UNIVERSITÉ ABDELHAMID IBN BADIS-MOSTAGANEM
FACULTÉ DES SCIENCES EXACTES ET L'INFORMATIQUE
DÉPARTEMENT DE MATHÉMATIQUES

Mémoire de Master

Spécialité : Modélisation, Contrôle et Optimisation

Thème

Sur Une Méthode de Résolution d'un Problème d'Optimisation Combinatoire

Présenté par

M^{elle}. BEKADA Karima Amina

Soutenu le 26 /05/2015

Devant le jury

Président	MECHDENE.M	**U. MOSTAGANEM.**
Examinateur	BELGACEM.R	**U. CHLEF.**
Encadreur	ABLAOUI.H	**U. MOSTAGANEM.**

Table des matières

INTRODUCTION

Ce mémoire aborde le thème de la résolution du problème du voyageur de commerce par des heuristiques.Ce dernier a acquis une importance capitale face à l'évolution de la société et à la prise en charge de ses besoins. Il s'agit d'un problème mathématique qui consiste à trouver un plus court chemin qui relie l'ensemble des villes séparées par des distances données. C'est un problème d'optimisation pour lequel on ne connait pas d'algorithme permettant de trouver une solution exacte en un temps "raisonnable".

Devant les contraintes rencontées lors de l'application des méthodes exactes, certains mathématiciens du $19^{ème}$ siècle tels que Dorigo,Monizzo[1],Coes[9],Vercchi,Cerny[10],Deneubourg [4],..,etc, ont proposé des méthodes approchées (heuristiques), donnant des résultats approximatifs avec des erreurs acceptables, qui se sont imposées comme solutions alternatives.

Le travail exposé ci-dessous est organisé de la manière suivante :

Au premier chapitre, nous donnons quelques rappels sur la théorie des graphes, au deuxième chapitre nous introduisons le problème du voyageur de commerce, le troisième chapitre est consacré à la description de quelques méthodes de résolution puis au dernier chapitre nous présentons quelques tests numériques.

Nous terminons notre mémoire par une conclusion et quelques références bibliographiques.

Chapitre 1

Notions sur les graphes

1.1 Graphes non orientés

Un graphe $G = (V, E)$ est défini par la donnée :

- d'un ensemble fini $V = \{v_1, v_2, ..., v_n\}$ dont les éléments sont appelés sommets

- D'un ensemble fini $E = \{e_1, e_2, ..., e_m\}$ dont les éléments sont appelés arêtes.

-D'une application e qui associe à chaque arête ses extrémités.

Si l'arête e relie les sommets a et b, on dira que ces sommets sont adjacents, ou incidents avec e, ou bien que l'arête e est incidente aux sommets a et b.

On appelle ordre d'un graphe le nombre n de sommets de ce graphe

1.1.1 Quelques types de graphes

Graphe simple

Un graphe est simple si au plus une arête relie deux sommets quelconques.

Graphe connexe

Un graphe est connexe s'il est possible, à partir de n'importe quel sommet, de rejoindre tous les autres en suivant les arêtes.

Un graphe non connexe se décompose en composantes connexes.

Graphe complet

Un graphe est complet si chaque sommet du graphe est adjacent à tous les autres sommets.

Graphe partiel et sous-graphe

Soit $G = (V, E)$ un graphe. Le graphe $G' = (V, E')$ est un graphe partiel de G si E' est inclus dans E . Autrement dit, on obtient G' en enlevant une ou plusieurs arêtes au graphe G.

Pour un sous-ensemble A de sommets inclus dans V , le sous-graphe de G induit par A est le graphe $G = (A, E(A))$,dont l'ensemble des sommets est A et l'ensemble des arêtes $E(A)$ est formé de toutes les arêtes de G ayant leurs deux extrémités dans A. Autrement dit, on obtient G' en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.

Chaînes et cycles

Une chaîne dans G, est une suite ayant pour éléments alternativement des sommets et des arêtes, commençant et se terminant par un sommet, et telle que chaque arête est encadrée par ses extrémités

On dira que la chaîne relie le premier sommet de la suite au dernier sommet. On définit la longueur d'une chaîne par le nombre d'arêtes qui constitue cette chaîne.

un cycle c'est une chaine fermée.

Graphes hamiltoniens

- On appelle cycle hamiltonien d'un graphe G, un cycle passant une et une seule fois par chacun des sommets de G .
- Un graphe est dit hamiltonien s'il possède un cycle hamiltonien.
- On appelle chaîne hamiltonienne d'un graphe G une chaîne passant une et une seule fois par chacun des sommets de G.

1.2 Graphes orientés

En donnant un sens aux arêtes d'un graphe, on obtient un graphe orienté.

Un graphe orienté G est défini par la donnée :

- D'un ensemble fini X (dont les éléments sont appelés sommets).
- D'un ensemble fini U (dont les éléments sont appelés arcs).

– D'une application $I : U \rightarrow X$

$$u \rightarrow I(u) = \text{extremité initiale de u}$$

– D'une application $T : U \rightarrow X$

$$u \rightarrow T(u) = \text{l'extremité terminale de u}$$

1.2.1 Chemins et circuits

Un chemin c du sommet a au sommet b est une séquance d'arêtes ($u_1, u_2, ..., u_p$) telle que pour tout $i = 1...p-1$,$T(u_i) = I(u_{i+1})$, $I(u_1) = a$ et $T(u_p) = b$.

Un circuit est un chemin où les extremités sont confondues.

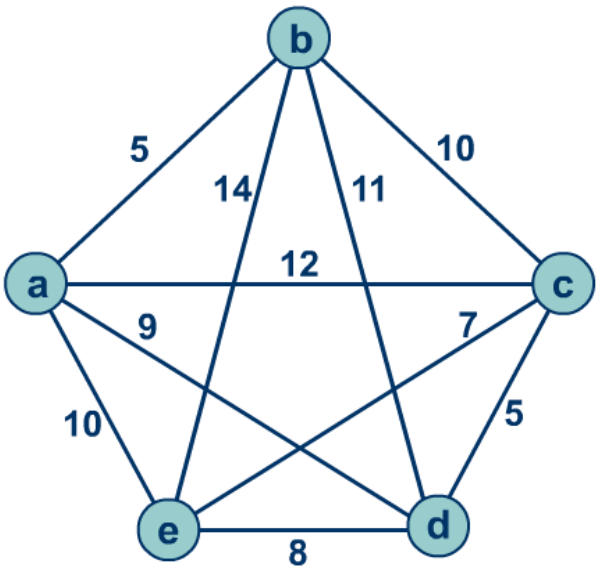
1.2.2 Arbres

On appelle arbre tout graphe connexe et sans cycles.

Un graphe sans cycles mais non connexe est appelé forêt

Exemple

Exemple 1.2.1 *La figure ci-dessous représente un graphe complet formé de 5 sommets et 10 arrêtes.*



Chapitre 2

Le problème du voyageur de commerce

Le problème du voyageur de commerce(en anglais travelling salesman problem :TSP), a étudié pour la première fois durant le $19^{ème}$ siècle. Il est l'un des problèmes les plus connus dans le domaine de la recherche opérationnelle. Jouer à trouver le meilleur parcours possible... et découvrez différentes méthodes mathématiques et informatiques proposées pour résoudre ce problème.

C'est déjà sous forme de jeu que William Rowan Hamilton a posé pour la première fois ce problème, dès 1859. Sous sa forme la plus classique, son énoncé est le suivant : Un représentant de commerce veut vendre sa marchandise dans un certain nombre de villes ,il doit donc planifier sa tournée de mannière à passer par toutes les villes une et une seule fois. Devant une telle situation fictive, le voyageur doit prévoir un trajet optimal à l'avance.

Les domaines d'application sont nombreux : problèmes de logistique, de transport aussi bien de marchandises que de personnes, et plus largement toutes sortes de problèmes d'ordonnancement. Certains problèmes rencontrés dans l'industrie se modélisent sous la forme d'un problème de voyageur de commerce, comme l'optimisation de trajectoires de machines outils : comment percer plusieurs points sur une carte électronique le plus vite possible ?

Ce problème classique d'optimisation combinatoire, appartient à la classe des problèmes dite NP-COMPLET , est un problème pour lequel on ne connait pas d'algorithmes permettant de trouver une solution exacte en un temps polynomial (un temps raisonnable).

Comme tout problème d'optimisation combinatoire, nous avons deux approches de résolution du problème :

– Les méthodes exactes

– Les méthodes approchées ou heuristiques permettent l'obtention de solutions approchées.

Définition 2.0.1 *Une heuristique est une technique qui améliore l'efficacité d'un processus de recherche, en sacrifiant éventuellement l'exactitude ou l'optimalité de la solution.*

Pour des problèmes d'optimisation (NPcomplets) où la recherche d'une solution exacte (optimale) est difficile (coût exponentiel), on peut se contenter d'une solution approchée donnée par une heuristique avec un coût moindre.Certaines heuristiques sont polyvalentes (elles donnent d'assez bons résultats pour une large gamme de problèmes) alors que d'autres sont spécifiques à chaque type de problème.

On distingue trois classes d'heuristiques :

1. Les heuristiques de construction qui élaborent graduellement la tournée en ajoutant une ville (noeud) à chaque étape. On arrête le processus dès qu'une solution est trouvée suivant un critère d'arret défini au préalable. Dans cette catégorie, il y a l'heuristique du plus proche voisin, l'algorithme glouton, l'algorithme de Christofides...etc.
2. Les heuristiques d'amélioration qui consistent, une fois qu'une tournée est générée par une heuristique de construction, à l'améliorer pour obtenir une tournée de qualité meilleure.Les algorithmes de recherche locale 2-opt et 3-opt sont des exemples les plus communément utilisés.Nous avons aussi l'algorithme de Lin-Kernighan, la recherche tabou .
3. les méta-heuristiques sont des heuristiques qui s'inspirent des problèmes réels (Biologie, thermodynamique,..etc) telles l'algorithme de colonie de fourmis(qui s'adapte très bien au problème du TSP), l'algorithme génétique, le recuit simulé...etc.

2.1 Formulation mathématique du problème

Soit $G = (V, E)$ un graphe non orienté que l'on peut supposer, sans perte de généralité, simple, complet et sans boucles.

A toute arête e de E est associé un cout c_e (ou c_{ij} si $e = (i, j)$).

On définit : $x_{ij}=\begin{cases} 1 & \text{si le sommet } j \text{ est visité juste aprés } i \\ 0 & \text{sinon} \end{cases}$

On suppose que c_{ij} représente la plus courte distance du sommet i au sommet j (quite à appliquer un algorithme de recherche des plus courtes distances entre deux sommets tel que la méthode de Dijkstra)

$$\begin{cases} \min \sum_{(i,j)\in E} c_{ij}x_{ij} \\ \sum_{\substack{i\neq j \\ i\in v}} x_{ij} = 1 \quad \forall j \in V & \text{(entrer une seule fois dans chaque ville)} \\ \sum_{\substack{j\in V \\ j\neq i}} x_{ij} = 1 \quad \forall i \in V & \text{(sortir une seule fois de chaque ville)} \\ \sum_{\substack{i\neq j \\ i\in s}} x_{ji} \leq \mid s \mid -1 \ \ \forall\ s = \{2, ..., n-1\} & \text{(élimination des sous-tours)} \\ x_{ij} \text{ et } x_{ji} \in \{0,1\} \quad \forall (i,j) \in E \end{cases}$$

Chapitre 3

Méthodes de résolution

Parmi les méthodes de résolution du TSP, nous allons nous intéresser aux algorithmes de colonies de fourmis, à la méthode de 2-opt, à la technique du recuit simulé et à la méthode du plus proche voisin.

3.1 Algorithme de colonie de fourmis

Les algorithmes de colonies de fourmis sont inspirés du comportement de fourmis et constituent une famille de méta-heuristique.

3.1.1 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 aportées 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 subsances volatiles que les fourmis perçoivent grâce à des capteurs sur leurs antènnes, déposent sur le sol par une glande située sur leur abdomen en créant ainsi des pistes chimiques.
Nous avons reproduit des expériences pour mettre en évidence cette particularité

3.1.2 Expériences

Expérience 1

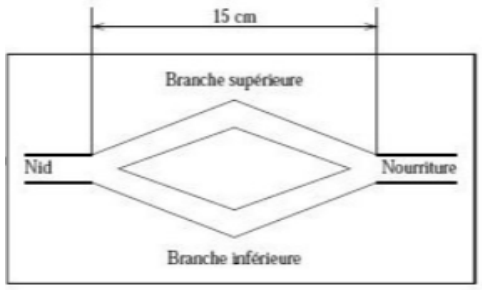
Protocole

1. Isoler des fourmis
2. Attendre un certain temps qu'elles se calment et reprennent un comportement naturel.
3. Construire des chemins reliant la sortie de la fourmilière à une source de nourriture.
4. Libirer les fourmis.

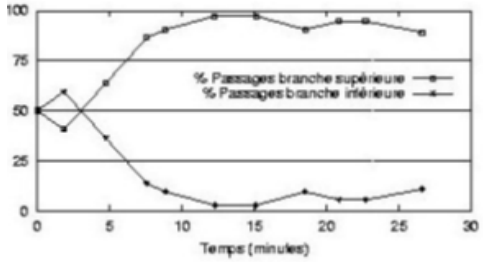
Observation Les fourmis se sont d'abord dispersées aléatoirement et, petit à petit, seuls les plus courts chemins ont été sélectionnés jusqu'à ce qu'il n'en reste qu'un parmi les plus optimaux.

Experience 2

Pont binaire de Deneubourg [4] La figure(3.1.1) repésente la configuration physique d'une éxpérience effectuée sur un nid d'une colonie de fourmis qu'un pont à deux voies de même longeur sépare d'une source de nourriture.On laisse les fourmis circuler librement sur ce pont.
La figure(3.1.2) est un graphe qui explique le système du déplacement des fourmis en fonction du temps. Il informe sur le nombre d'insectes empruntant chaque branche.



(a) (3.1.1)



(b) (3.1.2)

Observation Un moment après,on remarque que les fourmis choisissent le méme itinéraire.

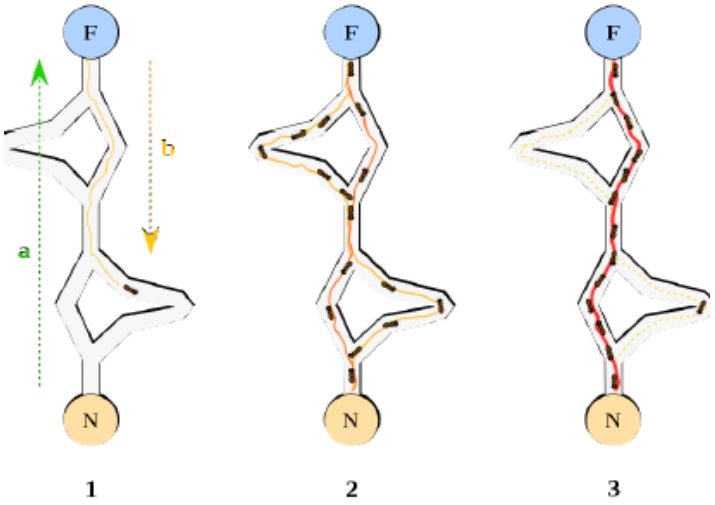
Explication Les fourmis déposent des phéromones en avançant,donc si elles sont plus nombreuses en haut, le chemin du haut
comportera plus de phéromones.

Expérience 3

Expérience du double pont binaire[5] Une seconde expérience représentée dans la figure(3.1.3) est réalisée sur le même nid de fourmis, elle consiste à augmenter la longeure d'une des deux branches du pont. On constate alors que les fourmis choisissent le chemin le plus court dans les deux sens pour revenir avec de la nourriture à la fourmilére.

Les phéromones sécrétés par les premiéres d'entre elles tracent une trajectoire qui attire sucessivement les autre fourmis contrairement au long chemin,qui lui, n'est marqué que dans le sens d'aller.

A partir de cette éxpérience, on déduit que le déplacement des fourmis dépend de deux mécanismes distincts : celui des pistes de phéromones et celui de la notion de distance.



(3.1.3)

3.1.3 Modèle explicatif

une fourmi parcourt plus ou moins au hasard l'environnement autour de la colonie.

Si celle-ci découvre une source de nourriture, elle entre directement au nid en laissant sur son chemin une piste de phéromones. Ces phéromones étant attractives, les fourmis passant à proximité vont avoir tendance à suivre, de façon directe, cette piste. En revenant à la fourmilère, ces mêmes fourmis vont renforcer la piste.

Si deux pistes sont possibles pour atteindre la même source de nourriture, celle étant la plus courte sera, dans le même temps, parcourue par plus de fourmis que la longue piste.

La piste courte sera donc de plus en plus renforcée, et donc de plus en plus attractive ; à terme, l'ensemble des fourmis a donc déterminé et « choisi » la piste la plus courte.

La longue piste, elle, finira par disparaître, les phéromones étant volatiles.

Les phéromones ont un rôle de marqueur de chemin : quand les fourmis choisissent leurs chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Cela leur permet de retrouver le chemin vers leur fourmilière lors du retour. D'autre part, les odeurs peuvent être utilisées par les autres fourmis pour retrouver les sources de nourritures déja trouvées par leurs congénères.

Les fourmis utilisent l'environnement comme support de communication : elles échangent indirectement de l'information en déposant des phéromones, le tout décrivant l'état de leur « travail ». L'information échangée a une portée locale, seule une fourmi située à l'endroit

où les phéromones ont été déposées y a accès. Ce système porte le nom de « stigmergie », et se retrouve chez plusieurs animaux sociaux (il a notamment été étudié dans le cas de la construction de piliers dans les nids de termites).
Maintenant on donne la définition suivante :

Définition 3.1.1 *La stigmergie est un mécanisme de coordination indirecte entre les agents. Le principe est que la trace laissée dans l'environnement par l'action initiale stimule une action suivante, par le même agent ou un agent différent. De cette façon, les actions successives ont tendance à se renforcer et ainsi conduisant à l'émergence spontanée d'activité cohérente, apparemment systématique.*

La stigmergie a d'abord été observée dans la nature — les fourmis communiquent en déposant des phéromones derrière elles, pour que d'autres fourmis puissent suivre la piste jusqu'à la nourriture ou la colonie suivant les besoins, ce qui constitue un système stigmergique. Des phénomènes similaires sont visibles parmi toutes les espèces eusociales comme les termites, qui utilisent des phéromones pour construire de grandes et complexes structures de terre à l'aide d'une simple règle décentralisée. Chaque termite ramasse un peu de boue autour de lui, y incorporant des phéromones, et la dépose par terre. Comme les termites sont attirés par l'odeur, ils déposent plus souvent leur paquet là où d'autres l'ont déjà déposé, ce qui forme des piliers, des arches, des tunnels et des chambres.

3.1.4 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 $n = \sum_{i=1}^{k} b_i(t)$ le nombre total des fourmis.
Chaque fourmis 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 déplacement suivante :

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^{\alpha}.\left(\eta_{ij}\right)^{\beta}}{\sum_{l \in N^k} (\tau_{il}(t))^{\alpha}.(\eta_{il})^{\beta}} & \text{si } \ j \in N^k \\ 0 & \text{sinon} \end{cases} \tag{3.1.4}$$

avec :

$\tau_{ij}(t)$ 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 fourmis k . Il représente l'ensemble des villes qui restent à visiter. Celles déja 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 itération et les traces des phéromones sont mises à jour selon la formule ci-dessous :

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{3.1.5}$$

Où,

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

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

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{n} \Delta\tau_{ij}^{k}(t) \tag{3.1.6}$$

représente la quantité des phéromones par unité de longeur déposée sur l'arête (i,j) par la $k^{ème}$ fourmi entre t et $t+n$, elle est donnée par :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L_k} & \text{si la } k^{ème}\text{fourmi traverse l'arête } (i,j) \text{ dans son tour} \\ 0 & \text{sinon} \end{cases} \tag{3.1.7}$$

Avec,

L_k :la longeur du tour de la $k^{ème}$ fourmi.

Q : un nombre positif constant.

Algorihtme

Debut

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 $i \neq j$

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

$\tau_{ij} \leftarrow c$

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 ($k = 1$ à n)

initialiser la liste $L_v$comme suit :

L_v ←sommet affecté

pour $i = 1$à 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 (3.1.7)

Fin pour

Fin pour

Mettre à jour les traces de phéromone selon la règle (3.1.5)

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

$gap \leftarrow (L - sol\_opt)/sol\_opt$

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

3.2 Amélioration 2-opt

En optimisation, 2-opt est un algorithme de recherche locale proposé par Croes en 1958 pour résoudre le problème du voyageur de commerce en améliorant une solution initiale. [9]

Définition 3.2.1 *On définit une 2-permutation dans le cycle hamiltonien H comme le sibustitution de deux arête $e_1, e_2 \in H$ par deux arêtes $e_3, e_4 \in E$ tel que le tour résultant est toujours hamiltonien dans G.*

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.

3.2.1 Principe

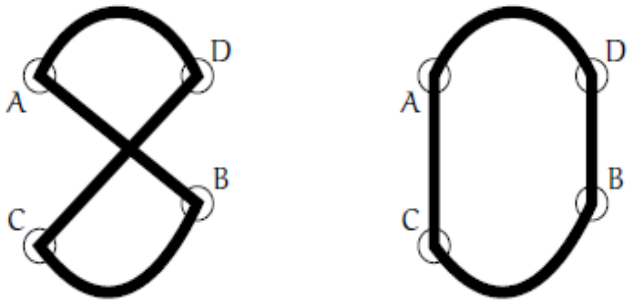
Le principe de l'algorithme « amélioration 2-opt » est assez simple et repose sur l'idée suivante :
– On génère un premier trajet
– On l'améliore jusqu'à ce qu'un critère d'arrêt ait été vérifié.

Génération du premier trajet On peut générer le trajet de telle sorte qu'à la $i - ème$ étape, le choix de la $(i + 1) - ème$ ville est la plus proche voisine de la $i - ème$ ville parmi les villes à parcourir.

Amélioration du trajet Une amélioration possible est de supprimer tout croisement.
En s'appuyant sur l'inégalité du parallélogramme, prenons le cas d'un trajet ou un chemin direct relie$A$ à B, un chemin relie B à C et ensuite un chemin relie C à D . On voit que si :

$$(AB + CD)^2 > (AC + BD)^2 \tag{3.2.1}$$

alors il y a présence de croisement et il apparaît plus efficace alors de passer de A à C, de relier Cà B puis de passer de B à D.



(Exemple de croisement)

Critère d'arrêt Un critère d'arrêt possible est que la fonction d'amélioration définie précédemment ne modifie plus le trajet. Autrement dit dès qu'il n'y a plus de croisement dans le trajet.

Algorithme

Debut

choisir une tourneé initiale T_1 et calcuer $C(T_1)$

donner n

donner la solution optimale (sol_opt)

$J \leftarrow 1$

amélioré←vrai

tantque (améliore ou $J < n$)

améliore ←faux

prendre une paire d'arêtes non adjacentes

construire T_2 et calculer$C(T_2)$

si $C(T_2) \leq C(T_1)$ alors

$T_1 \leftarrow T_2$

améliore←vrai

$J \leftarrow 1$

sinon

$J \leftarrow J + 1$

fin si

$gap \leftarrow (T_2 - sol\_opt)/sol\_opt$

fin tant que
fin de l'algorithme

3.3 Technique du recuit simulé

Cette technique, a été mise au point par : S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983, et indépendamment par V. Cerny en 1985 à partir de l'algorithme de Metropolis ; qui permet de décrire l'évolution d'un système thermodynamique.[9]
La méthode du recuit simulé est basée sur un processus utilisé en métallurgie pour obtenir un alliage sans défaut, ce processus est appelé « le recuit ».
On commence d'abord par chauffer le métal jusqu'à une certaine température où il devient liquide (les atomes peuvent donc circuler librement). Après avoir atteint ce stade, on abaisse la température très lentement de sorte à obtenir un solide. Si cette baisse de température est brusque on obtient alors un verre ; si au contraire cette baisse de température est très lente (laissant aux atomes le temps d'atteindre l'équilibre statistique),nous obtiendrons des structures de plus en plus régulières, jusqu'à atteindre un état d'énergie minimale correspondant à la structure parfaite d'un Crystal, on dit alors que le système est « gelé ».
Au cas où cet abaissement de température ne se ferait pas assez lentement, il pourrait apparaitre des défauts. Il faudrait alors les corriger en réchauffant de nouveau légèrement la matière de façon à permettre aux atomes de retrouver la liberté de mouvement, leur facilitant ainsi un éventuel réarrangement conduisant à une structure plus stable.

3.3.1 Idée principale

L'idée principale du recuit simulé (proposé par Metropolis en 1953) repose sur la simulation du comportement de la matière dans le processus du reuit. Le but est d'atteindre un état d'équilibre thermodynamique, cet état d'équilibre (où l'énergie est minimale) représente - dans la méthode du recuit simulé – la solution optimale d'un problème ; L'énergie du système sera calculée par une fonction coût (ou fonction objectif) spécifique à chaque problème (Ken-

dall). La méthode va donc essayer de trouver la solution optimale en optimisant une fonction objectif, pour cela, un paramètre fictif de température a été ajouté par Kirkpatrick, Gelatt et Vecchi. En gros le principe consiste à générer successivement des configurations à partir d'une solution initiale S_0 et d'une température initiale T_0 qui diminuera tout au long du processus jusqu'à atteindre une température finale ou un état d'équilibre (optimum global).

Propriété L'état d'un matériau dépend de la température à laquelle il est porté et la distribution de Gibbs-Boltzmann
mesure la probabilité$P(X)$ de visiter l'état X en fonction de son énergie $E(X)$ et de la température T :

$$P(X) = \exp(\frac{-\Delta E}{KT}) \tag{3.3.1}$$

avec ;
K est la constante de Boltzmann qui est égale à :

$$K = 1.3806488 \times 10^{-23} J.K^{-1}$$

Très souvent les heuristiques nous fournissent des solutions qui sont des optimums locaux. Afin d'éviter de tomber sur un optimum local on accepte de prendre une solution méme mauvaise avec une certaine probabilité.

3.3.2 L'algorithme de Metropolis

Dans l'algorithme de Metropolis, on part d'une configuration donnée, et on lui fait subir une modification aléatoire. Si cette modification fait diminuer la fonction objectif (ou énergie du système), elle est directement acceptée ; Sinon, elle n'est acceptée qu'avec une probabilité égale à $\exp \frac{-\Delta E}{KT}$, cette règle est appelée critère de Metropolis.

Algorithme

Debut

choisir une tourneé initiale T_1 et calcuer $C(T_1)$

donner la solution optimale (sol_opt)

donner n

donner la température initiale t_i

$J \leftarrow 1$

améliore←vrai

tantque (améliore ou $J < n$)

améliore ←faux

prendre une paire d'arêtes non adjacentes

construire T_2 et calculer$C(T_2)$

si $C(T_2) \leq C(T_1)$ alors

$T_1 \leftarrow T_2$

améliore←vrai

$J \leftarrow 1$

sinon

Tirer au hasard un nombre r dans l'intervalle $[0, 1]$

calculer la probabilité $P(T_2)$

si $r < P(T_2)$ alors

$T_1 \leftarrow T_2$

fin si

fin si

$gap \leftarrow (T_2 - sol\_opt)/sol\_opt$

fin tant que

fin de l'algorithme

3.4 Algorithme du plus proche voisin

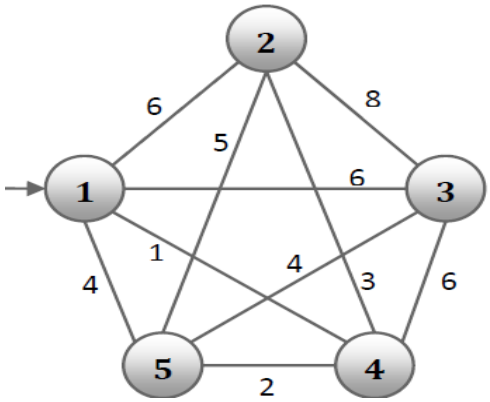
L'algorithme du plus proche voisin est une heuristique de construction simple . C'est l'un des premiers algorithmes utilisés pour déterminer une solution au problème du voyageur de commerce. Il donne rapidement une courte visite, mais généralement pas la solution optimale.C'est un algorithme de type 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 n'appartenant pas déjà au chemin, jusqu'à ce que tous les sommets aient été parcourus.

3.4.1 Principe

1. 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 enfin revenir à la première ville choisie, pour obtenir un cycle.

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



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

$$Sa = 1 + 2 + 4 + 8 + 6 = 21$$

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

1-4-5-3-2-1

Algorithme

Debut

Choisir un sommet $v_1 \in V$

1. Poser $k \leftarrow 1$

   pose $U = \{v_1\}$

   Tant que $k < n$

   $k \leftarrow k + 1$

   choisir v_k dans $X = V \backslash U$ qui vérifie

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

   faire

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

   Fin tant que

Fin de l'algorithme

Chapitre 4

Tests numériques

Nous avons programmé les différentes heuristiques citées plus haut en language **MATLAB** sur un pc **hp :** processeur intel(R) core(TM) i5-4210U, RAM : 4 Go .
Les exemples ont été tirés des instances d'OR-Library [13] ,[14]et [15]
Pour $n = 100$ (nombre d'itération) et $\alpha = 2, \beta = 6, \rho = 0.6, Q = 100$ pour l'algorithme de colonie de fourmis, on a obtenu les résultats suivants

		Colonie de fourmis			2-opt			Recuit simulé			plus proche voisin		
Le problème	La Solution optimale	solution	gap	CPU	solution	gap	CPU	solution	gap	CPU	solution	gap	CPU
eil76	538	562.61	0.0458	3.219162	570.10	0.0597	6.137006	711.99	0.3234	2.181180	866.52	0.6106	43.560958
d198	15780	17778	0.1266	43.182926	16189	0.0259	5.540566	18620	0.1800	3.300894	21845	0.3843	67.765814
bier127	118282	13218	0.1175	16.417401	120470	0.0185	6.645140	134320	0.1356	3.692836	167300	0.1444	53.992056
ch150	6528	6883	0.0544	22.692631	6622.8	0.0145	7.764088	8194.6	0.2553	6.415525	10329	0.5823	111.918344
gil262	2378	2693.8	0.1328	94.120935	2508.8	0.0550	10.73631	3241.5	0.3631	5.396725	3935.4	0.6549	101.826406
lin318	42029	49113	0.9113	148.085597	44364	0.0555	6.846716	54004	0.2849	4.750007	63149	0.5025	51.115310
u159	42080	49630	0.1794	138.248802	44364	0.5007	6.947012	54004	0.2834	4.017718	63149	0.5007	63.369596
st70	675	723.36	0.0717	3.818482	691.08	0.0238	4.254239	805.53	0.1934	4.327092	962.39	0.4258	39.633680
rat99	1211	1327.70	0.0964	6.434230	1295.5	0.0698	4.346159	1564.7	0.2921	4.750704	1996	0.6482	47.493920
berlin52	7542	7890.10	0.0462	1.543182	7841.4	0.3571	7.697362	8980.9	0.1908	6.092215	10235	0.3571	66.237113
kroC100	20749	22171	0.0686	8.575216	22007	0.0606	6.654826	26188	0.2621	8.362603	33490	0.6140	110.94232
pr226	80369	91829	0.1426	56.254449	82295	0.0240	5.373407	94685	0.1781	7.881604	10614	0.3206	173.485645
pr144	58537	61818	0.0566	16.245669	61244	9.4637	12.918220	61651	0.0532	8.372808	71006	0.2130	57.527618

D'après les tests numériques que nous avons pu effectué, on constate que la méthode du 2-opt donne de bons résultats (qualité des solutions et temps raisonnable d'exécution).
l'algorithme de colonie de fourmis donne des solutions plus présices mais de mauvais temps d'exécution).
Les solutions fournies par la méthode du recuit simulé sont de mauvaises qualité . Ce résultat est prévisible car cette dérnière est plutot indiquée pour des problèmes de grandes tailles.

Enfin la méthode du plus proche voisin que l'on peut considérer comme trés rudimentaire ne peut être utilisée que pour initialiser une tournée pouvant servir de point de départ à une autre heuristique.

CONCLUSION

Il est trés difficile de pouvoir classer les heuristiques : Chaque heuristique présente des avantages et des inconvénients.

Les méthaheuristiques telles que l'adaptation de la méthode de colonie de fourmis au problème du voyageur de commerce ou celle du recuit simulé sont bien adaptées pour résoudre des problèmes de grandes tailles quoique assez couteuses en temps.

Les méthodes spécifiques au problème telles que la méthode du 2-opt (qui donne de bon résultats) et celle du plus proches voisins sont les mieux indiquées pour des problèmes de petites tailles. Nous pensons plutot à des méthodes hybrides qui combinent plusieurs méthodes pour pouvoir obtenir de bon résultats. Chose que nous projetons de faire à l'avenir.

[1] A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.

[2] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italie, 1992

[3] Optimisation par colonies de fourmis. COSTANZO Andrea. LUONG Thé Van. MARILL Guillaume.

[4] J. Dréo, al, «Méta heuristiques pour l'optimisation difficile» livre Éditions Eyrolles ,2003.

[5] M. Dorigo, M. Birattari, and T. Stiitzle, «Ant Colony Optimization», Tsinghua University Press, Beijing, 2007.

[6] Local search in Combinatorial Optimization, E. Aarts et K. Lenstra, Wiley – 1997.

[9] G. A, Croes,. : A method for solving traveling salesman problems, Operations Res. 6, 1958, pp. 791-812.

[10]Optimization by Simulated Annealing [Article] / aut. Kirkpatrick, Gelatt et Vecchi // Science, New Series. - 13 Mai 1988. - 4598. - pp. 671-680.

[11]. (en) Christian Nilsson, « Heuristics for the Traveling Salesman Problem » , Linköping University, 2003

[12]http ://fr.wikipedia.org/wiki/Probl%C3%A8me_du_voyageur_de_commerce

[13] http ://people.brunel.ac.uk/~mastjjb/jeb/info.html.

[14] http ://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/.

[15] http ://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/STSP.html.

[16] https ://interstices.info/jcms/c_37686/le-probleme-du-voyageur-de-commerce.

[17]http ://webcache.googleusercontent.com/search ?q=cache :COkaCwPMSrMJ :polymorphe.free.fr/cour
pdf+&cd=3&hl=ar&ct=clnk&gl=dz

[18]http ://fr.wikipedia.org/