nétique3.111Schéma du fonctionnement de l'algorithme génétiquefigure.3.1

Remerciements

Je tiens à adresser mes remerciements les plus sincères à Monsieur Riadh FREFITA, mon enseignant du module complexité et modélisation. Elle nous a apporté son aide et a contribué à l'élaboration de ce travail ainsi qu'à la réussite de ce projet,Voyageur de commerce et algorithmes génétiques.

J'exprime mon gratitude à notre enseignant qui a consacré de son temps à nous assurer des seances de formations et n'a pas hésité à nous renseigner et à nous montrer comment procéder pour maintenir la bonne démarche à suivre.

Table des matières

Table des figures

1

Introduction

Le probléme du voyageur de commerce, consiste en la recherche d'un trajet minimal permettant à un voyageur de visiter n villes. En régle générale on cherche à minimiser le temps de parcours total ou la distance totale parcourue. On se donne n villes ainsi que les distances (longueur ou temps) entre chaque ville. On peut représenter le probléme sous forme déun graphe et déune matrice repèsentant le graphe (les villes, sont indices de 0 à n).

De facon gènèrale, on peut considèrer des graphes ou˘ certains arcs (chemins) n'existent pas. Dans la matrice reprèsentant le graphe, les arcs inexistants ont une distance strictement nègative (1). Par ailleurs, on ne considère pas les arcs reliant une ville à elle meme. Le voyageur de commerce devant revenir dans la ville de départ, on peut représenter une "tournée" par un vecteur de dimension n, contenant la liste des villes visitées dans l'ordre. Une tournée est définie à une permutation prés. Une fois la ville de départ fixée, elle est par contre unique. Le probléme du voyageur rentre dans la catègorie des problèmes NP-complets, à savoir que sa complexitè n'est pas polynomiale. Il existe deux grandes classes d'algorithme : les algorithmes dèterministes qui dèterminent la solution optimale (complexitè n!) et les algorithmes d'approximation qui se "contentent" de trouver une bonne solution (non optimale) dont le cout est voisin du cout minimal. Les algorithmes gènètiques font partie de cette seconde catègorie. On trouvera sur le Web ènormèment de ressources sur ce problème (voyageur commerce sur Google par exemple).

2

Context du projet

2.1 Introduction

Les dix recommandations formulées nous permettre d'optimaliser notre réseau routier de manière efficiente, rapide et abordable, tout en préservant la fonction sociale et économique que nous autorités et usagers lui attribuons.

2.2 Structurer le réseau routier

La croissance continue de la mobilité a entraîné une saturation du réseau autoroutier aux heures de pointe. Aux abords de grandes villes telles que Bruxelles et Anvers, les embouteillages en matinée et en soirée sont particulièrement importants. En dehors des heures de pointe, nous sommes également de plus en plus souvent confrontés à des embouteillages suite à des incidents ou d'autres types d'événements. Malgré ces difficultés, les autorités désirent faire transiter davantage le trafic par le réseau autoroutier. Par conséquent, la pression sur ce réseau va encore s'accroître et la fluidité du trafic y deviendra particulièrement vulnérable. De légères perturbations peuvent alors provoquer des embouteillages.

Tout orienter vers les autoroutes n'est donc pas une bonne stratégie. Au contraire ! A l'instar de nos pays voisins, nous sommes tenus de modifier la structure du réseau routier

de manière à réorienter les déplacements régionaux (courts) du réseau autoroutier vers un réseau (système) dédié. Ce transfert peut se faire en valorisant une partie du réseau routier régional actuel, ce qui lui permettrait de fonctionner comme un réseau de voies de circulation cohérent, parallèle au réseau autoroutier.

2.3 Problématique d'un voyageur de commerce

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. Quel chemin faut-il choisir afin de minimiser 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 : dans tous les cas, on parle de coût.
Donc le problème de voyageur de commerce consiste à la recherche de trajet minimal en traversant les villes.

2.3.1 Problématique générale

Conception multimodale du réseau routier :
Pour répondre à la demande en déplacements de la population, l'état développe une conception multimodale de la mobilité. Il s'agit de considérer le système de transports avec une vision globale où chaque mode de transport a son domaine de pertinence en fonction du motif de déplacement et de l'horaire de déplacement, tant du point de vue de l'efficacité (performance), de l'économie (individuelle et collective).
Organisation du réseau routier :
Pour garantir le fonctionnement des transports individuels motorisés, les déplacements sont organisés selon une logique de type radio-concentrique, du centre vers la périphérie.
Pour pouvoir maximiser l'utilisation du réseau et diminuer ainsi la pression sur les réseaux locaux, une augmentation de la capacité du système autoroutier est nécessaire. Cette optimisation peut être obtenue par différents types d'intervention.

2.4 Présentation

Etant donné un ensemble de villes séparées par des distances données, trouver l'itinéraire le plus court passant par chaque ville une et une seule fois.
En termes de graphe de complexité, on cherche le cycle hamiltonien le plus court.

2.4.1 Domaine d'application

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 ?

2.4.2 Principe d'implementation

A partir d'une matrice :
Cij : représente le coût du déplacement entre la ville i et la ville j, il faut trouver une permutation qui minimise la somme des coûts. Autrement dit, il faut trouver un cycle hamiltonien de longueur minimale dans un graphe pondéré complet.

2.4.3 Complexité

Ce problème est un représentant de la classe des problèmes NP-complets. L'existence d'un algorithme de complexité polynomiale reste inconnue. Un calcul rapide de la complexité montre qu'elle est en O(n !)avec villes il y a chemins possibles.

2.4.3.1 Exemple

En supposant que le temps pour évaluer un trajet est de 1 µs, le tableau ci-dessous montre l'explosion combinatoire du PCV.

# Villes	#Possibilités	Temps de calcul
5	12	12 µs
15	42 milliard	12 heures
20	60E15	1928 ans
25	310E21	9.8 milliard d'années

Figure 2.1: Modélisation d'un exemple

Nous remarquons que les possibilités augmentent exponentiellement avec le nombre de villes.

Figure 2.2: Modélisation du complexité de l'exemple

2.5 Plan de travail

Proposer une structure des classes avec les structures de donnèes et le fonctions membres.
Implèmenter les classes en validant toutes les fonctionnalitès élèmentaires.
Valider le programme complet sur des petits cas tests .
Valider sur des cas plus significatif.
Comparer diffÈrentes stratègies en essayant de trouver la meilleure.

2.6 Conclusion

La théorie des complexité des graphes sont de très vaste domaine, en évolution constante tant du point de vue des recherches fondamentales que de celui des applications. Durant ces modules, nous allons essayer, grâce aux outils appris en cours de résoudre des problèmes bien définis.

3

Solution proposée

3.1 Introduction

On doit établir une démarche générale permettant de résoudre le problème pour n'importe quel ensemble de villes :
Cette démarche s'appelle un algorithme.
Les algorithmes pour résoudre le problème du voyageur de commerce peuvent être répartis en deux classes :
les algorithmes déterministes qui trouvent la solution optimale.
les algorithmes d'approximation qui fournissent une solution presque optimale.
Elles permettent d'obtenir en un temps rapide de bonnes solutions, pas nécessairement optimales mais de qualité suffisante.

3.2 Algorithme gènètique

3.2.1 Présentation

Les algorithmes gènètiques s'appuient sur un principe de sélection des individus d'une population qui prèsentent des caractèristiques se rapprochant au mieux de ce que l'on cherche. Cette population èvoluant par ailleurs selon des critéres d'Èvolution gènètique choisi.
Dans le contexte du probléme du voyageur de commerce, un individu est une tournèe, un chemin et une population un ensemble de tournèes.
Il s'agit maintenant de d'unir un critére de sèlection ainsi que des régles d'Èvolution de la population.
En la matière, il y en a de nombreuses et nous n'en avons retenu que les principales. Il faut toutefois garder l'esprit qu'il y a trés peu de rèsultats thèoriques sur les algorithmes gènètiques et que le choix des critères et régles est donc heuristique.
Sachant que l'on a un instant donnè k une population,
Pk constituèe de p individus,
comment gènèrer une population Pk+1.
l'instant suivant toujours constituèe de p individus mais qui prèsente de meilleurs caractèristiques.
Il y a donc deux Ètapes : la gènèration d'une nouvelle population et la sèlection des "meilleurs" individus.

3.2.2 Historique et fonction d'adaptation

Il s'agit de caractèriser l'adaptation d'un individu au chemin visé.
Pour le probléme du voyageur de commerce, on peut simplement utiliser la distance de la tournèe comme fonction d'adaptation d'un individu, Ètant entendu que plus cette distance est petite meilleure est son adaptation.
Les algorithmes génétiques forment une famille très intéressante d'algorithmes d'optimisation, initiée par Charles Darwin au XIXème siècle.
Le principe des algorithmes génétiques s'inspire directement des lois de la sélection naturelle. On l'utilise dans la résolution de problèmes complexes, nécessitant des temps de calcul

élevés. Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes d'évolution de la nature :
croisements, mutations, sélections, etc..
Ils appartiennent à la classe des algorithmes évolutionnaires.
L'idée d'un algorithme génétique est tirée de la théorie darwinienne de l'évolution. Chaque groupe d'individus (animaux, espèces végétales...), appelé aussi population, donne lieu à une génération suivante par reproduction sexuelle.
Cette génération consiste à croiser les individus entre eux pour donner des descendants possédant les caractères des deux parents. En plus de ce croisement, des mutations de caractères interviennent aléatoirement dans la génération de la population suivante. Puis, cette nouvelle population subit une sélection, métaphore de la sélection naturelle : seuls les individus les mieux adaptés à l'environnement survivent.
Enfin à son tour, cette population donnera lieu par le même processus à une nouvelle population, qui sera encore plus performante dans son environnement. Pour développer un algorithme génétique, les individus sont des solutions, la sélection se fait grâce à leur qualité (évaluée à travers la fonction objectif), les croisements entre deux solutions sont faits à l'aide d'opérateurs de croisement, etc.

3.2.3 Principe de l'algorithme

Compte-tenu des diffèrentes statègies que l'on peut mettre en place et des diffèrents paramétres qui interviennent dans ces stratègies et qu'il n'y a pas de rèsultats de convergence, lié a la capacité d'un tel algorithme repose sur une bonne analyse des causes d'Èchecs et pas mal de bon sens.
De facon gènèrale, le problème que l'on cherche à rèsoudre prèsente beaucoup de locaux et le danger est de se retrouver pièger dans un tres mauvais minimum local.
C'est souvent du au fait qu'il n'y a pas assez de variabilitè et d'èvolution dans les populations gènèrèes. Si on augmente trop la variabilitè en forcant le cross-over par exemple, on risque d'avoir un comportement chaotique car les populations varient trop d'une itèration à l'autre.
L'algorithme est ègalement tres sensible à la population de dèpart. Le choix de parcours alèatoire n'est pas tres bon. Une assez bonne mèthode d'initialisation consiste à construire un chemin initial en prenant le plus proche voisin du point ou on se trouve.

3.2.4 Implementation de l'algorithme

Il s'agit de proposer une implèmentation de l'algorithme a priori indèpendante du problËme à rèsoudre. On introduira à cet effet une classe d'Individu, une classe de population et les fonctionnalitès dont on a besoin (crossover de deux individus, mutation, sÈlection, ...). L'interaction avec le probléme du voyageur de commerce se concentre au niveau de la classe Individu :

Èvaluation de la fonction d'adaptation, croos-over et mutation. Il y a plusieurs facons de gèrer cette interaction :

Par hèritage :

un chemin dèrivant de la classe individu (classe abstraite), la classe chemin devant proposer les mèthodes attendues par la classe individu

Par gènèricitè :

la classe individu etant gènèrique, le type de individu pouvant etre un chemin (individu<chemin>)

Je vous suggére d'utiliser la mèthode par hèritage. Il sera utile de prèvoir une fonction permettant de gènèrer un class qui contiendra le cotion minimal obtenu à chaque itèration, ce qui permettra de reprèsenter graphiquement le comportement de l'algorithme.

3.2.5 Initialisation des variables

Un individu→ trajet

Son évaluation→la longueur totale de ce trajet.

Un gène→ville où l'on passe à une certaine position dans le parcours.

ADN→ la liste des villes dans l'ordre du parcours.

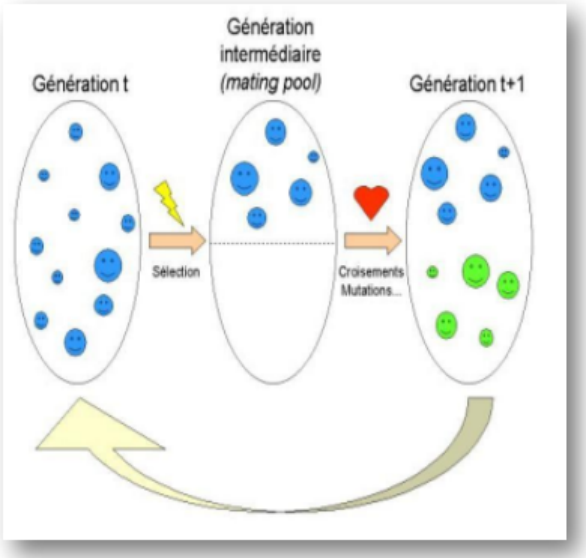

Figure 3.1: Schéma du fonctionnement de l'algorithme génétique

3.2.6 Description

Les étapes suivantes sont exécutées (figure 6) :

Tout d'abord, chaque individu est évalué. Pour cela, on calcule la valeur de la fonction objective, c'est-à-dire la longueur du cycle parcouru par le voyageur de commerce.

Puis, une étape de sélection est appliquée. Cette étape permet d'éliminer les moins bons individus et de garder uniquement les meilleurs en fonction de leur évaluation. Il existe plusieurs méthodes de sélection. Une sélection par rang ne fait pas intervenir le hasard, contrairement à la roulette, car elle choisit les n meilleurs individus de la population. Chaque individu a ainsi une probabilité de sélection dépendant de son évaluation, avec une plus forte probabilité pour les meilleurs individus.

L'étape suivante consiste à croiser les individus précédemment sélectionnés pour obtenir une nouvelle population. Deux parents sont donc choisis pour appliquer un opérateur de croisement afin d'obtenir un descendant (nouvel individu).

Il existe de nombreuses techniques de croisement ; dans le cas présent, nous utiliserons le « crossover en un point ». Cet opérateur consiste à recopier une partie du parent 1 et une partie du parent 2 pour obtenir un nouvel individu. Le point de séparation des parents est appelé point de croisement. Il faut cependant faire attention à ne pas visiter plusieurs fois la même ville (on ne recopie pas les villes déjà visitées), et à ne pas oublier de ville (on rajoute

à la fin les villes non prises en compte).

On prend l'exemple avec 8 villes et un point de croisement juste après la troisième ville.

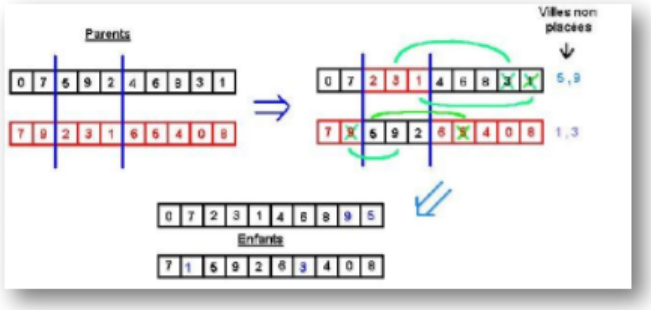

Figure 3.2: croissement

Pour mieux comprendre le croissement, voilà les étapes à suivre :

1. On choisit aléatoirement deux points de découpe.
2. On interverti, entre les deux parcours, les parties qui se trouvent entre ces deux points.
3. On supprime, à l'extérieur des points de coupe, les villes qui sont déjà placées entre les points de coupe.

Enfin, avant de revenir à la première étape, un procédé de mutation est utilisé pour diversifier les solutions au fur et à mesure des générations.

Cette mutation consiste à modifier aléatoirement une petite partie d'un caractère dans certains individus de la nouvelle génération. Cette étape est effectuée avec une très faible probabilité, et consiste par exemple à échanger deux villes consécutives dans un individu.

3.3 Algorithme des plus proches voisins

La première idée consiste à se rendre systématiquement dans la ville la plus proche de celle où l'on se trouve, telle que le poids entre la ville courante et la prochaine ville soit minimale :

sans y réfléchir plus, on pourrait penser que cette méthode résout effectivement le problème posé.

3.3.1 Principe

Le trajet est initialement vide (pas de villes visitées) On va partir d'une ville au hasard, que l'on met dans la liste des villes visitées.
On cherche la ville la plus proche que l'on ajoute elle-même dans la liste des villes visitées. On cherche la ville la plus proche de la cette nouvelle ville. Si la ville est déjà dans la liste des villes visitées, on prend la deuxième ville plus proche, la troisième si besoin, etc. et ainsi de suite..
Une manière d'accélérer fortement le calcul de ces solutions est de calculer auparavant le tableau des villes les plus proches (pour chaque ville, contient la liste des autres villes triées dans l'ordre du plus proche au plus éloigné).

3.3.2 Exemple

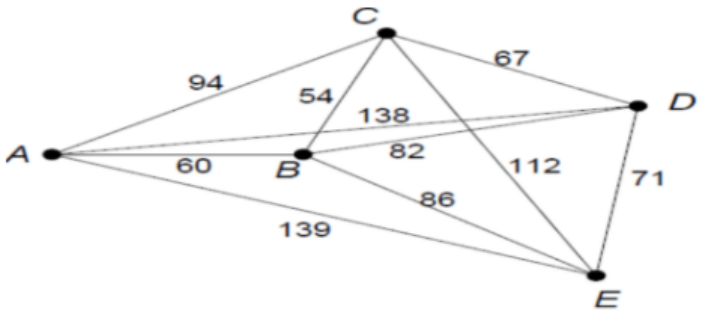

Figure 3.3: Exemple de cas : Algorithme des plus proches voisins

Par exemple, quel itinéraire devons-nous choisir pour visiter les cinq villes A, B, C, D et E de la figure ci-contre (les distances sont données en kilomètres) ? On commence par la ville A, les successeurs de la ville A sont B, C, E, D. Le plus court chemin c'est [AB].
Les successeurs de la ville B sont C, E, D. Le plus court chemin c'est [BC].
Les successeurs de la ville C sont B, E, D, A. Mais B et A sont déjà dans le parcours. Le plus court chemin c'est [CD]. Les successeurs de la ville D sont A, B, C, E. Mais B, A et C sont déjà dans le parcours. Le plus court chemin c'est [DE]. Et enfin [EA].
Donc le parcours est ABCDEA et le coût du chemin est égal à :
60+54+67+71+139=391 Km. Si on choisit un autre parcours pour cet exemple, on trouve que cette méthode peut donner des résultats non optimaux (le parcourt ABEDCA= 387Km).

3.4 Algorithme colonie de fourmis

3.4.1 Origine

L'idée originale provient de l'observation de l'exploitation des ressources alimentaires chez les fourmis. Elles sont capables collectivement de trouver le chemin le plus court entre une source de nourriture et leur nid. Des biologistes ont ainsi observé, dans une série d'expériences menées à partir de 1989, qu'une colonie de fourmis ayant le choix entre deux chemins d'inégale longueur menant à une source de nourriture avait tendance à utiliser le chemin le plus court.

3.4.2 Principe

Les algorithmes de colonies de fourmis sont des algorithmes inspirés du comportement des fourmis.
Le premier algorithme de colonies de fourmis proposé est appelé le Ant system (système fourmi).
Il vise notamment à résoudre le problème du voyageur de commerce, où le but est de trouver le plus court chemin permettant de relier un ensemble de villes.
L'algorithme général est relativement simple, et repose sur un ensemble de fourmis, chacune parcourant un trajet parmi ceux possibles.
À chaque étape, la fourmi choisit de passer d'une ville à une autre en fonction de quelques règles :
Elle ne peut visiter qu'une fois chaque ville ; plus une ville « visibilité ») ; plus l'intensité de la piste de phéromone disposée sur l'arrête entre deux villes est grande, plus le trajet aura de chance d'être choisi ; une fois son trajet terminé, la fourmi dépose, sur l'ensemble des arêtes parcourues, plus de phéromones si le trajet est court ; les pistes de phéromones s'évaporent à chaque itération.
Le partage des données sur les phéromones est le point fort de cette technique

3.4.3 Description

L'algorithme « Ant System » optimisant le problème du voyageur de commerce :
1) une fourmi choisit un trajet, et trace une piste de phéromone.

2) l'ensemble des fourmis parcourt un certain nombre de trajets, chaque fourmi déposant une quantité de phéromone proportionnelle à la qualité du parcours.
3) chaque arête du meilleur chemin est plus renforcée que les autres.
4) l'évaporation fait disparaître les mauvaises solutions

3.4.4 Complexité

O(|V.V.V|.c)

3.5 Algorithme de Christofides

L'algorithme de Christofides est un algorithme d'approximation pour le problème du voyageur de commerce, dans le cas métrique.
c'est-à-dire quand l'inégalité triangulaire est respectée(c'est-à-dire que pour tout triplet de sommet u,v,w, on a w(u,v)+w(v,w)>= w(u,w)).
L'analyse de cet algorithme est due à Nicos Christofides.

3.5.1 Problématique

Utiliser l'heuristique de Christofides (est une méthode de calcul qui fournit rapidement une solution réalisable, pas nécessairement optimale ou exacte, pour un problème d'optimisation difficile) pour construire un cycle puis l'algorithme de recherche locale 2-opt pour l'améliorer.

3.5.2 Implémentation de l'algoritmhe de Christofides

Christofides' est une heuristique de construction de cycle qui peut être appliqué aux graphes avec la propriété suivante :
un graphe complet G = (V,E,w) avec des poids d'arcs qui satisfont l'inégalité du triangle w(x,y) + w(y,z) w(x,z).
Trouver un arbre couvrant minimal T de G :
Soit O l'ensemble des sommets avec un degré impair dans T
Trouver un couplage parfait M avec un poids minimum sur les sommets de

Ajouter M à T pour obtenir multi graphe H

Trouver un cycle Eulérien de H

Convertir le cycle Eulérien en un chemin Hamiltonien en sautant les nœuds visités (short-cuts).

3.5.3 Traitement des itération

L'algorithme traite un ensemble Q contenant tous les sommets qui ne sont pas encore dans l'arbre (Initialement, Q est vide). Le programme itère pour chaque sommet v non encore présent dans l'arbre (utilisant le sommet 0 comme le sommet initial) et choisi l'arête de poids minimal (u, v) où u est un sommet déjà présent dans l'arbre. On obtient ainsi l'arête de poids minimum traversant le passage (puisqu'il relie une arête dans le MST avec une arête qui n'y est pas encore). Le sommet v est ensuite ajouté dans l'arbre. L'algorithme continue jusqu'à ce que Q soit vide de sorte que tous les sommets soient ajoutés dans le MST.

3.5.3.1 Arbre couvrant minimum (MST)

On a utilisé l'algorithme de Prim pour trouver l'arbre couvrant minimum dans G :

function MST(G = (V, E))

for v in V do

key[v] <– infinity

parent[v] <– NULL

insert v into Q

end for

key[0] <– 0

while !Q.empty() do

v <– Q.removeMin()

for u adjacent to v do

if u Q and weight(u, v) < key[u] then

parent[u] <– v

end if

key[v] <– weight(u,v)

end for

end while

end function

Figure 3.4: Noeuds

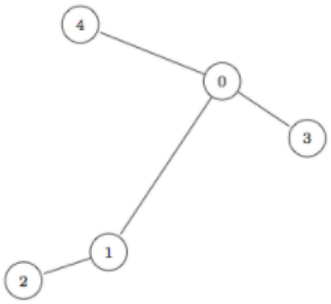

Figure 3.5: Trouvez le noeud

Sommets avec un degré impair dans le MST La prochaine étape consiste à trouver les sommets de degré impair dans le MST. Comme l'arbre est stocké dans une liste de proximités avec des vecteurs , cette procédure nécessite uniquement de vérifier la taille des vecteurs pour chaque nœud dans la liste. Parfaite adéquation pondérée pour sommets impairs Un graphe de connexe sans boucles à un n arrête et n-1 degrés.

Nous trouvons maintenant une correspondance parfaite de tous ces sommets de telle sorte que tous les sommets aient un degré pair.

Idéalement, nous devrions trouver une correspondance minimale, mais a la place, on a utilisé un algorithme glouton pour trouver une correspondance minimale approximative.

function PerfectMatching()

Input : odds (list of odd vertices), G (adjacency list)

while !odds.empty do

v <– odds.popFront()

length <–

for u odds do

if weight(u,v) < length then

length <– weight(u,v)

closest <– u

end if

end for

G.addEdge(closest,u)

odds.remove(closest)

end while

end function

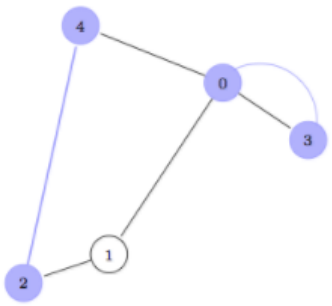

Figure 3.6: Trouvez le minimum des arcs

L'ensemble des sommets pairs sont maintenant ajoutés à notre MST, formant un nouveau multi graphe. Cycle Eulérien Ensuite, nous parcourons le graphe de façon à créer un cycle eulérien.

Pour commencer on choisi aléatoirement un nœud de notre multi graphe, si ce nœud à des voisins, on l'ajoute à une pile, on sélectionne un voisin, on supprime l'arrête qui les relies du graphe, puis on utilise ce voisin comme sommet courant.

Si notre sommet n'a plus de voisin, nous l'ajoutons à notre parcours et enlevons le sommet en haut de la pile pour l'utiliser comme notre sommet courant.

Nous continuons le tracé de notre parcours de cette façon jusqu'à ce que la pile soit vide et

que le dernier sommet n'ai plus de voisin.

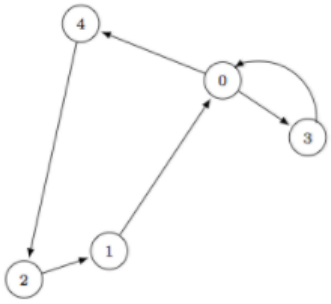

Figure 3.7: Cycle Eulérien

Chemin Hamiltonien Enfin, nous transformons notre cycle Eulérien en un chemin Hamiltonien en marchant le long de la tournée Euler, tout en vérifiant à chaque arrêt si ce nœud a déjà été visité. Si c'est le cas, nous ignorons ce nœud et passons au suivant.
Comme notre graphe satisfait l'inégalité du triangle, court-circuité de cette façon les sommets n'augmentera pas la longueur de notre chemin.

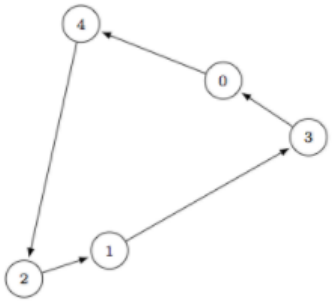

Figure 3.8: Conversion des cycles

Two-Opt Après qu'une tournée ai été construite en utilisant l'heuristique Christofides, on a appliqué l'algorithme de recherche locale 2-opt pour optimiser le chemin.
L'algorithme 2.opt examine chaque arête du parcours. Pour chaque arête, il regarde toutes les arêtes non adjacentes, et détermine si en enlevant ces deux arêtes puis en les remettant d'une autre façon cela raccourcirait le parcours.
Si c'est le cas, les arêtes sont échangées.
La recherche continue jusqu'à ce que le parcours ne puisse plus être optimisé de cette façon.

3.6 Autres algorithmes

SA : Simulated Annealing

EN : Elastic Net

SOM : Self Organizing Map

FI : Farthest Insertion Heuristic

3.7 Conclusion

Pour conclure, ce projet a constitué une étape intéressante dans notre apprentissage de la complexité des algorithmes. Ce travail nous a permis de faire une étude comparatives des plusieurs algorithmes ainsi la mise en œuvre de l'algorithme génétique qui est largement utilisé dans la recherche scientifique. Dans ce projet, nous avons eu la possibilité de dégager la solution optimale ainsi que d'appliquer notre algorithme élu pour PVC.

4

Conclusion et prespective

Outre de grands groupes de renommée internationale, de nombreuses petites et moyennes entreprises se sont spécialisées dans le domaine des systèmes de transport intelligents. Elles apportent une contribution décisive à l'innovation. L'information routière est essentielle pour améliorer les conditions de déplacement des usagers :

>l'informationprévisionnelle,qui propose des itinéraires alternatifs ou des périodes de départ optimales, permet aux usagers de programmer leurs déplacements ;

> l'information en temps réel sur les conditions de circulation favorise le confort des usagers, qui peuvent ainsi adapter leur conduite ou leur itinéraire.

Elle améliore la sécurité routière, en évitant les suraccidents. Globalement, une information de qualité permet :

de réduire les coûts économiques et environnementaux pour les usagers de la route, par la fluidification du trafic ou par l'amélioration de la sécurité des déplacements ;

aux autorités de mieux gérer les situations de crise.