Université Libre de Bruxelles
Master informatique parcours "Optimisation en Recherche Opérationnelle (ORO)"
Année académique 2018-2019

Rapport de Projet

The Symmetric Traveling Salesman Problem (STSP)

A.Gontier, C.Pelhâtre, M.Ocquident, M.Latif

Résumé

L'objectif de ce projet est d'étudier différentes méthodes de résolution pour le Symmetric Travelling Salesman Problem (STSP) aussi bien exactes qu'heuristiques. En effet, ce problème appartient à la classe $\mathcal{NP}$-complet et peut nécessiter l'utilisation de méthodes d'approximation pour sa résolution sur des instances de grandes tailles. Après avoir présenté les différentes modélisations mises place, nous présenterons les résultats expérimentaux obtenus.

Table des matières

1 Formulation exacte du STSP

Dans le cas du STSP les villes et les routes les reliant sont représentées dans un graphe complet $G = (V, E) \in K_{|V|}$ dont la matrice des coûts est symétrique *i.e.* que le coût pour aller d'une ville a vers b est le même que pour aller de b vers a. La résolution du STSP consiste à trouver un cycle Hamiltonien de coût minimum dans le graphe G.

On définit les variables suivantes :

$$x_{ij} = \begin{cases} 1 & \text{Si l'arête } (i,j) \text{ est selectionée dans une solution} \\ 0 & \text{sinon} \end{cases}$$

c_{ij} est le coût de l'arête (i, j) $\forall i, j \in V$.

Le Symmetric Travelling Salesman Problem (STSP) se décrit comme suit :

$$\min \sum_{i,j \in V} c_{ij} x_{ij} \tag{1}$$

$$\text{s.c} \sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \tag{2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \tag{3}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \qquad \emptyset \neq S \subseteq V, |S| \geq 2 \tag{4}$$

$$x_{ij} \in \{0,1\} \ \forall i, j \in V \tag{5}$$

On remarque qu'il y a trois groupes de contraintes :
— Les contraintes qui forcent chaque sommet à avoir un seul arc entrant, représentées par les équations (2)
— Les contraintes qui forcent chaque sommet à avoir un seul arc sortant, représentées par les équations (3)
— Les contraintes permettant de contraindre la solution à un cycle Hamiltonien sont représentées par les inéquations (4). Ces dernières, dont le nombre croît très rapidement, contraint tous les sous-tours possibles, et ce nombre de sous-tours est exponentiel en $|V|$.

2 Modélisation de Dantzing, Fulkerson et Johnson

Pour ne pas avoir à prendre en compte toutes les contraintes de sous-tours, qui sont exponentielles en n le nombre de sommets de G, La première méthode propose de simplement les relâcher pour réaliser une première résolution et obtenir un tour. Dans un second temps, nous tentons de vérifier si la solution obtenue est un cycle Hamiltonien. Si le tour obtenu vérifie cette propriété, alors on en déduis que la résolution a fournie une solution optimale. Dans le cas contraire, cela signifie que le tour obtenu contient un ou plusieurs sous-tours ; dans ce cas, on ajoute alors une contrainte afin de casser le premier sous-tour trouvé. En notant S l'ensemble des premiers sous-tours détectés, on obtient le problème suivant :

$$\min \sum_{i,j \in V} c_{ij} x_{ij} \tag{6}$$

$$\text{s.c} \sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \tag{7}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \tag{8}$$

$$\sum_{i,j \in S} x_{ij} \leq |s| - 1 \qquad \forall s \in S \text{ avec } |s| \leq (n-1) \tag{9}$$

$$x_{ij} \in \{0,1\} \ \forall i, j \in V \tag{10}$$

En réalité, Nous ajoutons deux contraintes car nous souhaitons contraindre le sous-tour dans les deux sens. Ensuite, on résout une nouvelle fois le problème en considérant l'ajout de ces deux nouvelles contraintes.
L'algorithme consiste en la répétition des opérations de résolution et d'ajout de contraintes cassantes de sous tours. Ce dernier se termine lorsqu'il n'existe plus aucun sous-cycle dans la solution, ce qui correspond à la solution optimale.
Ainsi, la résolution par la méthode d'ajouts successifs de contraintes cassantes de sous tour est plus rapide que si l'on avait conservé la contrainte initiale.
L'inconvénient présenté par cette méthode est qu'elle nécessite de résoudre à plusieurs reprises le même sous-problème (à l'ajout de contraintes prêt). Le pire des cas que peut rencontré l'algorithme est lorsqu'il doit ajouter une à une toutes les contraintes de sous-tours qui sont exponentielles en fonction du nombre de sommets. En pratique, le nombre de sous-tours à casser n'est pas très grand par rapport au nombre de sous-tours possibles, ce qui nous permet d'avoir une résolution plus rapide qu'en écrivant toutes les contraintes.

3 Modélisation de Miller, Tucker et Zemlin

Dans cette modélisation, les contraintes de sous-tours sont modélisées différemment ; Elle propose d'ajouter des variables u entières sur chaque sommet et une contrainte polynomiale en la taille des données, n correspondant au nombre de sommets :

$$\min \sum_{i,j \in V} c_{ij} x_{ij} \tag{11}$$

$$\text{s.c} \sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \tag{12}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \tag{13}$$

$$(n-1)x_{ij} + u_i - u_j \leq (n-2) \qquad \forall i, j \in V, i \neq j, i \neq 1, j \neq 1 \tag{14}$$

$$x_{ij} \in \{0,1\} \ \forall i,\ j \in V, u \in \mathbb{Z} \tag{15}$$

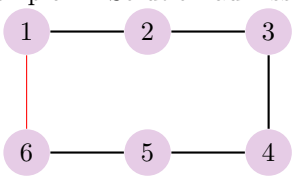
Cette formulation compacte [1] permet de résoudre le problème du STSP avec n'importe quel solveur IP.

Si l'arc (i, j) est dans la solution, la variable x_{ij} vaut 1, et la contrainte (14) peut donc s'écrire comme suit :

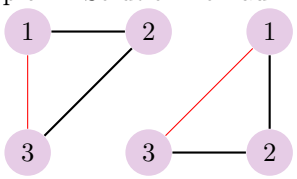
$$u_i \leq u_j - 1, \qquad i, j \in V, i \neq j, i \neq 1, j \neq 1.$$

Les variables u sont entières donc on a $u_i < u_j$. On pourrait interpréter ces variables u comme suit : Pour tout arc (i, j) appartenant à une solution du problème, on a sur le sommet i, une valeur entière u_i et sur son successeur j, une valeur u_j telle que $u_i < u_j$ sauf pour le sommet 1. Comme le sommet 1 est exclu, on peut réaliser un chemin où les variables u_i augmentent sur chaque sommet (Exemple 1). Ainsi, le prédécesseur de 1 aura la valeur u la plus élevée du tour, et le successeur de 1 aura la plus petite valeur u du tour. Les sous-tours sont impossibles car on aurait alors un deuxième sommet dans le cas de 1 c'est a dire un sommet qui à un arc (i, j) tel $u_i \geq u_j$ (Exemple 2). Et comme cette modélisation conserve les contraintes (2) et (3) qui obligent tous les sommets à être visités, nous sommes assurés d'obtenir un cycle Hamiltonien.
Exemples : les valeurs dans les noeuds sont les valeurs des variables u_i

Exemple 1 : Solution admissible



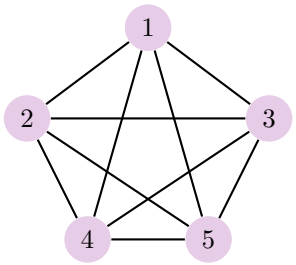
Exemple 2 : Solution non admissible



(On peut simplement dire qu'une seule arrête rouge est autorisée)

4 Heuristique de Christofides

L'algorithme de Christofides est une heuristique qui résout le STSP à au plus 3/2 l'optimal si le problème respecte l'inégalité triangulaire. Il se déroule comme suit :



Graphe complet à 5 sommets

1. On nomme compacte une formulation qui contient un nombre polynomial de variables et de contraintes en la taille des données du problème

— On calcul T l'arbre couvrant de poids minimum. (Exemple Figure 1)

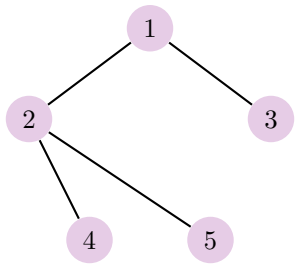


FIGURE 1 – Arbre couvrant de poids minimum

— On met dans un sous-graphe les sommets de cardinalité impaire de T et on calcul M le couplage parfais de poids minimum dans ce sous-graphe. (Exemple Figure 2)

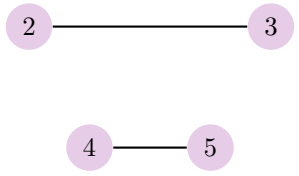


FIGURE 2 – Couplage parfait de poids minimum dans le sous-graphe des sommets impairs du MST

— On cherche le circuit Eulérien de poids minimum dans l'union de l'arbre T et du couplage M
— On construit le circuit Hamiltonien à partir du circuit Eulérien. Pour ceci, on parcourt le circuit Eulérien et quand on retombe sur un sommet que l'on à déjà vu, on le saute et on relie les deux sommets qui sont à sa gauche et sa droite ensemble. Ceci est possibles grâce à l'inégalité triangulaire en hypothèse. (Exemple Figure 3), on remplace les deux arrêtes rouges du circuit Eulérien par une arrête verte pour construire le circuit Hamiltonien.

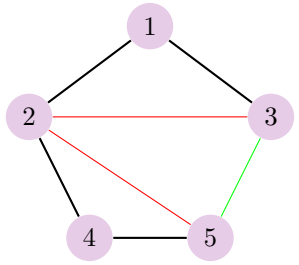


FIGURE 3 – Transformation du circuit Eulérien vers le circuit Hamiltonien

4.1 Preuve de la propriété d'approximation de l'algorithme de Christofides

Comme nous l'avons présenté précédemment, l'algorithme de Christofides consiste en la résolution successive des sous problèmes suivants :
Soit $G = (V, E)$, un graphe non orienté et une matrice des coûts symétrique $D = [d_{ij}]$ $\forall i, j \in |V|$ associée à ce graphe respectant l'inégalité triangulaire. Les sous problèmes s'énoncent comme suit

Sous problème 1 : Recherche d'un arbre recouvrant de poids minimal ou MST [2] noté T.
Posons S l'ensemble des sommets de degré impair de T

Sous problème 2 : Recherche d'un couplage parfait de poids minimum M dans le sous graphe induit par S noté G *i.e.* contenant toutes les arêtes ayant leurs deux extrémités dans S.

Sous problème 3 : Construction d'un cycle Eulérien C à partir des arêtes dans l'ensemble $T \cup M$.

Sous problème 4 : Construction d'un cycle Hamiltonien à partir des du cycle C en effectuant la transformation suivante :

2. Minimum spanning tree

Si un noeud v est visité deux fois, remplacer les arc (u, v) et (v, w) par un arc direct (u, w)

Définition Cycle Eulérien
Dans un graphe connexe, un cycle Eulérien est un chemin simple qui passe par tous les arcs du graphe.

Définition Cycle Hamiltonien
Dans un graphe connexe, un cycle Hamiltonien est un chemin qui passe une seule et unique fois par chacun des sommets du graphe

L'algorithme de Christofides est un algorithme d'approximation possédant les propriété suivante :

Théorème (Approximation de l'algorithme de Christofides)**.** *L'algorithme de Christofides garantie une solution qui est dans le pire cas inférieure à $\frac{3}{2}$ fois la solution optimale.*

Les éléments de preuve pour ce théorème sont proposés dans l'ouvrage de Papadimitriou et Steiglitz[1] ainsi que dans celui de Wolsey[2]. Nous allons vous présenter les grandes ligne de cette dernière qui nous permettent de comprendre pourquoi cet algorithme fourni une bonne approximation de la solution.

Éléments de preuve

Tout d'abord, nous rappelons les conditions que doit vérifier le Graphe G pour être résolu par cette méthode :

Conditions générales sur le problème du STSP Soit $G = (V, E)$, un graphe connexe non orienté possédant une matrice des coûts $D = [d_{i,j}] \in \mathcal{M}_{|V|\times|V|}(\mathbb{R}_+)$ $\forall i, j \in V$ La matrice D doit également vérifier les conditions suivantes :

- La matrice D est symétrique *i.e.* $\forall i, j \in V$, $d_{ij} = d_{ji}$ - Dans notre cas, cela signifie que la distance séparant la ville i de la ville j est la même que celle entre les villes j et i.
- $d_{jj} = 0$ $\forall j \in V$ - Dans noter cas, cela signifie que le trajet entre la même ville est de distance nulle.
- D respecte l'inégalité triangulaire *i.e.* $\forall\ 0 \leq i, j, k \leq 1$

$$d_{ij} + d_{jk} \geq d_{i,k}$$

  Dans notre cas, nous pouvons interpréter cette condition comme le fait qu'un chemin allant directement de la ville i à la ville k sera toujours inférieur à une chemin passant par les villes i, j et enfin k.

Notation : Dans la suite du rapport et conformément à la littérature étudiée, le problème du TSP symétrique satisfaisant l'inégalité triangulaire sera appelé ΔTSP.

Appartenance à la classe $\mathcal{NP}$-Complet Comme nous l'avons évoqué dans le début de ce rapport, le problème du ΔTSP est appartient à la classe des problèmes $\mathcal{NP}$-complet. Cela peut se prouver en réalisant une réduction polynomiale du problème de décision du cycle Hamiltonien vers le problème de décision du ΔTSP.

Nom	Cycle Hamiltonien (CH)
Instance	Un graphe $G = (V, E)$
Question	Existe-t-il un cycle Hamiltonien *i.e.* un cycle passant une fois et une seule par chaque sommet.

TABLE 1 – Formatlsation du problème de décision du circuit Hamiltonien - Source : Complexité Algorithmique - G.Fertin, J.X.Rampon (Université de Nantes)

Nom	TSP symétrique vérifiant l'inégalité triangulaire (ΔTSP)
Instance	Un graphe $G = (V, E) \in K_{\|V\|}$, un poids c_e sur chaque arc $e \in E$
Solution	Un cycle Hamiltonien CH dans G
Mesure	La longueur de CH : $L_{CH} = \sum_{e\in CH} c_e$

TABLE 2 – Formatlsation du problème d'optimisation du ΔTSP - Source : Complexité Algorithmique - G.Fertin, J.X.Rampon (Université de Nantes)

Théorème (Classe de complexité de Cycle Hamiltonien)**.** *Le problème de décision CH appartient à la classe $\mathcal{NP}$-Complet*

En opérant la réduction polynomiale CH $\leq_P$ ΔTSP, *i.e.* transformant une instance de CH en une entrée pour notre problème en temps polynomial, on peut montré que ΔTSP est $\mathcal{NP}$-complet.

Démonstration. La preuve est laissée au lecteur à titre d'exercice □

Cette précision, pouvant être considérée comme *triviale* s'avère intéressante pour comprendre la dernière opération de l'algorithme de Christofides.

Multigraphes, couplage parfait de poids minimum et arbre couvrant de poids minimum Lors de la résolution du problème, nous avons eu besoin d'utiliser la structure Python de multigraphe ; La définition de cette classe de graphe est la suivante :

Définition Multigraphe
Un multigraphe est un graphe permettant l'existence de plusieurs arêtes entre deux même sommets

Définition Multigraphe Eulérien
Un multigraphe $G = (V, E)$ est Eulérien si et seulement si G est connexe et l'ensemble des noeuds de V possèdent un degré pair.

La structure du multigraphe est importante dans la résolution du ΔTSP et notamment dans l'étape concernant le sous problème du couplage parfait de poids minimum M. En effet dans ce dernier, il est possible de trouver des arcs qui auront déjà été identifiés dans le problème de l'arbre couvrant de poids minimum T ; La redondance de certaines arêtes peut s'avérer importante dans le sous problème de détection du circuit Eulérien de poids minimal.

On peut également définir les éléments de théorie des graphes suivants utilisés dans les différentes étapes de l'algorithme Christofides :

Définition Arbre couvrant de poids minimal
Soit $G = (V, E)$ un graphe non orienté. Un arbre couvrant de G est un sous-graphe couvrant de G qui est un arbre, *i.e.* un sous-graphe couvrant de G qui est à la fois connexe et sans cycles simples.
Soit $G = (V, E)$ un graphe non orienté à valuations quelconques. Un arbre couvrant de poids minimal de G est un arbre couvrant de G dont la valuation est la plus petite parmi celles de tous les arbres couvrants de G.

Définition Couplage parfait de poids minimum
Soit $G = (V, E)$ un graphe simple. On appelle couplage $M \subseteq E$ un ensemble d'arêtes tel que deux arêtes quelconques de ne sont pas adjacentes.
Un couplage est parfait s'il sature tous les sommets du graphe.
Le couplage parfait de poids minimum M est un couplage dont la somme des poids des arêtes le composant est minimale.

Approximation de Christofides

Théorème. *L'algorithme de Christofides est une $\frac{1}{2}$-approximation du Δ TSP.*

Note : En d'autres termes, l'algorithme de Christofides propose une approximation de la solution qui n'excède pas $1 + \frac{1}{2}$ fois la valeur de la solution optimale pour le problème du ΔTSP.

Démonstration. La preuve de ce théorème résulte de l'enchaînement des opérations effectuées par l'algorithme. Notons d'abord que le graphe $G = (V, T \cup M)$ est eulérien ; En effet, dans le cas :
— un noeud possède un degré pair dans T, il aura ce même degré dans le graphe G.

— un noeud possède un degré impair en T, alors cela signife qu'il possède une arête incidente qui est incluse dans le couplage M

Notons de plus que ce graphe est connexe car il contient les arêtes de l'arbre recouvrant (contenant et connectant tous les sommets par définition).

Posons τ, un tour optimal pour le ΔTSP *i.e.* une séquence de sommet parcourus et minimisant le coût total du problème noté $c(\tau)$. L'application de l'algorithme de Christofides pour l'obtention d'un graphe G correspond à la somme des coûts des arêtes de M et T et le tour obtenu n'est pas nécessairement optimal. Ce faisant, on peut établir l'inégalité suivante :

$$c(\tau) \leq c(G) = c(T) + C(M)$$

Posons $\hat{\tau}$ comme le cycle obtenu dans le graphe par application de l'algorithme ; On obtient l'inégalité suivante

$$c(T) \leq c(\hat{\tau})$$

Posons $S = \{i_j\}_{j \in [\![1,n]\!]}$, l'ensemble des sommets de degré impair de T ordonné selon leurs apparition dans $\hat{\tau}$. Considerons maintenant deux couplages des noeuds de degré impair de T :

$$M_1 = \{[i_1, i_2], [i_3, i_4], \ldots, [i_{2m-1}, i_{2m}]\} \text{ et } M_2 = \{[i_2, i_3], [i_4, i_5], \ldots, [i_{2m}, i_1]\}$$

correspondant à un cycle pour le ΔTSP. Par application de l'inégalité triangulaire respectée par les instances de ce problème, on peut en déduire l'inégalité suivante :

$$c(\hat{\tau}) \geq c(M_1) + C(M_2)$$

Le couplage M étant le couplage optimal de poids minimum, on peut écrire :

$$c(\hat{\tau}) \geq 2 \times c(M) \iff c(M) \leq \frac{1}{2} c(\hat{\tau})$$

En substituant les inégalités trouvées précédemment dans celle du tour de coût minimal τ, obtient :

$$c(\tau) \leq c(G) = c(T) + C(M) \leq c(\hat{\tau}) + \frac{1}{2} c(\hat{\tau})$$

On peut ainsi conclure avec la relation suivante :

$$c(\tau) \leq \frac{3}{2} c(\hat{\tau})$$

Ce qui met fin à la preuve de l'approximation d'une solution obtenue par l'algorithme de Christofides. □

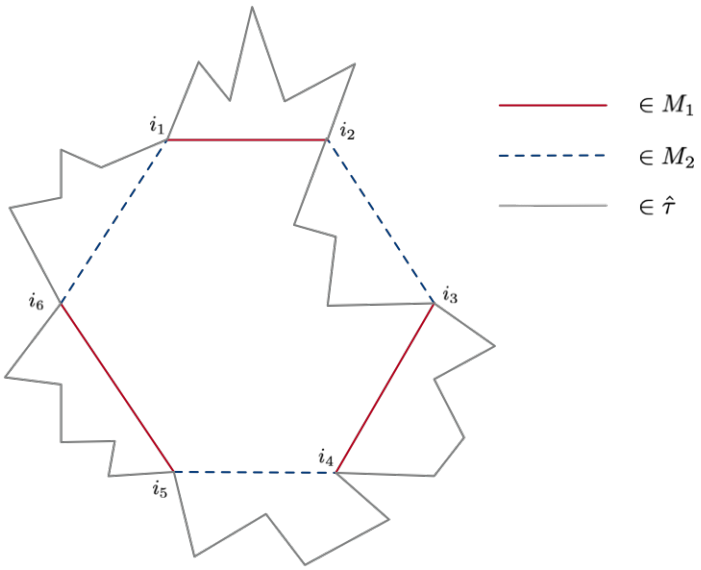


FIGURE 4 – Représentation graphique de la construction opérée dans l'algorithme de Christofides - Visualisation de l'inégalité triangulaire.

5 Comparaison des performances des modèles

L'ensemble des méthodes ont été réalisé en Python 3 et les tests ont été effectués avec un pc acer sous ubuntu18 avec les caractéristiques suivantes :

— RAM : 4 Gio
— CPU : Intel® Core^{TM} i3-7130U CPU @ 2.70GHz × 4
— GPU : Intel® HD Graphics 620 (Kaby Lake GT2)

Les résolution ont été faites avec un temps limite de 5 minutes pour les deux méthodes exactes (noté : "Interrompue").

On récupère la valeur de la solution Z et le temps d'exécution de la méthode. Ceci pour les trois méthodes :

— Modèle 1 : Résolution exacte avec la modélisation de Dantzing, Fulkerson et Johnson
— Modèle 2 : Résolution exacte avec la modélisation de Miller, Tucker et Zemlin
— Christophides : Heuristique de Christophides. Vous trouverez dans la colone Approx, la qualité de l'heuristique obtenue. Cette dernière est calculé par un ratio avec la solution de la résolution 2. On rappelle que ce ratio ne peut pas dépasser 1.5

Le détail de l'exécution est indiqué dans le fichier results.txt Nous obtenons des solutions non-admissibles lors d'une exécution interrompue du modèle 1 c'est pourquoi nous précisons toujours la taille du circuit obtenu. Pour ce qui est du modèle 2, aucun résultat n'est renvoyé lors d'une interruption ("null").

Instance	Modèle 1		Modèle 2		Christophides		
	Z	temps	Z	temps	Z	temps	Approx
tsp10_0	275.260305	208.611 s	275.260305	0.134 s	293.996231	0.009	1.07
tsp10_1	237.590061	21.874 s	237.590061	0.152 s	237.590061	0.008 s	1.0
tsp10_2	227.273317	34.105 s	227.273317	0.121 s	229.359251	0.003 s	1.01
tsp10_3	207.387869	0.524 s	207.387869	0.13 s	216.433102	0.003 s	1.04
tsp10_4	337.660256	5.92 s	337.660256	0.146 s	337.660255	0.003 s	1.0
tsp10_5	258.448011	4.525 s	258.448011	0.126 s	288.608243	0.002 s	1.12
tsp10_6	297.888304	10.542 s	297.888304	0.137 s	297.888304	0.007 s	1.0
tsp10_7	339.335008	5.658 s	339.335008	0.15 s	339.9102	0.012 s	1.0
tsp10_8	322.961537	4.349 s	322.961537	0.249 s	365.822034	0.01 s	1.13
tsp10_9	345.597235	0.821 s	345.597235	0.15 s	372.80066	0.007 s	1.08
tsp25_0	367.928329	Interrompue	415.870874	10.195 s	453.960140	0.011 s	1.09
tsp25_1	442.768001	Interrompue	461.219792	5.386 s	504.044163	0.007 s	1.09
tsp25_2	418.409279	Interrompue	null	Interrompue	499.125904	0.008 s	NA
tsp25_3	423.336307	Interrompue	461.489977	23.712 s	531.016394	0.013 s	1.15
tsp25_4	388.363921	Interrompue	null	Interrompue	486.489861	0.015 s	NA
tsp25_5	401.302691	Interrompue	432.422439	1.083 s	483.63397	0.015 s	1.12
tsp25_6	388.408411	Interrompue	407.480777	28.728 s	456.38298	0.011 s	1.12
tsp25_7	364.241358	Interrompue	null	Interrompue	473.23481	0.018 s	NA
tsp25_8	393.6104	Interrompue	398.427279	0.582 s	403.315156	0.012 s	1.01
tsp25_9	432.572818	Interrompue	451.468695	1.591 s	480.316064	0.011 s	1.06

Exécution des 20 premières instances avec un temps limite de 5 min

On observe que la méthode de cassage des sous-tours prend énormément de temps pour trouver la solution optimale, surtout dans les pires cas (lorsqu'on a beaucoup de sous-tours à casser). C'est sans appel la méthode la plus lente. Par ailleurs, cette méthode a l'avantage de renvoyer un sous tour lorsqu'on l'interrompt.

La deuxième modélisation trouve les solutions exactes beaucoup plus rapidement en moyenne, mais la résolution peut prendre plus de 5 minutes pour certaines instances à 25 villes ou plus.

L'heuristique de Christofides est beaucoup plus rapide que les résolutions exactes (100 à 1000 fois plus rapide que la modélisation 2 par exemple). Et bien heureusement, elle respecte l'approximation de $\frac{3}{2}$ prouvée ci-dessus.

6 Conclusion

En conclusion, on peut dire que l'idée de casser les sous-cycles devient très mauvaise lorsque la taille du problème augmente car le nombre de sous-cycles à casser suit cette augmentation de manière exponentielle. Pour palier à cette faiblesse, on pourrait améliorer l'algorithme en ajoutant des choix heuristiques mais la complexité dans le pire des cas resterait exponentielle.

La deuxième modélisation est une idée intéressante et efficace pour trouver une solution optimale. Elle permet de résoudre facilement des instances de moyenne et petite tailles. Mais le STSP reste un problème difficile et il faut plus de temps pour que les résolutions des grandes instances terminent.

L'heuristique de Christophides est une résolution beaucoup plus rapide que ses homologues exactes du fait qu'elle ne soit justement pas exacte. Elle permet de se faire très rapidement un bon aperçu de la solution optimale. En plus, elle a la propriété sympathique d'avoir une approximation prouvée à $\frac{3}{2}$ de la solution optimale.

Références

[1] Christos H. Papadimitriou et Kenneth Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Englewood Cliffs, NJ : Prentice Hall, 1982. isbn : 0131524623 9780131524620 0486402584 9780486402581.

[2] L.A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998. isbn : 9780471283669.