Sébastien Noël, Métaheuristiques hybrides pour la résolution du problème d'ordonnancement de voitures dans une chaîne d'assemblage automobile, mémoire présenté comme exigence partielle de la maîtrise en informatique, université du Québec à Chicoutimi à Montréal, 2007,page 20.

[18] A thesis submitted for the award of degree of doctor of philosophy in computer science under the faculty of technology submitted by page 9-10.

[19] COSTANZO Andrea, LUONG Thé Van et MARILL Guillaume, Optimisation par colonies de fourmis, 2006.page 12-13.

[20] Yoshiki Tamura, Tomoko Sakiyama, and Ikuo Arizono Ant Colony Optimization Using Common Social Information and Self-Memory, page 2.

[21] MGA Verhoeven, Emile HL Aarts, PCJ Swinkels Systèmes informatiques de la future génération 11 (2), 175-182, 1995.

[22] http://pedrohfsd.com/2017/08/09/2opt-part1.html 12:24 21/06/2021

[23] Matthias Englert, Heiko Röglin, Berthold Vöcking SODA, 1295-1304, 2007

[24] Dr. Nour El-Houda Golea Mémoire Outils De Programmation Pour Les Mathématiques Université Mostafa Benboulaid - Batna2 Faculté Des Mathématiques Et Informatique Département Socle Commun Mi, page 4.

[25] Yassine Ariba-Jérome Cadieux Manuel Matlab, Icam-Toulouse, Version 0.1 page 10.

[Dorigo 02] Un chapitre dans son livre est spécifiquement dédié aux algorithmes élémentaires des colonies de fourmis dans un livre généraliste sur le méta heuristique.

[26] Mahi M, Baykan Ö K and Kodaz H 2015 A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem Applied Soft Computing 30 pp 484–490.