LE PROBLÈME DU VOYAGEUR DE COMMERCE

**Positionnement thématique**

Informatique (informatique pratique), mathématiques (analyse, probabilités)

**Ce TIPE fait l’objet d’un travail de groupe.**

**Mots-clés** :

Français Anglais

Optimisation combinatoire Combinatorial optimization

Algorithme glouton Greedy algorithm

Complexité polynomiale Polynomial complexity

Probabilité Probability

Méthode de Monte-Carlo Monte Carlo method

Heuristique Heuristic

**Bibliographie commentée (650 mots max)**

Le problème du voyageur de commerce a été posé pour la première fois sous la forme d’un jeu par William Rowan Hamilton en 1859. Son énoncé est le suivant : « Un voyageur de commerce doit visiter une et une seule fois un nombre fini de villes et revenir à son point d’origine. Trouvez l’ordre de visite des villes qui minimise la distance totale parcourue par le voyageur »

Ce problème d’optimisation combinatoire appartient à la classe des problèmes NP-Complets. Cela signifie qu’il n’existe pas d’algorithme de résolution en temps polynomial. Ainsi les algorithmes permettant de trouver la solution exacte au problème ont un coût temporel élevé et l’on leur préfère les algorithmes d’optimisation, plus rapides, qui trouvent une solution approchée.

*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 ?*

Le problème du voyageur de commerce peut être modélisé à l’aide d’un graphe constitué d’un ensemble de sommets et d’un ensemble d’arêtes. Chaque sommet représente une ville, une arête symbolise le passage d’une ville à une autre, et on lui associe un poids qui représente la distance entre celles-ci.

Résoudre le problème du voyageur de commerce revient à trouver dans ce graphe un cycle passant par tous les sommets une unique fois (un tel cycle est dithamiltonien) et qui soit de longueur minimale. Le langage de programmation employé est Python.

Pour pallier ce problème, nous nous sommes d’abord intéressés à l’algorithme de Kruskal [2]. Il permet d’établir l’arbre couvrant de poids minimal passant par toutes les villes. Ainsi, les arêtes les plus importantes dans le graphe sont évitées. Cet algorithme ne résout toutefois pas le problème du voyageur de commerce. En effet, il ne passe pas nécessairement une seule fois par chaque ville. Il exhibe simplement des tronçons de courte distance.

Ensuite, nous avons essayé d’approcher le chemin minimal entre les villes de façon à trouver le meilleur compromis entre précision et rapidité. Nous avons alors appliqué différentes méthodes algorithmiques : la méthode du plus proche voisin, la méthode de l’insertion et la méthode de la descente locale [3]. Nous avons également codé l’algorithme du recuit simulé [4] qui s’inspire de la configuration des atomes dans un métal lorsqu’il refroidit. Ces méthodes d’approximation ont l’avantage d’avoir un coût temporel peu élevé.

Puis, après avoir comparé les différents résultats renvoyés par les algorithmes, nous avons étudié leur complexité. Nous les avons estimées à l’aide d’une régression polynomiale [5] pour mettre en évidence le degré du coût temporel. Cela nous a permis d’avoir une représentation graphique de leur durée d’exécution.

Enfin, nous avons étendu notre étude au théorème de Bearwood-Halton-Hammersley [6]. Nous nous sommes de fait intéressé à la taille du chemin le plus court dans le cas où le nombre de villes tend vers l’infini. Nous nous sommes aidés de la méthode de Monte-Carlo [7] pour vérifier expérimentalement le théorème.

**Problématique retenue (50 mots)**

Nous établirons différents algorithmes pour déterminer ou approcher la distance totale la plus courte possible entre les villes. Ensuite nous comparerons leur complexité temporelle ainsi que les solutions qu’ils renvoient.

**Objectifs du TIPE (100 mots)**

PARTIE COMMUNE

1)Modélisation informatique du problème : représentation du graphe

2)Réalisation des algorithmes « basiques » : algorithmes plus proche voisin, insertion et descente locale

3) Démonstration mathématique de certains résultats

PARTIE Partenaire 1

4) Réalisation de l’algorithme d’explosion combinatoire

5) Etude de l’efficacité et de la précision des algorithmes pour un nombre de villes qui tend vers l’infini

PARTIE Partenaire 2

6) Réalisation du recuit simulé

7) Comparaison du coût temporel des algorithmes

Ma PARTIE

8) Réalisation d’algorithme génétique, avec différents opérateurs de sélection (croisement)

9) Comparaison des différents algorithmes (complexité temporelle, meilleure solution)

**Références bibliographiques (5 à 10)**

[1] TSP - Infrastructure for the Traveling Salesperson Problem

<https://www.jstatsoft.org/index.php/jss/article/view/v023i02/v23i02.pdf>

(Michael Hahsler, Kurt Hornik)

[2] Arbre Recouvrant de Poids Minimal

<http://www.irem.univ-bpclermont.fr/IMG/pdf/3Recouvrant.pdf>

(Philippe Lac,Malika More)

[3] [Le voyageur de commerce](https://interstices.info/auteur/jean-charles-billaut/)

<https://interstices.info/jcms/c_37686/le-probleme-du-voyageur-de-commerce>

([Yifang Li](https://interstices.info/auteur/yifang-li/), [Yannick Kergosien](https://interstices.info/auteur/yannick-kergosien/), [Jean-Charles Billaut](https://interstices.info/auteur/jean-charles-billaut/))

[4] Recuit simulé

Modélisation de phénomènes aléatoires: introduction aux chaînes de Markov et aux martingales

(Thierry Bodineau)

[5] Régression polynomiale avec Scikit-learn

[https://darques.eu/blog/index.php/2017/05/09/regression-polynomiale-avec-scikit-learn](https://darques.eu/blog/index.php/2017/05/09/regression-polynomiale-avec-scikit-learn/)

(Michaël)

[6] Recueil de modèles aléatoires

<https://hal.archives-ouvertes.fr/hal-01897577v2/document>

(Djalil Chafai, Florent Malrieu)

[7] Méthode de Monte-Carlo

<http://cermics.enpc.fr/~bl/PS/SIMULATION-X/poly-monte-carlo-x.pdf>

(Laure Elie et Bernard Lapeyre)

**DOT**

* [Septembre à décembre 2018] Implémentation des algorithmes dits classiques permettant de donner des solutions approchées et exactes au problème de voyage de commerce
* [Janvier 2019] Réalisation des graphes début octobre permettant de visualiser les villes ainsi que les chemins les reliant
* (Février 2019] Regroupement des différentes méthodes algorithmiques dans un même code pour comparer le chemin qu’elles renvoient
* [Mars 2019) Echec de l’emploi de l’algorithme de Dijtskra pour la détermination du plus court chemin
* [Avril 2019] Compréhension et implémentation des algorithmes génétiques avec différents opérateurs de sélection pour résoudre le problème du voyageur de commerce

**ABSTRACT**

In the Travelling Salesman Problem, algorithms that allow to calculate the shortest path are very costly in terms of computation time. Therefore, we decided to code more efficient optimization algorithms.

My partners applied algorithms based on thermodynamics and artificial intelligence. As for me, I used genetic algorithms inspired by Charles Darwin’s theory of natural evolution. These algorithms reflect the process of natural selection where the fittest individuals are selected for reproduction. It was necessary to determine the value of many parameters in order to optimize the solution.