REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE MATHEMATIQUE ET INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

N° :……..

MEMOIRE DE FIN D'ETUDE

Présenté pour l'obtention du Diplôme MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Informatique Décisionnelle et Optimisation
Par

-NECHE Samir -ALILI Zohir

Sujet

Méthode OCA pour le PVC appliqué au réseau routier algérien

Devant le jury :

……	Université … …….	Rapporteur
HEMMAK Allaoua	Université de M'sila	Encadreur
……	Université … …….	Examinateur

Année universitaire : 2020/2021

Remerciements

Nous tenons à remercier tout premièrement ALLAH le tout puissant pour la volonté, la santé et la patience, qu'il nous a donné durant toutes ces longues années.

Ainsi, nous tenons également à exprimer nos vifs remerciements à Mr HEMMAK ALLAOUA pour accepter de nous suivre tout le long de la réalisation de ce mémoire.

Nous tenons à remercier vivement toutes personnes qui nous ont aidés à élaborer et réaliser ce mémoire, ainsi à tous ceux qui nous ont aidés de prés ou de loin à accomplir ce travail.

Nos remerciements vont aussi à tous les enseignants et le chef de département d'Informatique qui a contribué à notre formation par ailleurs, Nos remerciements à tous les membres du jury qui ont accepté de juger notre travail.

En fin, nous tenons à exprimer notre reconnaissance à tous nos amis et collègues pour le soutien moral et matériel…

Dédicace

Nous dédions ce modeste travail

A nos parents

A nos très chers frères, à nos chères sœurs et à toutes nos familles

A tous nos amis que Nous ne pouvons pas les nommés tous ici.

A tous nos collègues sans exception.

ملخص

الهدف من هذا المشروع هو حل مشكلة البائع المتجول والتي يقصد منها التجول بين مجموعة من المدن انطلاقا من مدينة و المرور بجميع المدن مرة ومرة واحدة و العودة إلى نقطة الانطلاق باستعمال خوارزمية معسكرات النحل التي هي طريقة من طرق الحلول المثلى للمشاكل المتعددة من أجل إيجاد اقصر حلقة أو دورة هاملتونية في وقت مقبول بالتطبيق على شبكة الطرق الجزائرية،(الولايات الجزائرية الثمانية و الاربعون) المسافة بين الولايات مستخرجة من تطبيق غوغل خرائط.

الكلمات المفتاحية: مشكلة البائع المتجول، خوارزمية معسكرات النحل،الحلول المثلى للمشاكل المتعددة، حلقة هاملتونية.

Abstract

The objective of this project is to solve the traveling salesman problem, which consists of traveling between a group of cities starting with one city and going through all the cities once and only once and returning to the starting point using the bee colony algorithm, which is a combinatorial optimization method in order to find the shortest Hamiltonian cycle or circuit at a reasonable time by applying to the Algerian road network, (the forty-eight Algerian wilayas) the distance between cities taken from Google Maps application.

Keywords: TSP.BCO. combinatorial optimization, hamiltonian cycle.

Résumé

L'objectif de ce projet est de résoudre le problème du voyageur de commerce, qui consiste de voyager entre un groupe de villes en démarrant par une ville et traverser toutes les villes une et une seule fois et revenir au point de départ en utilisant l'algorithme de colonie d'abeilles, qui est une méthode d'optimisation combinatoire afin de trouver le plus courte cycle ou circuit hamiltonien à un temps raisonnable en appliquant sur le réseau routier algérien, (les quarante-huit wilayas algériens) la distance entre les villes extraite de Google Maps application.

Mots clés: PVC, ACO, optimisation combinatoire, cycle ha miltonien.

Table des matières

INTRODUCTION GENERALE

Introduction générale

En mathématiques, *l'optimisation* recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction, avec ou sans contraintes.

En théorie, un problème d'optimisation combinatoire se définit par l'ensemble de ses instances, souvent infiniment nombreuses. Dans la pratique, le problème se ramène à résoudre numériquement l'une de ces instances, par un procédé algorithmique.

Pour la résolution de ses problèmes, de nombreuses méthodes ont été développées en Recherche Opérationnelle (RO) et en Intelligence Artificielle (IA) afin de les résoudre. Ces méthodes peuvent être classées en deux grandes catégories :

- Les méthodes exactes (complètes) capables de trouver la solution optimale si elle existe,
- Les méthodes approchées (incomplètes) qui perdent la complétude afin de gagner en efficacité.

Certaines méthodes ont permis de trouver des résultats optimaux pour des problèmes de taille raisonnable, mais comme le temps de calcul nécessaire pour trouver une solution risque de croître de façon exponentielle avec la taille du problème, les méthodes exactes rencontrent des difficultés dans le cas de problèmes de taille importante, mais les méthodes approchées ont prouvé leur efficacité dans ce domaine et de trouver des solutions pour des problèmes de grande taille.

Depuis une trentaine d'années une nouvelle génération de méthodes puissantes est apparue et qui s'appelle « ***Métaheuristiques*** ».

Parmi les problèmes d'optimisation les plus connu le problème de voyageur de commerce PVC (en anglais Traveling Salsman Problem TSP) où il existe un commerçons arriver a un pays dont il doit passez par tous les villes de ce pays en démarrant par une ville passons par tous les villes une et une seul fois et à la fin aller à la ville de départ en essayions de minimiser la distance totale.

La problématique posée est comment résoudre le problème de PVC en utilisant l'optimisation par colonie d'abeille OCA (en anglais Bee Colony Optimization BCO) en prenant le réseau routière algérien.

L'objectifs de ce travail est de trouver une solution optimale (proche de la solution optimale) et dans un temps raisonnable.

Le présent mémoire organisée en quatre chapitres :

Dans le premier chapitre, nous avons parlai sur le problème de voyageur de commerce.

Dans le deuxième chapitre nous avons constant les différents recherche récentes sur le PVC.

Le troisième chapitre parle sur la phase de conception et les algorithmes utilisés ainsi les données.

Le quatrième chapitre présente l'implémentation du travail et la discussion sur les résultats

Et à la fin une conclusion générale regroupe les points forts de ce travail et ceux manquants.

Chapitre I

Le Problème de voyageur de commerce

PVC

1.1. Présentation du Le problème du voyageur de commerce:

Le problème du voyageur de commerce (Traveling Salesman Problem en anglais ou TSP) est l'un des problèmes de transport les plus étudiés en optimisation combinatoire. Il a été présenté pour la première fois sous forme de jeu par William Rowan Hamilton en 1859. L'objectif était alors de trouver le meilleur parcours possible en termes de durée, que devait emprunter un voyageur de commerce pour visiter un nombre fini de villes une et une seule fois, en retournant au point de départ à la fin du parcours.

Dès lors, le problème commença à s'étendre et eut de plus en plus d'applications notamment dans le domaine du transport d'individus et de marchandises, dans la planification (choisir l'ordre ou les trajectoires des opérations à effectuer pour terminer une tâche le plus rapidement possible ; comme dans le perçage de points dans des cartes électroniques), etc.

Tout comme la plupart des problèmes d'optimisation combinatoire, le problème du voyageur de commerce possède la particularité d'avoir un énoncé simple et facile à comprendre. Il n'en demeure pas moins qu'il soit très difficile à résoudre lorsque le nombre de villes augmente. D'ailleurs, dans l'un de leurs papiers, Garey et Johnson mettent en évidence la NP-difficulté du problème. (CHENTLI, 2018)

Un voyageur de commerce désire visiter un certain nombre de villes, débutant et finissant son parcours dans la même ville en visitant chacune des autres villes une et une seule fois. Il désire sélectionner la tournée qui minimise la distance totale parcourue.

Ce problème est connu sous le nom du problème du voyageur de commerce (en anglais travelling salesman problem : TSP) et est NP-Complet. La complexité en temps des algorithmes exacts proposés croît exponentiellement avec n (la taille du problème ou le nombre de villes). Plusieurs méthodes d'approximation (heuristiques 1) ont été proposées qui approchent en temps raisonnable la solution optimale.

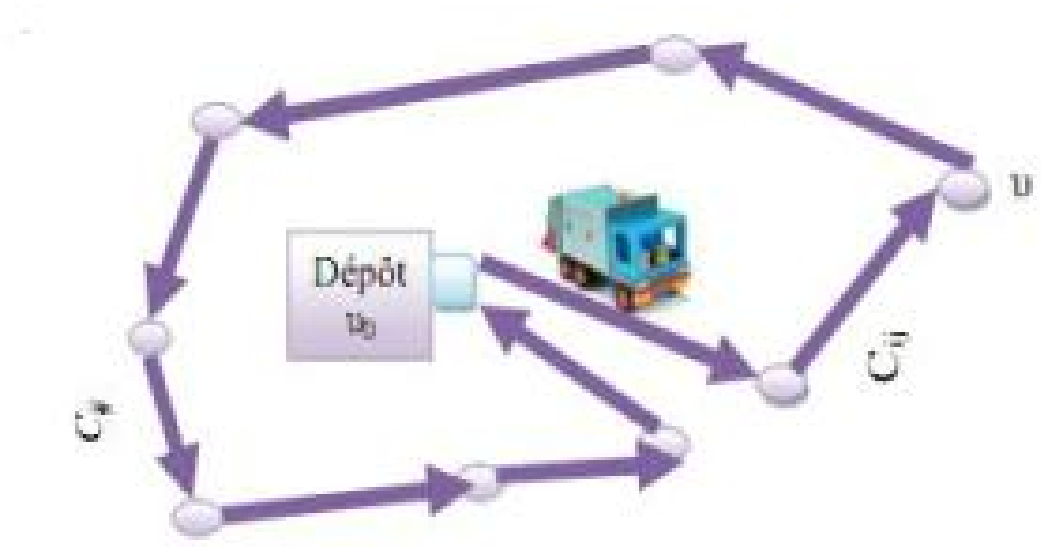


Figure 1 : Présentation graphique du problème de voyageur de commerce TSP

1.2. Histoire non-exhaustive du TSP

- ✓ Premières références en 1832.
- ✓ 1859, W.R. Hamilton : énonce avec un voyageur de commerce.
- ✓ 1949, J.B. Robinson, \On the Hamiltonian game (a traveling-salesman problem)". Première référence sous sa forme moderne.
- ✓ Années 30-50 : popularise dans la communauté mathématique
- ✓ par M. Flood et les chercheurs du RAND.
- ✓ 1954, RAND : G. Dantzig, R. Fulkerson, et S. Johnson,

Solution of a large-scale traveling-salesman problem".

Solution exacte pour les 49 capitales des états américains

(Introduction des coupes et du B&B).

- ✓ 1962, M. Held et R.M. Karp : introduction d'heuristiques
- ✓ basées sur la programmation dynamique.
- ✓ 1973 : Heuristique de Lin and Kernighan.
- ✓ 1976 : Heuristique de Christofides.
- ✓ Années 80 - aujourd'hui : diverses heuristiques et métaheuristiques

1.3. Définition

1.3.1 Définition d'un cycle ou d'un circuit hamiltonien

Un cycle hamiltonien est un cycle qui passe une fois et une seule par chaque sommet du graphe. On dit aussi **tournée** ou **tour**.

Si le graphe est orienté, on définit de la même manière un circuit hamiltonien.

Dans un graphe de n sommets, le nombre de tournées est égal à (n-1) !, qui correspond aux nombres de permutations de n-1 éléments. ((n-1)! et non n! car on peut partir d'un sommet quelconque).

A chaque arête (ou arc si le graphe est orienté) on affecte un nombre cij qui peut, par exemple, représenter un coût, un temps ou… une longueur. On l'appelle longueur de l'arête (ou de l'arc).

1.3.2. Définition du problème du voyageur de commerce

Définition 01 :

Un voyageur de commerce doit visiter n villes données en passant par chaque ville exactement une fois. Il commence par une ville quelconque et termine en retournant à la ville de départ. Les distances entre les

villes sont connues. Il faut trouver le chemin qui minimise la distance parcourue. La notion de distance peut-être remplacée par d'autres notions comme le temps qu'il met ou l'argent qu'il dépense.

Le TSP présente plusieurs caractéristiques communes aux problèmes d'optimisation combinatoire et fournit ainsi une plate-forme idéale pour étudier les méthodes générales applicables à un grand nombre de ces problèmes. (TetiBernard, et al., 2006)

Définition 02 :

Cas non orienté ou symétrique cij = cji

Il s'agit de déterminer parmi tous les cycles hamiltoniens, celui (ou ceux) de longueur minimale.

Cas orienté (ou non symétrique) cij ≠ cji

Il s'agit de déterminer parmi tous les circuits hamiltoniens, celui (ou ceux) de longueur minimale.

Le problème du voyageur de commerce est le cas le plus simple du problème d'organisation de tournées de livraisons.

1.4. Formulation mathématique du problème du voyageur de commerce

1.4.1. Enoncé

On détermine les variables nécessaires pour faire renonciation mathématique du Problème du voyageur de commerce :

$d_{\varepsilon\sigma}$ = distance entre la ville ε et la ville σ ;

n = nombre de villes;

$x_{\varepsilon\sigma}$= variable binaire qui prend la valeur I si la ville ε est visitée immédiatement avant la villeσ. Sinon, cette variable prend la valeur 0.

1.4.2. Formulation.

On doit minimiser la longueur du cycle hamiltonien,[1] donc la fonction-objectif sera :

$$\sum_{\varepsilon=1}^{n} \sum_{\sigma=1}^{n} d_{\varepsilon\sigma} x_{\varepsilon\sigma} \qquad (1)$$

Les contraintes sont les suivantes :

$$\sum_{\varepsilon=1}^{n} x_{\varepsilon\sigma} = 1 \text{ pour tout nœud } \sigma \qquad (2)$$

$$\sum_{\sigma=1}^{n} d_{\varepsilon\sigma} = 1 \text{ pour tout nœud } \varepsilon \qquad (3)$$

$$\sum_{\varepsilon\in E} \sum_{\sigma\in E} x_{\varepsilon\sigma} \leq |E| - 1 \quad \text{pour tout } E \subset N \text{ avec } \quad 2 < |E| \leq n\text{-}1 \qquad (4)$$

La contrainte (2) s'assure qu'on ne sort qu'une seule fois de chacun des points et la contrainte (3) vérifie qu'on entre qu'une seule fois à chaque point. Donc, on s'assure que chaque point est visité une seule fois.

[1] Un cycle Hamiltonien : c'est un cycle qui relie tous les sommets une seule fois.

Ensuite on ajoute la contrainte (4) pour que des sous tours ne se forment pas. Dans la contrainte (4), |E| représente la cardinalité de l'ensemble E. (MUNTEANU, 2009)

1.4.3. Problème du voyageur de commerce symétrique et asymétrique

Le voyageur de commerce présente une particularité au niveau des caractéristiques du graphe. Cette particularité est liée aux arcs du graphe, car il peut y avoir une masse inégale d'après le sens dans lequel on le parcourt. La matrice de ce problème est une matrice asymétrique. Si les distances sont égales la matrice du problème sera symétrique.

Dans le cas symétrique le problème doit respecter la condition suivante :

$$d_{\varepsilon\sigma}=x_{\varepsilon\sigma} \qquad \forall \epsilon, \sigma$$

De plus les distances peuvent respecter l'inégalité du triangle, donc on applique la condition :

$$d_{\varepsilon k} \leq d_{\varepsilon\sigma} + d_{\sigma k} \; \forall \epsilon, \sigma, k$$

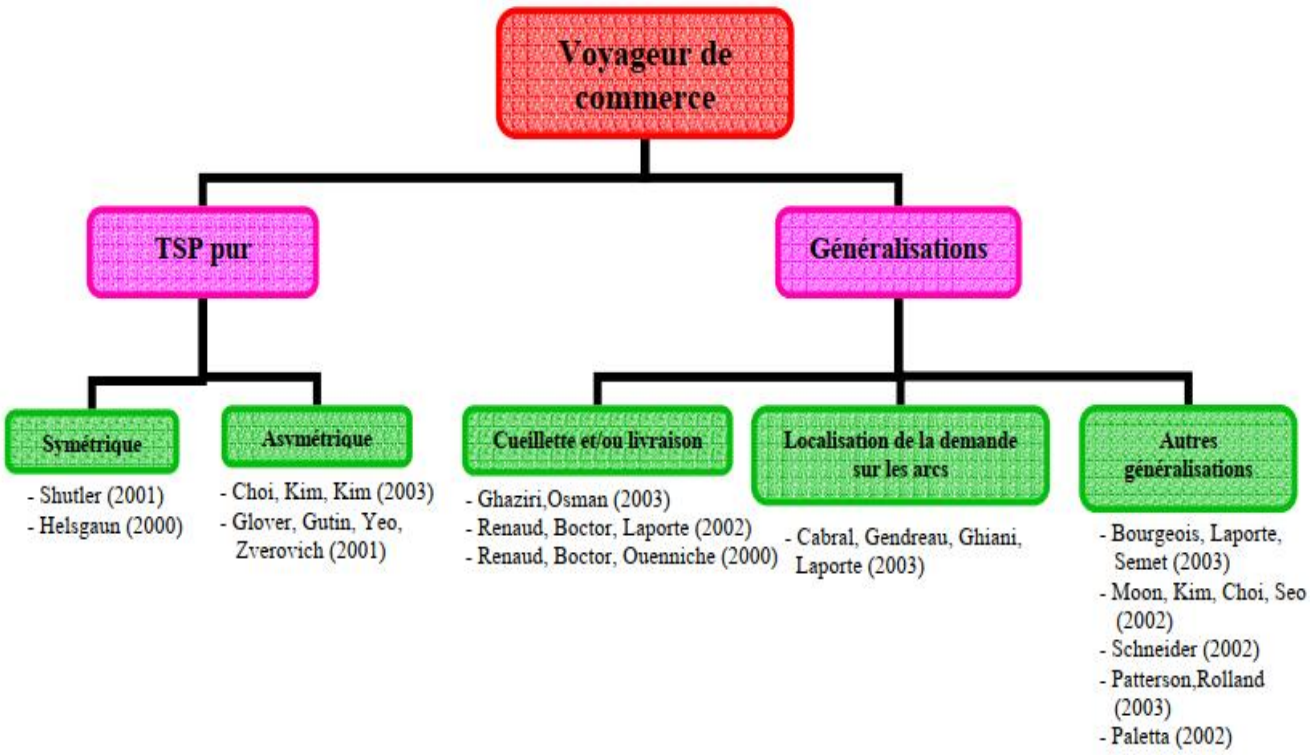


Figure 2 : Classification des problèmes du voyageur de commerce et ses généralisations

- **Problème du voyageur de commerce symétrique**

Shutler (2001) présente une approche de résolution pour le problème du voyageur de commerce symétrique en améliorant un algorithme de séparation et d'évaluation progressive pour un problème de l'ordre de cent à cinq cents nœuds. Jusqu'à ce moment, la meilleure heuristique est considérée celle de Heisgaun (2000) et l'algorithme de Lin-Kernigan (Glover, Gutin et al, 2001).

- **Problème du voyageur de commerce asymétrique**

Glover, Gutin et al. (2001) développent trois (3) heuristiques de construction pour un graphe orienté, où chacun des arcs présente des valeurs différentes. Ces heuristiques représentent de nouvelles approches pour la construction du problème de voyageur du commerce asymétrique. L'heuristique basée sur la méthode Karp-Steele patching (la première développée par Gutin et Glover) donne des solutions qui sont proches de l'optimum (à 3.36 % de l'optimum).

1.5. Extension au problème du voyageur de commerce

Une autre particularité du problème du voyageur de commerce est donnée par le type d'opérations effectuées. C'est-à-dire qu'on peut avoir différents types de problèmes du voyageur de commerce en modifiant le type d'opérations.

Problème du voyageur de commerce avec retour à charge

Gendreau, Hertz et al. (1996) proposent une heuristique en respectant l'inégalité du triangle et des coûts symétriques. En appliquant cette heuristique, ils ont obtenu un résultat situé au 3/2 de l'optimum. Les tests ont été effectués sur trente problèmes de 100, 200 et 300 nœuds. Le type d'heuristique utilisé était l'algorithme GENIUS développé par Gendreau, Hertz et Laporte. Cet algorithme est constitué de deux (2) phases : la phase d'insertion {GENeralized Insertion) et la phase de post-optimisation {Unstringing et Stringing). Pour la phase d'insertion {GENeralized Insertion) ils ont choisi trois (3) nœuds et ils ont ensuite inséré d'autres sommets pour former un cycle hamiltonien.

Pour la phase de post-optimisation {Unstringing et Stringing), ils ont considéré le retrait de chaque nœud et après la réinsertion de ces nœuds dans le tour pour tenter de réduire le coût de la tournée.

Mladenovic et Hansen (1997) proposent une variante de GENIUS, avec une structure de recherche, appelée GENIUS-VNS. En utilisant cette méthode, ils ont obtenu une croissance de 0,4 % par rapport à la méthode proposée par Gendreau, Hertz et Laporte.

Par contre, le temps de calcul avait augmenté de 30 % par rapport à la méthode GENIUS.

Ghaziri et Osman présentent une nouvelle heuristique basée sur les réseaux de neurones, nommée SOFM. Ils utilisent quatre (4) types d'interactions : la première interfère avec les clients ayant une grande distance alors que la deuxième interfère avec les clients qui nécessitent une cueillette. Les deux (2) autres interactions traitent la modalité d'interagir entre l'ensemble (les deux types des clients) et les dépôts et la modalité d'interaction entre les chaînes. Les tests ont été effectués sur des problèmes ayant entre 100 et 1000 nœuds. (CHENTLI, 2018)

1.6. Complexité

Le PVC est un problème NP-difficile car le problème de décision qui lui associé est NP-complet. Il peut s’´enoncer en un problème de décision comme suit : Etant donné un entier k positif. Existe-t-il un cycle passant une et une seule fois par chaque sommet tel que la somme des coûts des arcs utilisés soit $\leq$ k ? (KOUIDER AIAD, 2009)

Proposition :

Le nombre de cycles Hamiltoniens dans un graphe complet non orienté k_n *est égal à* $\frac{(n-1)!}{2}$.

En effet, un cycle Hamiltonien dans $k_n$correspond `a une permutation de l'ensemble de ses sommets. Or, une permutation engendre 2n cycles Hamiltoniens identiques dansk_n. Donc, les n! permutations engendrent $\frac{n!}{2n} = \frac{(n-1)!}{2}$ cycles Hamiltoniens.

La première idée qui vient a` l'esprit lorsqu'on veut résoudre PVC est certainement celle de la recherche exhaustive. Le principe en est tr`es simple, mais au prix d'une complexité en temps très élevée : il consiste a` déterminer la longueur de tous les cycles Hamiltoniens, puis à sélectionner la meilleure.

Comme dans un graphe completk_n, il y a $\frac{(n-1)!}{2}$ parcours possibles (cycles Hamiltoniens), n étant égal au nombre de villes.

Ainsi, pour 3 villes, il n'y a qu'un parcours possible, avec 5 villes il y a 12 parcours possibles.

Mais quand n = 25, le nombre de parcours devient tr´es important. Nous sommes confrontés

La au problème d'explosion combinatoire au sens ou le nombre de calculs nécessaires pour trouver la solution idéale s'accroît de manière exponentielle. Plus simplement, pour une petite augmentation de la complexité du problème (ajout d'une ville), le nombre de parcours possibles peut s'accroître de plusieurs millions. Supposons un ordinateur assez rapide pour évaluer une possibilité en une microseconde, voici un petit aperçu de cette explosion :

Nombre de villes «n»	Nombre de possibilités	Temps de calcul
5	12	12 μs
10	181440	0,18 s
15	43 milliars	12 heures
20	60.10^{15}	1928 ans
25	310.10^{21}	9,8 milliards d'années
32	411.10^{31}	13.10^{10} milliards d'années

La table 1. Nombre de possibilités et temps de calcule pour n villes

La table 1. Démontre bien l'explosion du nombre de possibilités lorsqu'on augmente le nombre de villes à visiter. Donc il s'avère très vite qu'une énumération exhaustive est impraticable pour un nombre élevé´e de villes. Par conséquent, lorsqu'on dépasse une quinzaine de villes, le problème ne peut plus être résolu à l'optimal.

1.7. Applications

Il existe de nombreuses utilisations pratiques du TSP dans la vie réelle. Bien sûr, le plus commun dont sont des problèmes de routage de transport. Dans cette paragraphe, certains des plus populaires

1.7.1. Le processus de peinture des carrosseries automobiles

Le processus de peinture des carrosseries automobiles est complexe et, dans certains cas, il comprend le passage des carrosseries dans une cuve de peinture. Après une série d'une peinture donnée, on change de peinture. Le passage d'une peinture à une autre génère un coût fixe, dû au temps d'arrêt pour le nettoyage de la cuve ainsi qu'à la perte des matières premières résiduelles. Le coût qui en découle dépend des peintures qui se succèdent.

Le problème est de déterminer l'ordre de passage des différentes peintures de manière à minimiser tous les coûts de changement.

Ce problème est modélisé par un problème de voyageur de commerce. On définit un graphe dont les sommets sont associés aux différents coloris. Chaque arc (le problème est non symétrique car passer de i à j n'a pas le même coût que passer de j à i), représente le passage d'une peinture à la suivante. La longueur correspond au coût de changement de coloris. (Stiven).

1.7.2. Le vendeur

Un vendeur prendrait pour visiter chaque emplacement géographique dans une liste spécifiée telle cette distance totale minimale est parcourue. Prenons le cas d'un vendeur voyageant de porte à porte dans une certaine subdivision de maisons. Ne serait-il pas très commode que le vendeur pourrait obtenir une liste de toutes les maisons dans cette subdivision spécifiée dans le plus ordre optimal de visite.

Que diriez-vous d'un vendeur qui doit visiter des centaines de villes répartis dans tout un pays ? Ou, qu'en est-il d'un groupe musical en tournée à travers le monde? Connaître le circuit optimal qui visitera chaque ville une fois pourrait économiser potentiellement des jours ou même des semaines de voyage au voyageur. Envisagez un poste livreur qui se rend au travail le matin avec un camion plein de colis à livre Dans quel ordre ces colis doivent-ils être livrés pour minimiser la distance totale parcourue ?

Pour toutes les instances mentionnées jusqu'à présent, les nœuds du graphique correspondraient à les emplacements géographiques, et les distances seraient des valeurs métriques basées sur les longueurs des routes reliant les localités. Actuellement, la plupart des pays du monde, y compris les villes, bâtiments et points de repère, a déjà été cartographié électroniquement dans un avion. Résoudre ces cas mentionnés, il suffirait de spécifier les emplacements souhaités à visité, puis laissez un algorithme de solution TSP faire le reste. (Zambito, 2006)

1.7.3. Parcourir de la perceuse.

Considérez certains des machines dans une chaîne de montage. Il existe des machines dont le seul but est de percer divers trous dans un certain morceau de matériau. Le matériau peut être une carte de circuit imprimé, le châssis d'un véhicule, ou même un morceau de bois pour construire une étagère à livres.

La perceuse est repositionnée par des moteurs qui glissent le long des rails de sorte que la perceuse puisse se déplacer vers n'importe quelle position dans une certaine zone. Il faudra un certain temps pour repositionner la perceuse en fonction de la distance que la perceuse doit parcourir.

Une solution au TSP pourrait être utilisée pour trouver l'ordre optimal dans lequel les trous doivent être percés. Dans le cas d'un assemblage ligne, économisant plusieurs secondes pour terminer le processus pour chaque pièce signifie produire beaucoup plus de pièces à la fin de la journée.

Pour résoudre ce problème en utilisant l'application TSP, laissez simplement les sommets représenter les emplacements que les trous doivent être percés et laisser les bords être les distances entre eux.

1.7.4. Placement de connexion mécanique.

Une autre application à laquelle une solution au TSP peut être appliquée est électronique ou placement de connexion mécanique. Considérez le câblage d'un circuit imprimé, ou le câblage électrique dans un grand bâtiment, ou même la disposition de la plomberie dans un bâtiment.

Dans bon nombre de ces cas, les connexions doivent être disposées de telle sorte que les composants soient tous connectés dans un cycle. Dans le cas d'un circuit imprimé, les connexions sont les fils et les composants sont les transistors, les résistances, etc. Quand on parle de la configuration électrique d'un bâtiment, les connexions sont les fils et les composants sont les interrupteurs, les prises, luminaires, etc. Enfin, pour l'aménagement de plomberie d'un bâtiment, les raccordements sont les tuyaux et les composants sont les robinets et les robinets d'eau. Dans tous ces cas, vous serez essayé de trouver un chemin hamiltonien le plus court afin d'économiser de la matière et d'optimiser le flux en réduisant la durée du cycle. Circuits de connexion ou câblage de composants électroniques de sorte que le courant doit parcourir une distance aussi minimale que possible finira par augmenter l'efficacité et la performance globale. (Zambito, 2006)

1.8. Méthodes de résolutions

Parmi les méthodes de résolution du TSP, nous allons nous intéresser aux algorithmes de colonies de fourmis, la méthode du plus proche voisin, Algorithme Génétique et la méthode de Colonie d'abeille

1.8.1. Algorithme de colonie de fourmis

Les algorithmes de colonies de fourmis sont inspirés du comportement de fourmis et constituent une famille de méta-heuristique. (BEKADA, 2015)

- **Historique**

La méta-heuristique "ant colony optimization"est inspirée par les travaux de Deneubourg[4].

Ce concept est relativement récent puisqu'il a commencé en 1991 avec Colorni,Dorigo et Maniezzo. Il avait pour but de résoudre le problème du voyageur de commerce.

Le premier algorithme s'inspire du comportement des fourmis recherchant un chemin entre leur fourmilière et une source de nourriture. L'idée originale s'est depuis diversifiée pour résoudre une classe plus large de problèmes.

La solution étant insatisfaisante, des améliorations ont été apportées en 1995. Dès 1994 elle a pu être appliquée à d'autre problème d'optimisation combinatoire.

- **Les fourmis :**

Les fourmis sont des insectes qui communiquent "chimiquement" à l'aide de phéromones. Ce moyen de communication leur permet, entre autre, de choisir un plus court chemin. Aucune entité ne les contrôle, elles sont toutes auto-organisées.

- **Les phéromones :**

Les phéromones sont des substances volatiles que les fourmis perçoivent grâce à des capteurs sur leurs antennes, déposent sur le sol par une glande située sur leur abdomen en créant ainsi des pistes chimiques.

- **Algorithme de colonie de fourmis pour le problème du voyageur de commerce**

Dans l'algorithme de colonie de fourmis, chaque fourmi est initialement placée sur une ville (sommet de graphe), chacune possède une mémoire qui stocke la solution partielle qu'elle a construite auparavant.

Soient b_i (t) l'ensemble des fourmis dans la ville i au temps t tel que i = 1,……, k et

n =$\sum_{i=1}^{k}$ bi (t) le nombre total des fourmis.

Chaque fourmi est un agent avec les caractéristiques suivantes :

La fourmi choisit la prochaine ville de destination avec une probabilité qui est en fonction de la distance et de la quantité des phéromones présente sur l arête de connexion. Cette probabilité est formulée à travers la règle de dépl

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^{\alpha}.(\eta_{ij})^{\beta}}{\sum_{l \in N^k}(\tau_{il}(t))^{\alpha}.(\eta_{il})^{\beta}} & \text{si } j \in N^k \\ 0 & \text{sinon} \end{cases}$$

avec :

τ_{ij} (t) est l'intensité de la trace de phéromones dans l'arête (i; j) à l(instant t.

$\eta_{ij=\frac{1}{d_{ij}}}$ "visibilité" : une information heuristique à priori valable, ou d_{ij} la distance entre la ville i et la ville j, l'idée étant d'attirer les fourmis vers les villes les plus proches.

α, β Sont deux paramètres qui déterminent l'influence relative à la trace de phéromones et de l'information heuristique.

N^k Est le voisinage faisable de la fourmi k. Il représente l'ensemble des villes qui restent à visiter. Celles déjà visitées sont retirées de la liste.

La construction de la solution s'achève lorsque chaque fourmi complète un tour et dépose une quantité de phéromones sur toutes les arêtes (i; j) visitées. Au temps t, elle choisit la prochaine ville où elle s'y

positionnera au temps t + 1. Afin d'accomplir leurs tours, les fourmis effectuent n itérations et les traces des phéromones sont mises à jour selon la formule ci-dessous :

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

Où, ρ est un coefficient tel que $0 < \rho < 1$

(1- ρ) est l'évaporation de la phéromone entre le temps t et t + n

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{n} \Delta\tau_{ij}^{k}(t) \qquad (2)$$

représente la quantité des phéromones par unité de longueur déposée sur l'arête (i; j) par la k^{éme}

fourmi entre t et t + n , elle est donnée par :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L_k} & \text{si la k}^{éme}\text{ fourmi traverse l'arête (i; j) dans son tour} \\ 0 & \text{sinon} \end{cases} \qquad (3)$$

Avec

L_k :la longueur du tour de la k^{éme} fourmi.

Q : un nombre positif constant.

Algorithme

Début

Donner n (n le nombre de villes)

Déterminer la matrice des distances ou la matrice des coûts d(i,j)

Choisir le nombre de fourmis n

n fourmis n villes

Initialisation

$\alpha, \beta, Q, \rho, \tau_{ij}(0)$ = c (avec c > 0)

Donner la solution optimale (sol_opt)

pour i = 1 à n faire

Pour j = 1 à n faire

Si i ≠ j

$\eta_{ij} \leftarrow 1/d_{ij}$

$\tau_{ij} \leftarrow c$

Sinon

$\tau_{ij} \leftarrow 0$

Fin si

Fin pour

Fin pour

pour t= 1 à t_max (t_max = nombre maximum d'itérations)

Placer aléatoirement les n fourmis sur les n sommets du graphe

pour chaque fourmi k (k = 1 a n)

initialiser la liste L_v comme suit :

$L_v \leftarrow$ sommet affecté

pour i = 1à n(ville) faire

Pour k = 1 à n(fourmi) faire

N^k : ensemble des villes non visitées par la fourmi k.

A chaque pas, la fourmi k située dans une ville *i* choisira une ville *j* du voisinage possible N^k (villes non visitée) selon la règle de déplacement

Fin pour

Actualiser la liste des villes visitées par la fourmi k

$L_v \leftarrow L_v \cup \{j\}$

Fin pour

pour k = 1 à n(fourmi) faire

Calculer le coût de la tournée L^k (égal à la somme des poids des arêtes qui la composent L^k)

Pour i = 1 à n(ville) faire

Déposer une piste $\Delta\tau_{ij}^{k}(t)$ sur le trajet $Tabou^k$ conformément à l'équation *(3)*

Fin pour

Fin pour

Mettre à jour les traces de phéromone selon la règle (1)

$L \leftarrow \min(L^k)$ on trouve le longueur minimum pour chaque tour de chaque fourmi

gap ← (L ← sol_opt) / sol_opt

Afficher (le nombre d itération t, le coût de la tournée L et l erreur gap)

si *gap = 0*

Stop

Fin si

Fin pour

Fin de l'algorithme

1.8.2. Algorithme du plus proche voisin

L'algorithme du plus proche voisin est une heuristique de construction simple. C'est l'un des premiers algorithmes utilisés pour déterminer une solution au problème du voyageur de commerce. Il donne rapidement une courte visite, mais généralement pas la solution optimale est un algorithme de type glouton qui fait un choix à chaque étape, sans jamais remettre en cause ce choix, On choisit une première ville au hasard et on construit un chemin en allant vers la ville la plus proche appartenant pas déjà au chemin, jusqu'à ce que tous les sommets aient été parcourus. (BEKADA, 2015)

- **Principe**

On part d'une ville quelconque et l'on se dirige vers la ville la plus proche sans repasser par une ville déjà visitée.

La première étape repose sur le choix aléatoire d'une première ville, et les étapes suivantes consistent à se déplacer de ville en ville en appliquant la règle du plus proche voisin, c'est-à-dire en sélectionnant la prochaine ville telle que le poids entre la ville courante et la prochaine ville soit minimal, et ce, jusqu'à avoir visité toutes les villes. Il faut en…n revenir à la première ville choisie, pour obtenir un cycle.

Exemple Le graphe ci-dessous représente les distances entre 5 villes

$$D=\begin{bmatrix} - & 6 & 6 & 1 & 4 \\ 6 & - & 8 & 3 & 5 \\ 1 & 8 & - & 6 & 4 \\ 1 & 3 & 6 & - & 2 \\ 4 & 5 & 4 & 2 & - \end{bmatrix}$$

S∝=1+2+4+8+6=21

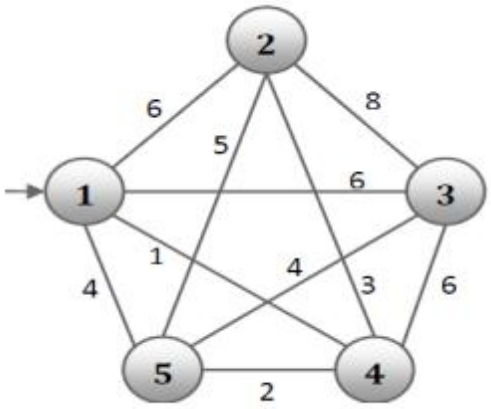


Figure 3 : graphe 5 villes

Si on part de la première ville, la tournée trouvée par la méthode du plus proche voisin est :

1-4-5-3-2-1

- **L'algorithme du plus proche voisin utilisé pour déterminer une solution au problème du voyageur de commerce**

Algorithme

Début

Choisir un sommet $v1 \in V$

1. Poser k $\leftarrow$1

pose U = $\{v_1\}$

Tant que k < n

k $\leftarrow$ k + 1

choisir v_k dans X = V \ U qui vérifie

d (v_{k-1}, v_k) = $\min_{x \in X}$ d (v_{k-1}, $\propto$)

faire

$U \leftarrow U \cup \{v_k\}$

Fin tant que

Fin de l'algorithme

1.8.3. Algorithme génétique

- **Langage spécifique à la génétique**

L'algorithme s'inspire grandement de la théorie de l'évolution et des principes génétiques donc nous utiliserons un langage spécifique à la génétique. (ANDRÉ, 2006)

- **Individu** : un trajet possible
- **Population** : un ensemble de trajets
- **Mutation** : modification aléatoire dans le trajet d'un individu
- **Adaptation** : plus le trajet est court, plus il est adapté au problème
- **Sélection naturelle** : élimination des individus les moins adaptés
- **Principe**

1. On génère aléatoirement une première population d'individu
2. On fait la sélectionne la moitié des individus (sélection naturelle)
3. On arrange les individus restant en couple et on crée pour chaque couple un couple d'enfant qui représente des trajets ayant des caractéristiques du trajet de leurs deux parents
4. les enfants peuvent avoir une chance sur q d'avoir une mutation (paramètre q définie par l'utilisateur)
5. On retourne à l'étape 2.

Première génération : On génère aléatoirement n trajet possible. Le nombre n est fixé par l'utilisateur.

Plus il sera grand plus l'algorithme sera efficace et long.

Sélection : On ne souhait garder que la moitié des individus, il existe deux méthodes pour faire cette sélection.

Méthode élitiste : On ne conserve que la meilleure moitié

Méthode par tournois : on regroupe les individus par couple et on les laisse se battre. Le moins adapté et supprimé et l'autre est conservé

On appliquera chacune des deux méthodes une fois sur deux.

Passage à la génération suivante on regroupe les individus par couple et chaque couple passe par l'opérateur croisement

L'opérateur de croisement prend en entrée deux individus et donne en sortie ces deux mêmes individus ainsi que deux "enfants" qui reprennent des caractéristiques de leurs deux "parents". L'endroit ou il y a croisement est définie aléatoirement.

Durant leurs naissances (ou juste après, nous sommes en informatique, nous avons le droit) on applique a un individu sur q l'opérateur mutation qui modifie de façon aléatoire le trajet. Par exemple inter changer la position de deux points dans le trajet.

Critère d'arrêt : cet algorithme ne termine à priori pas. Il faut donc demander à l'utilisateur de fixer un nombre m fini d'itération à faire.

Algorithme

Données : n, i, q, L % n est le nombre d'individu par population, i le nombre d'itération voulu, q

L'inverse du nombre de chance qu'un individu soit un mutant à la naissance, et L contient les villes à parcourir avec des informations sur les distances entre les villes

Résultat : L′ : liste % contient la liste du parcours de villes à effectuer

début

P′ ← Générer-population-initial S

pour z = 1 à i faire

si z = 0 mod 2 alors

P ← Sélection-élitiste P

fin

si z = 1 mod 2 alors

P ← Sélection-par-tournois P

fin

P ← Passage-génération-suivante P

fin

Élément-le-mieu-adapté P

fin

Complexité en i ∗ $O(l^2)$

Application sur un exemple

Choix des paramètres Un des points faibles de cet algorithme, sinon son caractère aléatoire, c'est le « grand » nombre de variable que doit fournir l'utilisateur. (ANDRÉ, 2006)

Nous avons obtenus des résultats satisfaisants, en un temps raisonnable, avec les paramètres suivant:

n=2 000

q=1

i =1 00

Temps : 47 sec

Parcours : 1039.9

1.8.4. L'algorithme de Little

L'algorithme de Little est un algorithme de résolution du TSP par Branch & Bound

- La séparation consiste à considérer l'inclusion ou l'exclusion d'un trajet (i, j) dans une tournée. Chaque séparation produisant deux branches, l'arbre de recherche est binaire (***figure*** 4b)
- L'évaluation fournit une borne inférieure du coût de la tournée en effectuant des opérations sur la matrice de coûts (***figure*** 4a). (Little, 1963)

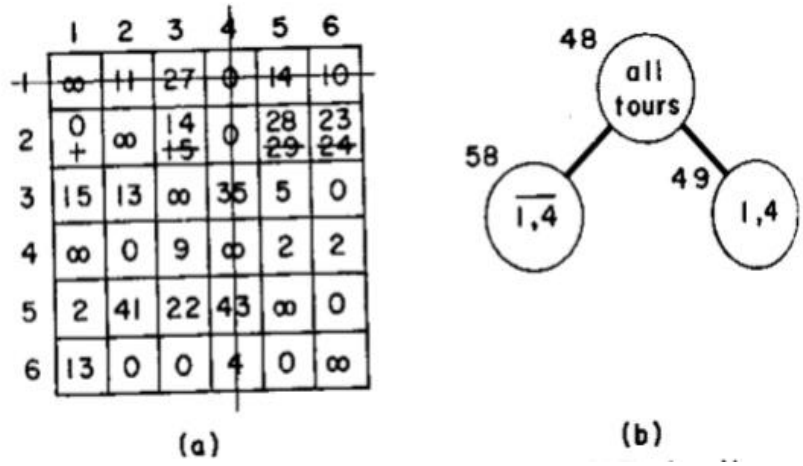


Figure 4 (tirée de l'article)

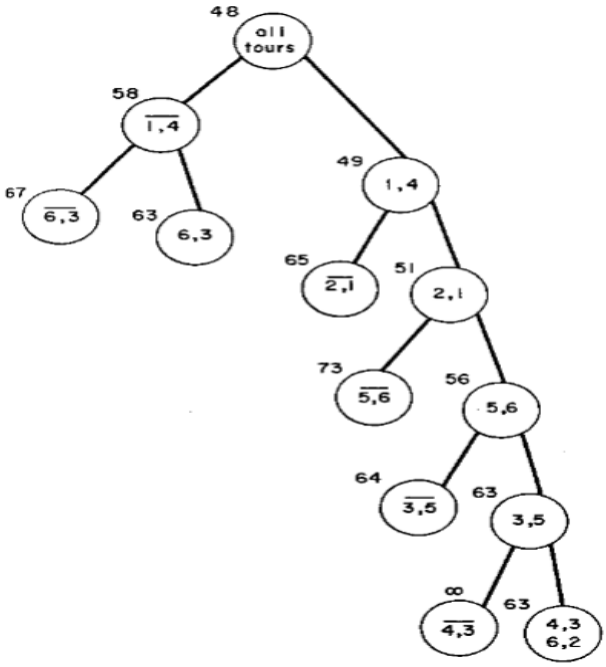


***figure 05* l'arbre de recherche est binaire**

On développe l'arbre de façon préférentielle vers la droite (Inclusion de trajets) pour rapidement aboutir à une

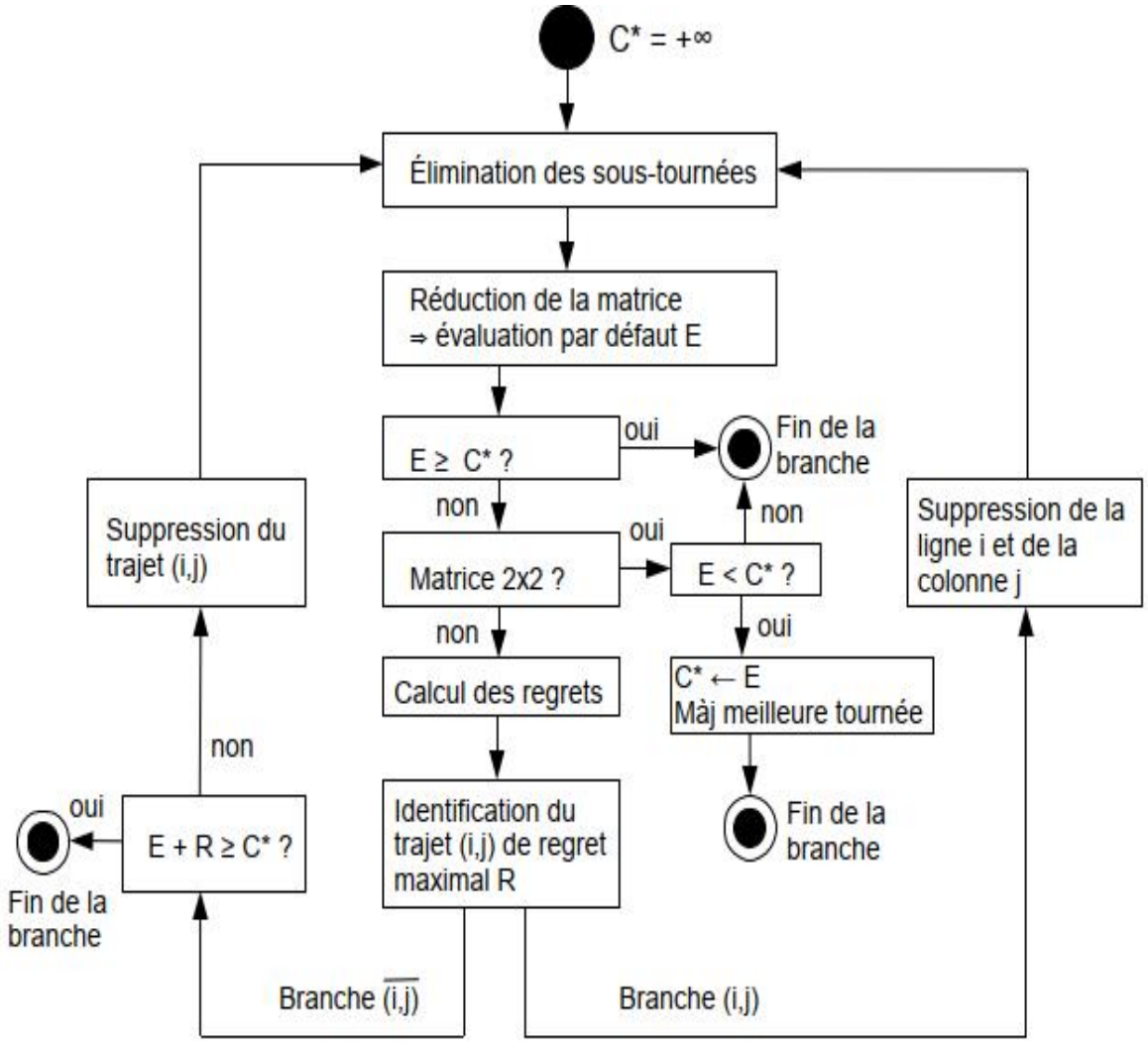
tournée complète T, de coût c Dans la ***figure 05***, T contient les trajets(1,4), (2,1), (5,6),

(3,5), (4,3) et (6,2), ce qui donne T = (1,4,3,5,6,2,1), de coût 63

On développe ensuite les autres branches pour tenter de trouver une meilleure tournée que T. Dès qu'un nœud a un coup supérieur ou égal à c, on peut interrompre la branche Dans la figure, toutes les

branches sont de coût ≥ 63, on est donc sûr que T est la tournée optimale.

- **Diagramme de L'algorithme de Little**



- **Appliquer l'algorithme de Little pour résoudre le problème de voyageur de commerce**

Avec 6 villes et 60 tournées possibles (120 si coûts asymétriques)

B: Bordeaux

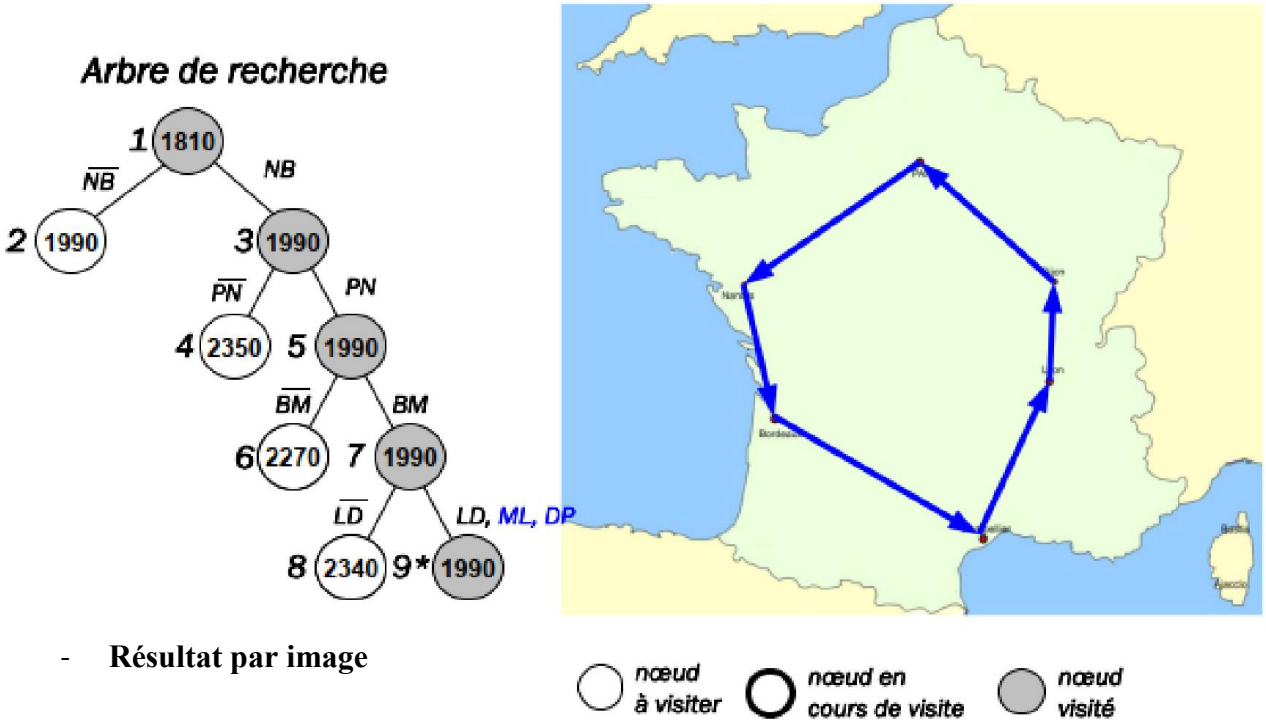
L: Lyon

N: Nantes

P: Paris

M: Montpellier

D: Dijon



- **Résultat par image**

Figure06 : **Résultat par image de l'algorithme de Little**

1.8.5. La méthode de colonie d'abeille

Dans cette méthode, les abeilles artificielles représentent des agents qui en collaborant les unes avec les autres, pour résoudre le problème de PVC et cette méthode est l'objet de notre mémoire cf. Chapitre 3.

1.9. Conclusion

Le problème du voyageur de commerce est un problème d'optimisation combinatoire plus utilisé pas spécialement dans le domaine de transport mais pour plusieurs autres domaines.

CHAPITRE 2

Etat de l’art

4.1. Introduction

Les problèmes mathématiques liés au problème du voyageur de commerce ont été traités dans les années 1800 par le mathématicien irlandais Sir William Rowan Hamilton et par le mathématicien britannique Thomas Penynton Kirkman.

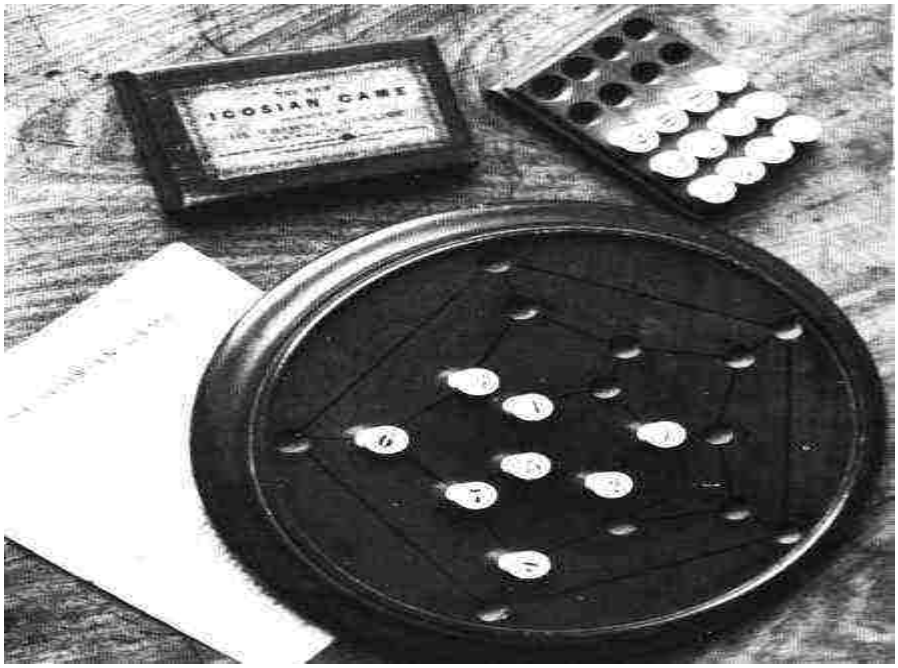


Figure1 jeu Icosian de Hamilton

La figure 1est une photographie du jeu Icosian de Hamilton qui oblige les joueurs à effectuer des tournées à travers les 20 points en utilisant uniquement les connexions spécifiées.

Les origines du TSP sont obscures. Dans les années 1920, le mathématicien et économiste Karl Menger en fit la publicité auprès de ses collègues viennois. Dans les années 30, le problème réapparut dans les cercles mathématiques de Princeton. Dans les années 1940, il a été étudié par des statisticiens (Mahalanobis (1940), Jessen (1942), Gosh (1948), Marks (1948)) en lien avec une application agricole et le mathématicien Merill Flood l'a popularisé auprès de ses collègues de la RAND Corporation. . Finalement, le TSP a gagné en notoriété comme le prototype d'un problème difficile en optimisation combinatoire : examiner les tours un par un est hors de question en raison de leur grand nombre, et aucune autre idée n'était à l'horizon depuis longtemps.

4.2.Le premier grand TSP

Une percée est survenue lorsque George Dantzig, Ray Fulkerson et Selmer Johnson (1954) ont publié une description d'une méthode de résolution du TSP et illustré la puissance de cette méthode en résolvant une instance avec 49 villes, une taille impressionnante à l'époque. Ils ont créé cette instance en choisissant une ville dans chacun des 48 États des États-Unis (l'Alaska et Hawaï ne sont devenus des États qu'en 1959) et en ajoutant Washington, D.C. ; les coûts de déplacement entre ces villes étaient définis par les distances routières. Plutôt que de résoudre ce problème, ils ont résolu le problème des 42 villes obtenu en supprimant Baltimore, Wilmington, Philadelphie, Newark, New York, Hartford et Providence. En fin de

compte, une tournée optimale à travers les 42 villes a utilisé le bord reliant Washington, D.C. à Boston; puisque le chemin le plus court entre ces deux villes passe par les sept villes supprimées, cette solution du problème des 42 villes donne une solution du problème des 49 villes.

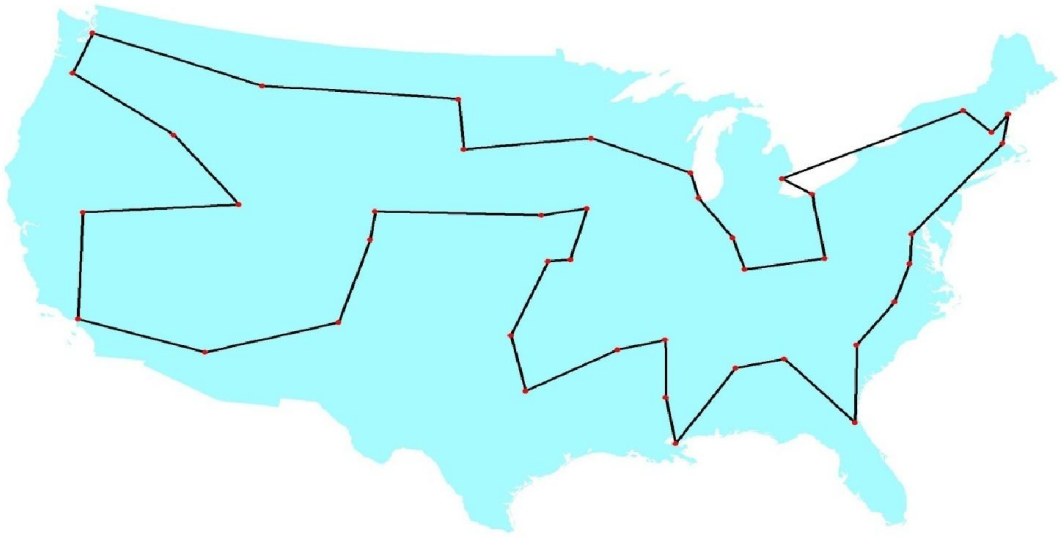


Figure 2 : Toure optimale de 42 villes aux Etats-Unis.résolu par Dantzig, Fulkerson et Johnson en 1954.

4.3.Concours de Procter and Gamble

Proctor and Gamble ont organisé un concours en 1962. Le concours nécessitait de résoudre un TSP sur 33 villes spécifiées. Il y avait un lien entre de nombreuses personnes qui ont trouvé l'optimum. Un chercheur précoce TSP, le professeur Gerald Thompson de l'Université Carnegie Mellon, a été l'un des gagnants.

Figure 3 le concours de Procter and Gamble en 1962

4.4.120 villes d'ouest allemand

Groetschel (1977) a trouvé le circuit optimal de 120 villes d'Allemagne de l'Ouest.

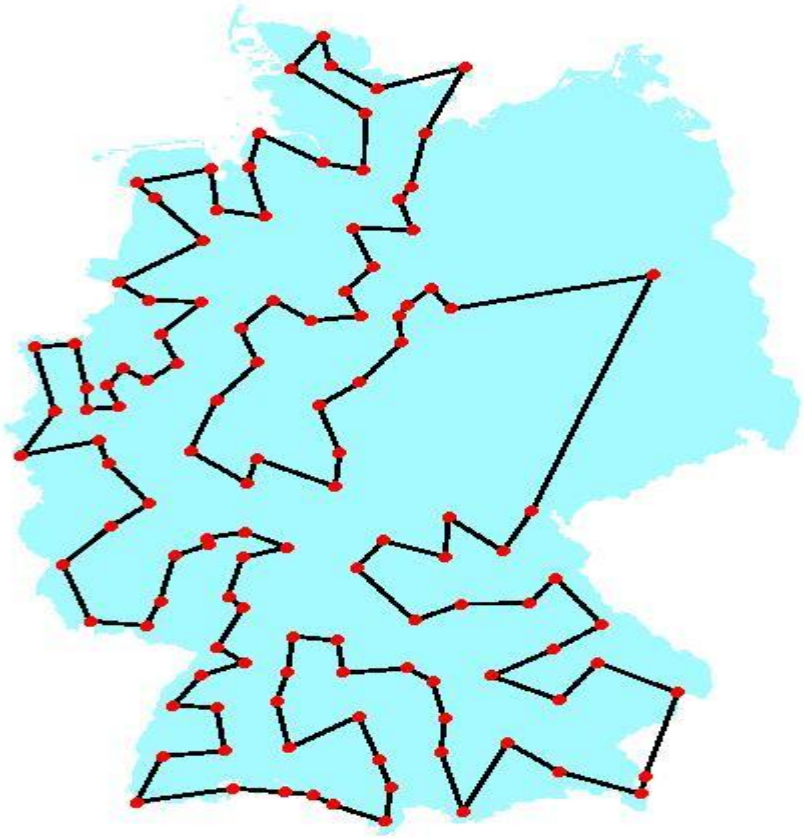


Figure 4 le tour optimale de 120 villes d'ouest allemandes

4.5. 532 emplacements en USA

Padberg et Rinaldi (1987) ont trouvé le tour optimal de 532 emplacements de switch AT&T aux États-Unis.

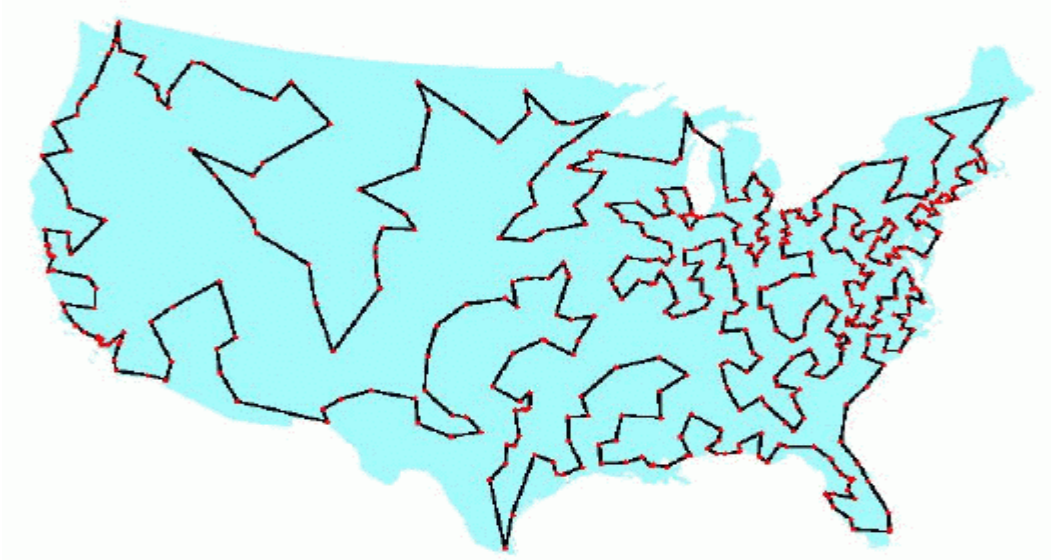


Figure 5 : l'optimum tour de 532 emplacements en USA Padberg et Rinaldi (1987)

4.6. 666 villes dans le monde

Groetschel et Holland (1987) ont trouvé le tour optimal de 666 endroits intéressants dans le monde.

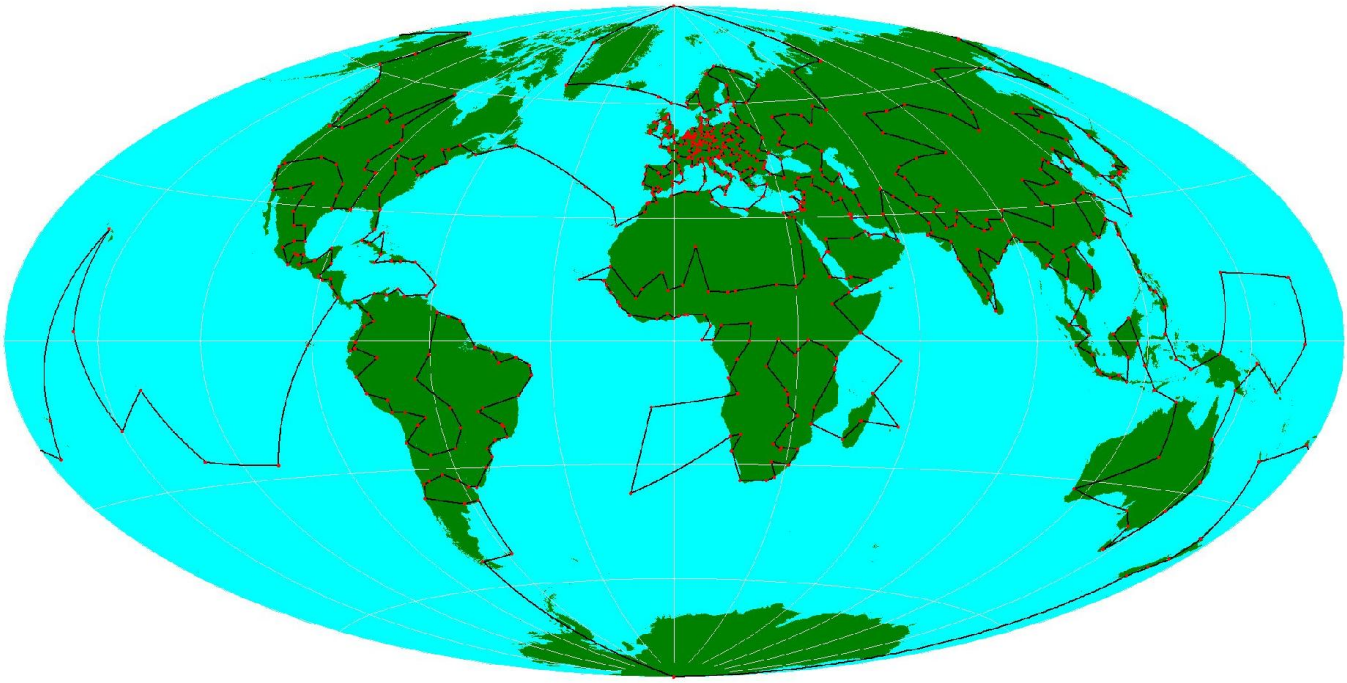


Figure 6 : le tour optimal de 666 endroits par Groetschel et Holland (1987)

4.7.2392 points

Padberg et Rinaldi (1987) ont trouvé le circuit optimal grâce à une disposition de 2 392 points obtenue auprès de Tektronics Incorporâtes

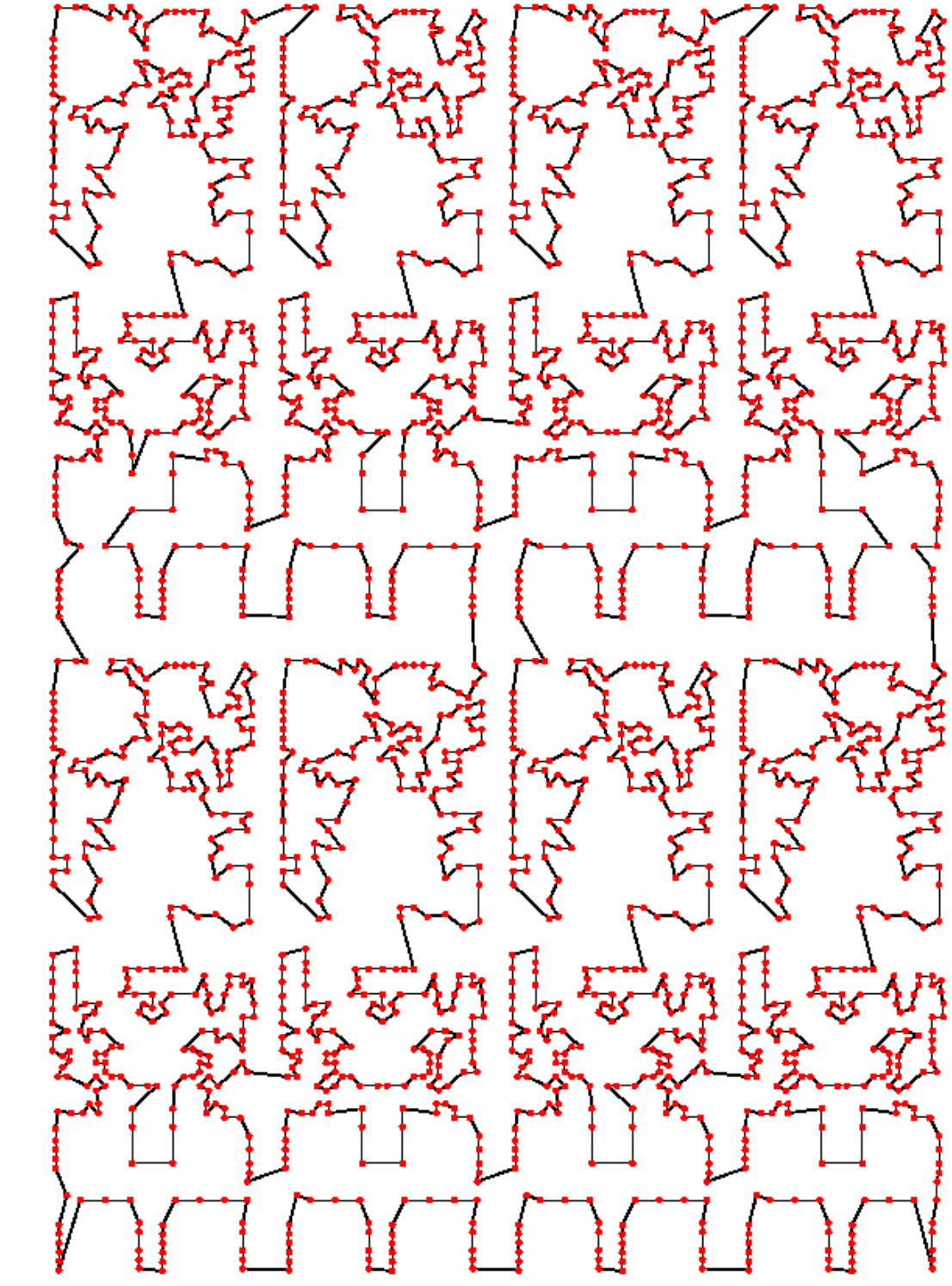


Figure 7 : le meilleure tour de 2 392 points par Padberg et Rinaldi (1987)

4.8.7397 TSP villes

Applegate, Bixby, Chvátal et Cook (1994) ont trouvé le circuit optimal pour un TSP de 7 397 villes qui a surgi dans une application de réseau logique programmable aux laboratoires de AT&T Bell.

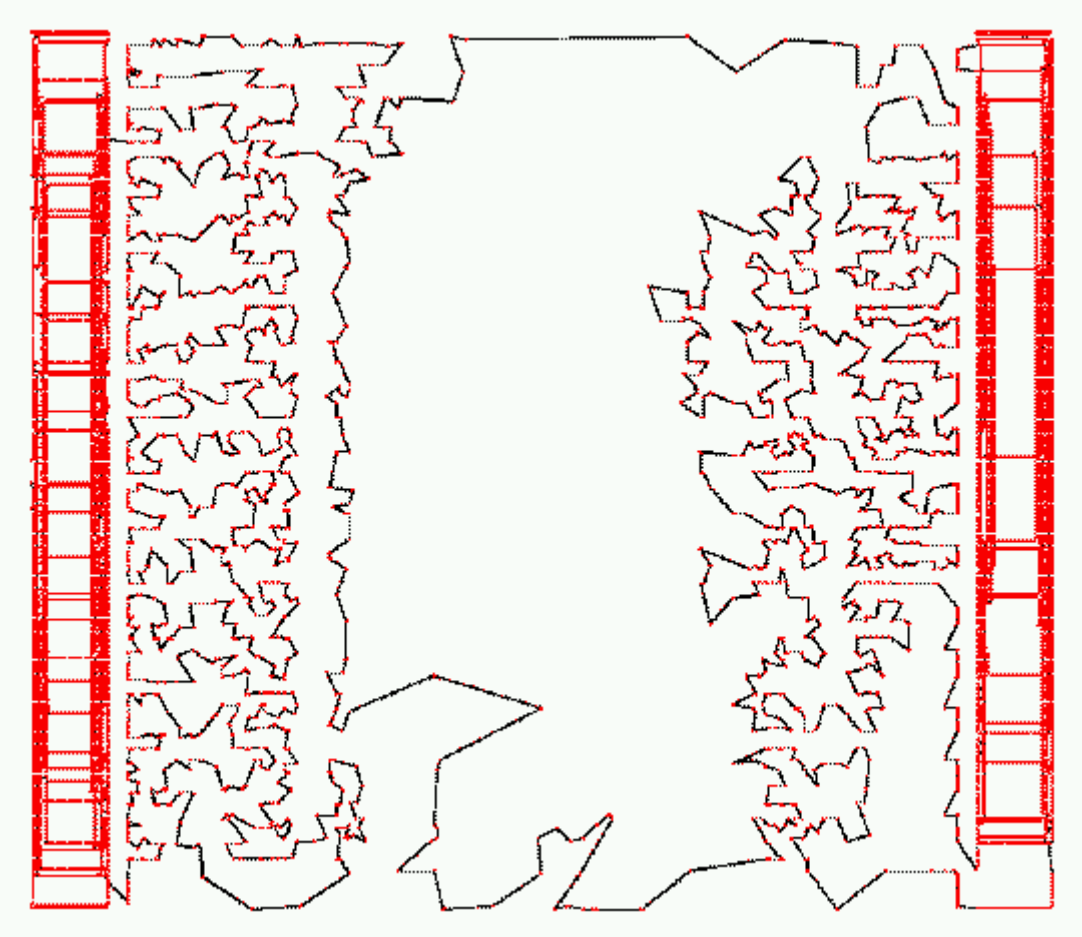


Figure 8 : circuit optimal pour un TSP de 7 397 villes par Applegate, Bixby, Chvátal et Cook (1994)

4.9. 13509 villes en USA

Applegate, Bixby, Chvátal et Cook (1994) ont trouvé le circuit optimal pour un TSP de 7 397 villes dans les Etats-Unis avec la population de plus que 500.

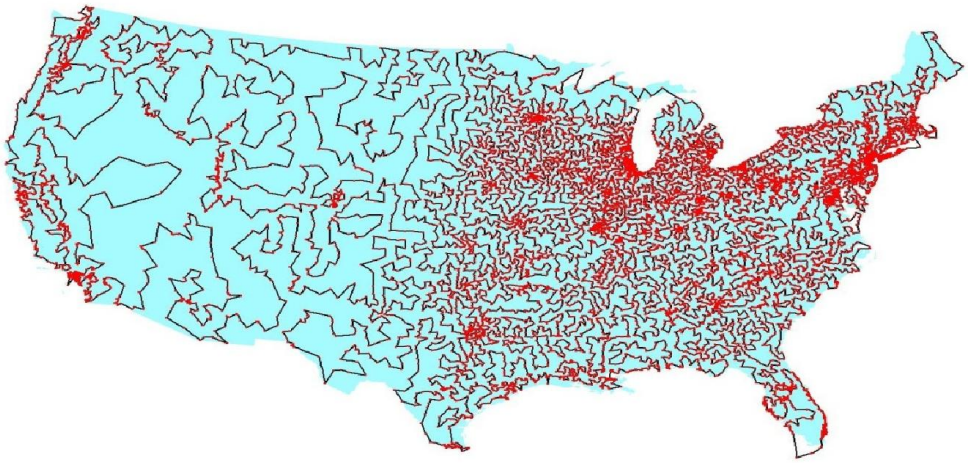


Figure 9 : le circuit optimal pour un TSP de 7 397 villes

4.10. 15112 villes en Allemagne

Applegate, Bixby, Chvátal et Cook (2001) ont trouvé le circuit optimal pour un TSP de 15112 villes en Allemagne.

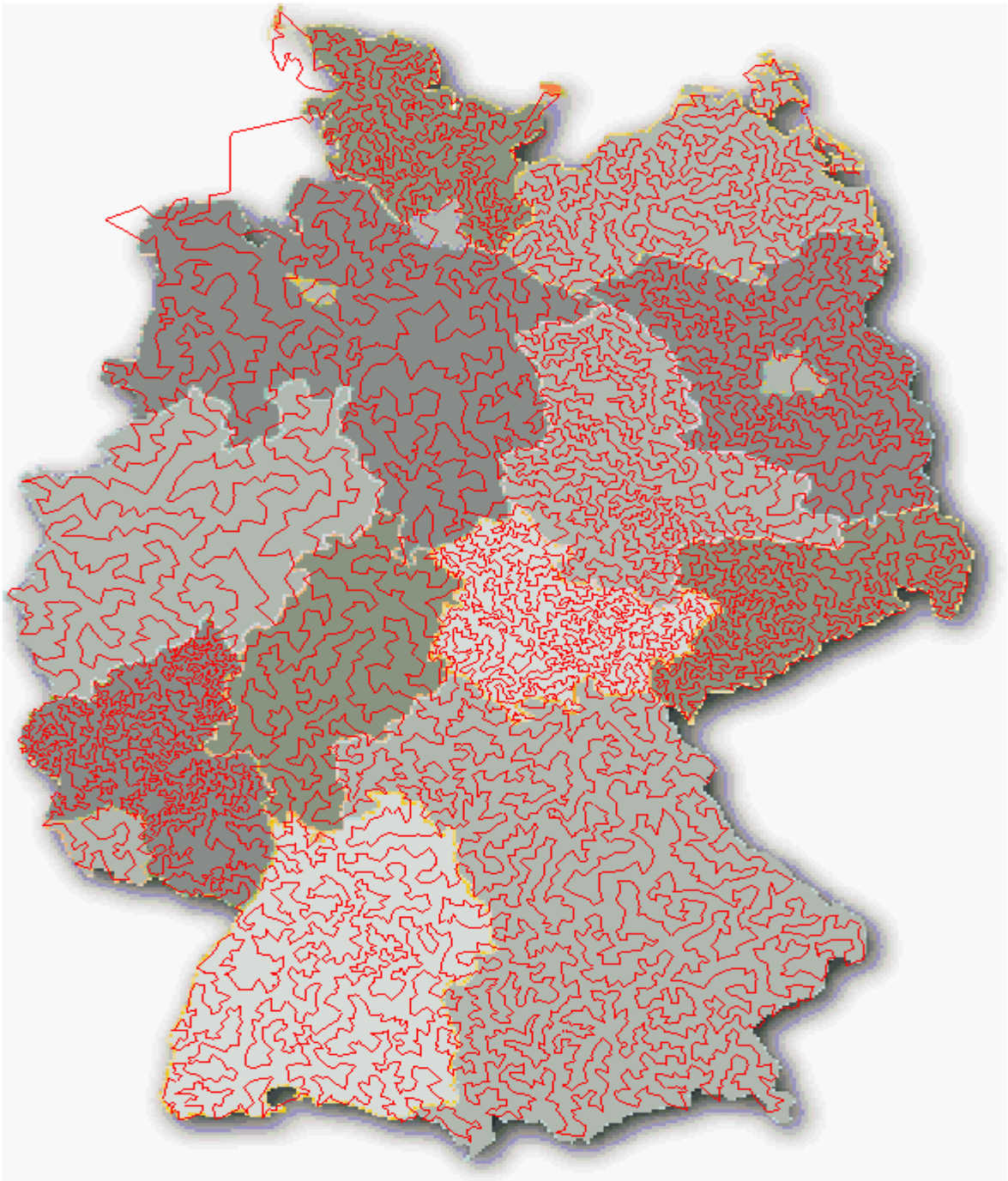


Figure 10 : l'optimum tour pour 15112 villes en Allemagne.

4.11. 24978 villes en suede

Applegate, Bixby, Chvátal, Cook, et Helsgaun (2004) ont trouvé le tour optimale de 24978 villes en suède.

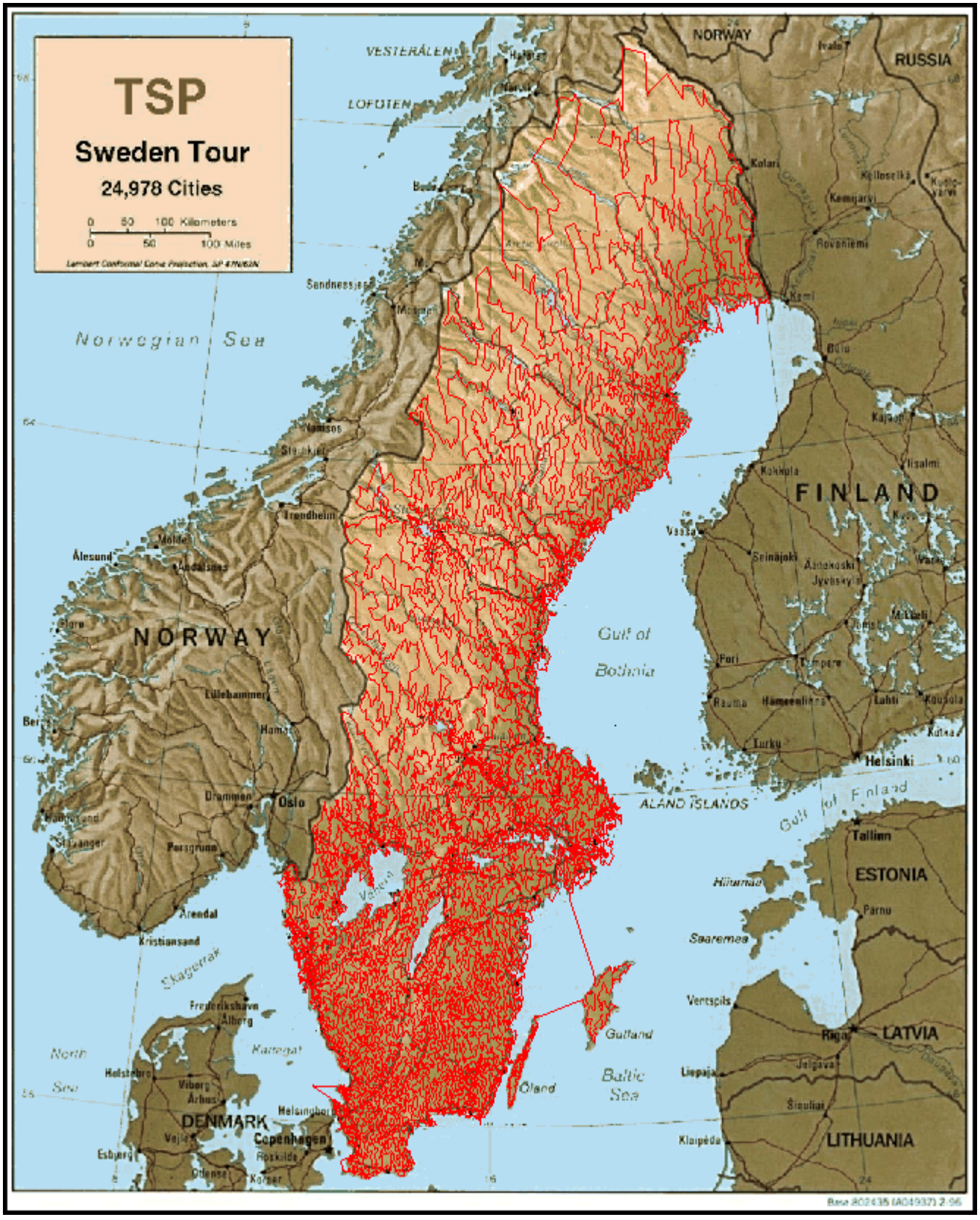


Figure 11 : l'optimum tour de 24978 villes en suède

4.12. conclusion

Plusieurs recherches ont été faites sur le problème de voyageur de commerce à pour but de trouver le meilleur tour pas seulement pour les villes mais pour des différents points telle que les circuits électroniques.

CHAPITRE3

Méthode OCA pour le PVC appliqué au réseau routier algérien

3.7 Introduction

L'un des essaims les plus organisés et les plus rigoureux dans leur vie sont ceux des abeilles et Comme on a vue dans le premier chapitre la méthode de Colonie d'abeille est l'une des méthodes de résolution de problème de voyageur de commerce. Dans ce chapitre on va parler en détaille sur cette méthode et l'application de celle-ci sur le réseau routier algérien.

3.8 La méthode de Colonie d'abeille

Dans cette méthode, les abeilles artificielles représentent des agents qui en collaborant les unes avec les autres, pour résoudre le problème de PVC, Historiquement l'algorithme HONEY-BEE a été réalisé pour la première fois vers 2004 par CRAIG A.TOVEY à GEORGIA TECH en collaboration avec SUNIL NAKRANI par la suite au début de l'année 2005, XIN-SHE YANG à l'Université de CAMBRIDGE a développé le VIRTUAL BEE ALGORITHM (VBA) pour résoudre des problèmes d'optimisation numérique, Un peu plus tard en 2005, HADDAD, AFSHAR et leurs collègues ont présenté un algorithme dit HONEY-BEE MATING OPTIMIZATION (HBMO) qui a ensuite été appliqué à la modélisation de réservoirs et de clustering et en 2006, B.BASTURK et D.JARABOGO en Turquie, ont développé un algorithme appelé ARTIFICIAL BEE COLONY (ABC) pour l'optimisation de fonctions numériques (Allaoua, 2020).

3.8.1 Principe de la méthode

Dans cette méthode l'emplacement da la source de nourriture représente la solution possible au problème et la quantité du nectar de cette source correspond a une valeur de fitness, dans la première étape la méthode génère une population initiale de n solutions distribués d'une façon aléatoire et chaque solution x_i (i=1,2,….,n) est initialisée par les scouts ou bien les éclaireuses et représente un vecteur, après l'initialisations des processus de recherches faits par les butineuses actives et inactives et les éclaireuses et les butineuses actives recherchent dans le voisinage de la source précédente x_i de nouvelle source v_i ayant plus de nectar, et calculent leur fitness. Si la nouvelle source contient plus de nectar de la source précédente celle-ci est remplacée par la nouvelle sinon l'ancien est conservée.

Les abeilles évaluent ces informations tirées de toutes les butineuses actives, et choisissent les sources de nourriture en fonction de la valeur de probabilité P_i associé à cette source dont :

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}$$

Ou fit_i est la fitness de la solution *i*, qui est proportionnelle à la quantité du nectar de la source de nourriture de la position *i*.

La source de nourriture dont le nectar est abandonné par les abeilles, les éclaireuses la remplacent par une nouvelle source.

3.8.2 L'algorithme de colonie d'abeille

Le pseudo code de l'algorithme basic de colonie d'abeille est défini par -Masaryk University, Brno, Czech Republic , Wed 08 Apr 2009 – comme suit (Allaoua, 2020):

1. Initialise population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met)
//Forming new population.
4. Select sites for neighbourhood search.
5. Recruit bees for selected sites (more bees for best **e** sites) and evaluate fitnesses.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and evaluate their fitnesses.
8. End While.

L'algorithme nécessite un nombre de paramètres :

- Nombre des abeilles éclaireuses n
- Nombre de sites sélectionné m de n sites visité.
- Nombre de meilleurs sites e de m sites sélectionné.
- Nombre des abeilles recrutée pour les meilleurs sites e (nep ou n2)
- Nombre des abeilles recrutées pour les autres sites (m-e) sélectionnée qui est nsp or (n1)
- Initialise ngh qui inclut les sites et leur voisinage et le critère d'arrêt.
- Nombre de répétitions des étapes de l'algorithme imax.

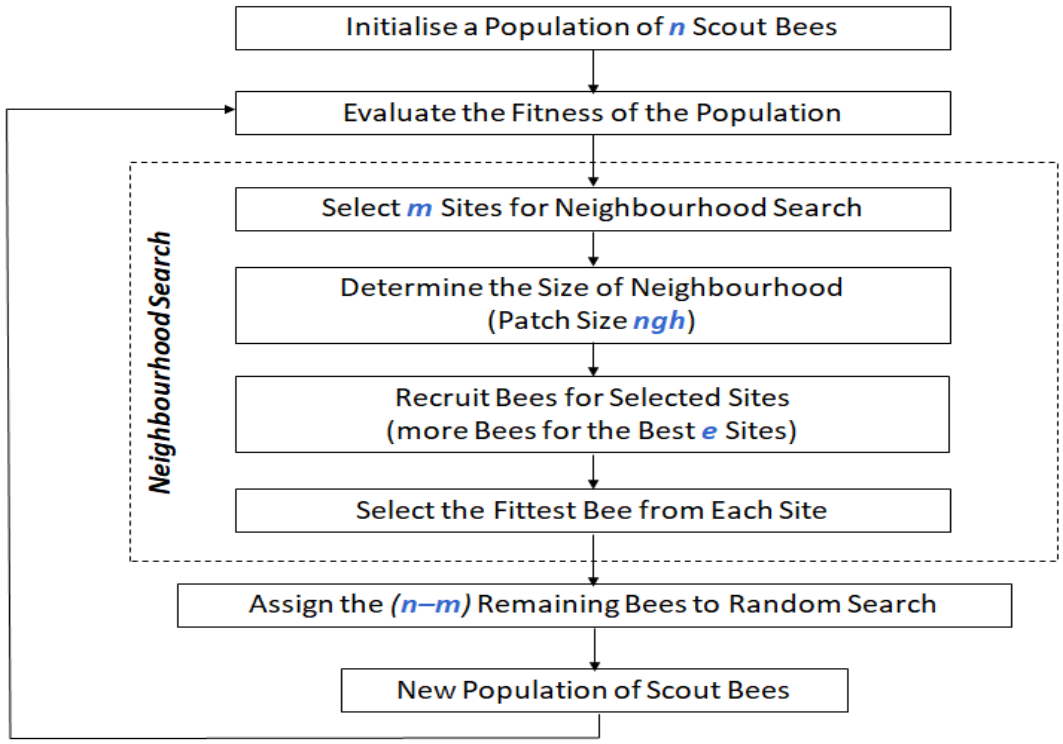


Figure 1 : Flowchart du basic algorithme de colonie d'abeille (Allaoua, 2020)

3.9 L'algorithme de MCA pour le PVC

L'algorithme détaillé pour résoudre le PVC est écrit comme suit :

BCOTSP (int k , int [][] d) // k nombre de villes , d matrice kxk des distances entre les villes : données

{

// Paramètres à ajuster

Int n; // taille de la population en fonction de k

Int m = n ; // nombre total de butineuses ; sources sélectionnées en fonction de k

Int e= m ; // nombre de butineuses pour les sources riches en nectar en fonction de k

Int n1 ; // scouts (taille du voisinage) pour les sources riches un constant

Int n2 ; // scouts (taille du voisinage) pour les sources moins riches un constant dont n1>n2

Maxiter ; // nombre d'itérations de l'algorithme en fonction de k

//----------//

//neighbor : retourne un voisin X (une solution , un tableau) aléatoire de la solution d'indice i dans la

//population en permutant deux villes aléatoires

int [] neighbor (int i) {

X[]=pop[i] ;

Ind1 = int(k*rand()) ;

Ind2 = int(k*rand()) ;

```
Tmp= X[ind1] ;
X[ind1]= X[ind2] ;
X[ind2]=Tmp ;
return X ;
//-------------------------// Initialisation, création de la population initiale pop, une matrice nxk
{ for(i=0 ; i<n ; i++)
for(j =0 ; j<k ; j++)
{do
ind = Int(k * Rand())
while (pop[i][ind] != 0 )  // le tour (le cycle) doit être hamiltonien
pop [I][ind]= j :}}
//-----------------------------------------------------------------------------
// Calcul de la longueur de la solution d'indice i dans la population pop
int longueur(int i) {
int s=0:
for(j=0 ; j<n-1 ; j++)
s+=d[pop[i][j]] [pop[i][j+1]]
return s+ d[pop[i][n-1]] [pop[i][0]]  }
//-------------------------------------------------
// Évaluation des solutions de la population, calcul de leurs fitness, leurs longueurs
{  int Sumfitness =0 ;  // à utiliser dans la roulette de sélection
for(i=0 ; i<n ; i++)
{ Fitness[i] = longueur(i) ;  // à utiliser dans la roulette de sélection
Sumfitness += Fitness[i] ;
If (fitness[i] < lopt)
{xopt= pop[i] ; lopt= fitness[i] ;} // Mise à jour de la meilleure solution trouvée jusqu'ici
//-------------------------------------------------
// Fonction roulette pour simuler la roue de la fortune dans la sélection des solutions
int roulette ()
{  float s = 0 ;
float s1 = sumfitness*rand() ; // arrester la roulette
Int j =0 ;
while (s < s1)
{ s+=fitness[j] ; j++;}  // tourner la roulette
```

```
return j ; }
//-------------------------------------------------
// Programme principal
Lopt= infini ; xopt= null ;  // solution (cycle) optimale recherchée et sa longueur
for (iter = 0; iter < maxiter; iter ++)    // boucle principale
// On construit une nouvelle population pop1 à partir de pop en 3 étapes
{
// Première étape : les sources riches
for(i=0 ; i<e ; i++)
{  r = roulette() ;
x= pop[r] ;
for (j=0 ; i<n1 ; i++)  // calcul de n1 voisins de la solution pop[i]
{  b= neighbor(r) ;  // calcul d'un voisin aléatoire de la solution courante sélectionnée d'indice r
if (longueur(b) < longueur(x))  // recherche du meilleur voisin
x = pop[b] ;  }
pop1[i]= x ;  }  // prise du meilleur voisin
}  // fin de la première étape
//-----------------------------------------------------------------
// Les sources moins riches en nectar
for(i=e ; i<m ; i++)
{  r = roulette() ;
x= pop[r] ;
for (j=0 ; i<n2 ; i++)  // calcul de n2 voisins de la solution pop[i]
{ b= neighbor(r); // calcul d'un voisin aléatoire de la solution courante sélectionnée d'indice r
if (longueur(b) < longueur(x))  // recherche du meilleur voisin
x = pop[b] ;  }
pop1[i]= x ;  }  // prise du meilleur voisin
}  // fin de la deuxième étape
//-------------------------------------------------------------------------------------------
// On remplace les n-m solutions restantes par de nouvelles solutions aléatoires
for(i=m ; i<=n-1 ; i++)
for(j =0 ; j<k ; j++)
{do
ind = Int(k * Rand())
```

```
while (pop1(I,ind) !=0)
pop1 [i][ind]= j}  // fin de la troisième étape
//-------------------------------------------------------------------------------
pop = pop1 ;  //  la population courante dévient pop1
} // fin de la boucle principale
printf( xopt , fopt) ]  }
```

3.10 Les instances de PVC

Pour appliquer l'algorithme de colonie d'abeille sur le problème de PVC on a besoin des données comme des entrées avec les paramètres de l'algorithme à ajuster, Pour le PVC les données ce sont les distances entre les villes qui sont organisée dans une matrice appelée la matrice des distances, ces données sont organisé dans des fichiers .tsp qui sont des instances pour le problème de PVC. Les instances sont téléchargées depuis le site https://github.com/coin-or/jorlib/tree/master/jorlib-core/src/test/resources/tspLib/tsp.

3.11 Les données pour le réseau routier algérien

Pour les villes algériennes ou plutôt les 48 wilayas algériennes, nous avons utilisées Google Maps pour calculer les distances entre les villes, par la suite on a crée un fichier Excel et convertir-le en fichier txt puis en fichier .tsp en ajoutant l'entête du fichier.

```
NAME: alg48
TYPE: TSP
COMMENT: 48 cities in Algeria, street distances (google Maps distances)
DIMENSION: 48
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: FULL_MATRIX
DISPLAY_DATA_TYPE: TWOD_DISPLAY
EDGE_WEIGHT_SECTION
0 1380 993 1518 1398 1463 1265 561 1361 1368 1075 1513 1118 1209
 ……
1829 453 331 884 1457 140 768 702 829 253 662 156 333 204 578 0
EOF
```

3.12 Conclusion

La méthode MCA est simple à utilisé pour résoudre le PVC et elle donne un bon résultat par l'ajustement de ces paramétrés, elle est basée sur le concept de coopération qui rend les abeilles plus efficaces et ainsi arrivées à leurs buts rapidement.

Chapitre 4
Résultats et discussion

4.1 Introduction

Dans ce chapitre on va présenter notre implémentation de MCA pour le PVC et le langage à utilisée avec des discussions sur les résultats obtenu

4.2 Java

java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au *SunWorld*.

La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java (Wikipédia).

Java est un langage de programmation d'utilisation simple très proche de C++ (la syntaxe représente une version améliorée de C++).

Par la suite on a utilsé un environnement de développement intégré (IDE) spécifique pour la programmation en Java qui est JCreator 5LE , qui englobe sur la même application **:** éditeur, débogueur et compilateur, il est écrit uniquement en C++ et consomme moins de ressources.

4.3 Report des résultats

Sur un ordinateur HP avec un processeur Intel® Core™ i7-3770 CPU @ 3.40 Ghz et une RAM de 8 Go sur un système d'exploitation Windows 7 on a les résultats qui figure dans le tableau suivant :

La stratégie de test adoptée pour la méthode MCA pour le PVC porte sur plusieurs paramètres tel que la taille de la population, nombre totale des butineuses, nombre de butineuses pour les sources riches, les éclaireuses pour les zones riches et celle pour les zones moins riches et le nombre d'itération maximale

Taille du population	nombre totale des butineuses	nombre de butineuses pour les sources riches	scouts pour les zones riches n1	scouts pour les zones moins richee	nombre d'itération max maxiter	Longueur de solution	Temps d'execution
480	240	144	15	7	10	25568	0.112
480	240	144	15	7	100	24704	0.943
480	240	144	15	7	1000	23723	7.091
480	240	144	15	7	10000	22819	68.316
480	240	144	15	7	100000	22085	724.299

Tableau 1 des résultats

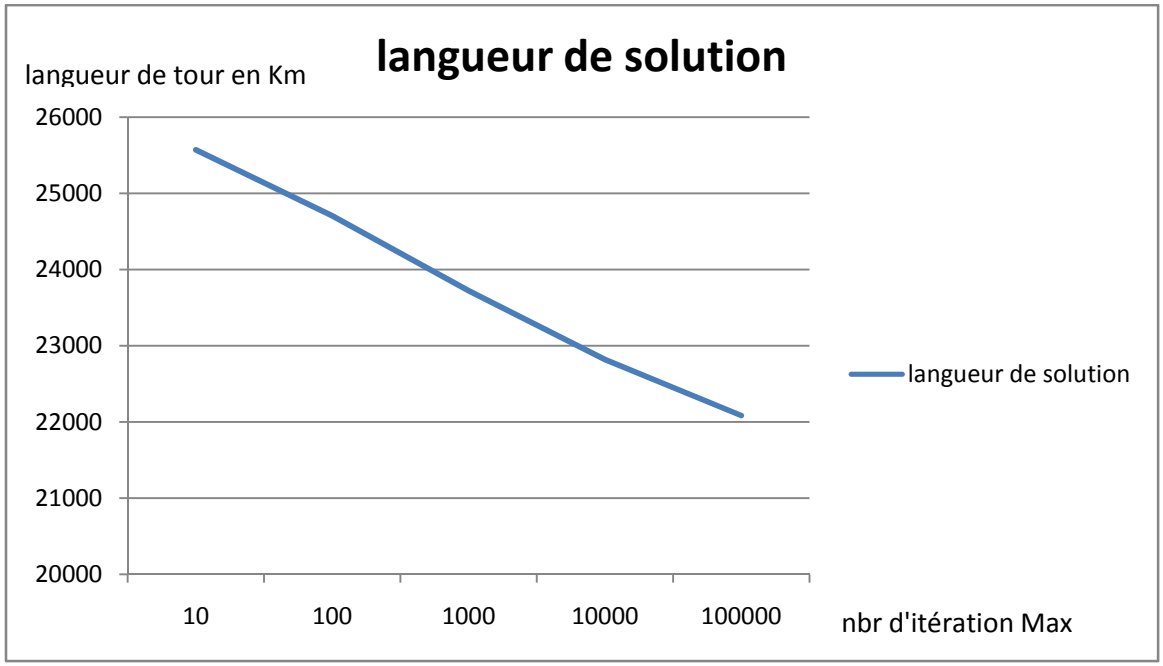


Figure 1 Courbe représente la langueur de solution par apport de nombre d'itérations

D'après la courbe on remarque que la langueur de solution est inversement proportionnelle avec le nombre d'itération.

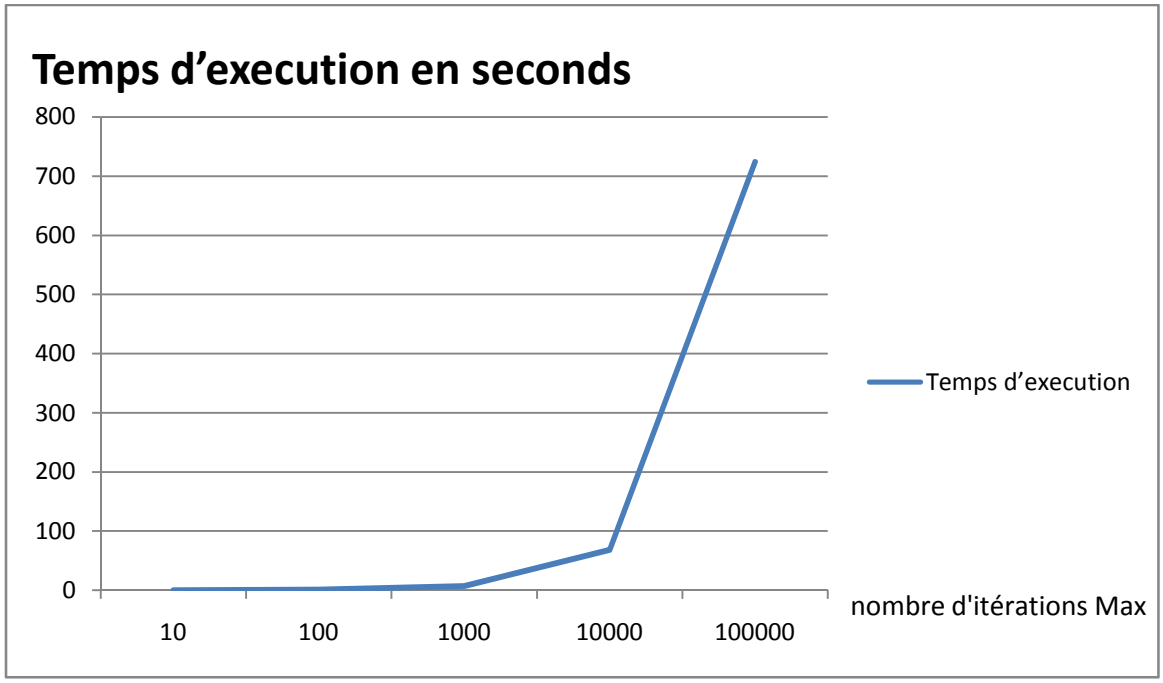


Figure 2 Courbe représente le temps d'exécution par apport de nombre d'itérations

Par contre le temps d'exécutions se croitre par apport le nombre d'itérations.

Conclusion générale

Conclusion générale

Le problème de voyageur de commerce est un problème d'optimisation combinatoire qui est simple à résoudre par des méthodes exactes si le nombre de villes est petit, mais si le nombre de villes est supérieure à dix le problème soit plus difficile à résoudre ou bien nécessite un très longtemps peu aller jusqu'à des années de calcule.

Par l'utilisation de la méthode de colonie d'abeille on peu réduire le temps pour obtenir une solution n'est pas exacte mais plus proche à la solution exacte et dans un temps raisonnable.

Plusieurs recherches dans ce domaine à été fait pour avoir une solution plus proche à la solution exacte dans plusieurs pays, mais pas dans l'Algérie, normalement c'est la première fois que ce problème a été posé sur le réseau routier algérien.

Cependant, il est évident que plusieurs améliorations restent à faire que se soit sur le coté de l'algorithme pour minimiser la langueur de cycle entre les villes en conservant le temps d'exécution pour que soit minimale aussi, ou de coté des donnée en ajoutant les 10 wilayas crée dernièrement.

Références bibliographiques

Bibliographie

ANDRÉ BIANCHERI TCHILINGUIRIAN Résolution du Problème du Voyageur de Commerce -Métaheuristique-. - 2006.

BEKADA Karima Amina Sur Une Méthode de Résolution d'un Problème d'Optimisation Combinatoire // Sur Une Méthode de Résolution d'un Problème d'Optimisation Combinatoire. - MOSTAGANEM : UNIVERSITÉ ABDELHAMID IBN BADIS-MOSTAGANEM, 2015. - pp. 8, 9;10;11.

CHENTLI Hayet MODÈLES ET MÉTHODES POUR UNE CLASSE DE PROBLÈMES DE RAMASSAGE ET LIVRAISON // MODÈLES ET MÉTHODES POUR UNE CLASSE DE PROBLÈMES DE RAMASSAGE ET LIVRAISON. - alger : Université des Sciences et de la Technologie Houari Boumediene, 2018. - p. 12.

KOUIDER AIAD Soumia VERS UN CRYPTOSYSTEME BASE SUR LE PROBLEME DU VOYAGEUR DE COMMERCE // VERS UN CRYPTOSYSTEME BASE SUR LE PROBLEME DU VOYAGEUR DE COMMERCE. - Alger : [s.n.], 2009. - p. 94.

Little J. D, Murty, K. G, Sweeney, D. W., & Kar Operations Research [Book]. - [s.l.] : INFORMS, 1963. - Vol. 11 : p. 989.

MUNTEANU Gheorghe MODÉLISATION MATHÉMATIQUE, SIMULATION ET OPTIMISATION DES RÉSEAUX DE TRANSPORT. - MONTRÉAL : UNIVERSITÉ DU QUÉBEC , 2009. - p. 8.

Stiven Garry
https://www.academia.edu/8050033/INTRODUCTION_AUX_PROBLEMES_COMBINATOIRES_DIFFICILES_ [Online] = INTRODUCTION AUX PROBLEMES COMBINATOIRES "DIFFICILES " // https://www.academia.edu.

TetiBernard Tadunfock and Pauline Fotso Laure heuristiques du problème du voyageur de commerce, CARI06 [Article] // heuristiques du problème du voyageur de commerce. - 2006. - CARI06 : Vol. 1.

Zambito Leonardo The Traveling Salesman Problem: A Comprehensive Survey // The Traveling Salesman Problem: A Comprehensive Survey. - 2006.

Allaoua, H. (2020). cours méthodes d'optimisation.