Heuristiques du problème du voyageur de commerce

TADUNFOCK TETI Bernard* — FOTSO Laure Pauline**

* Département d'informatique
Université de Yaoundé I
Ville Yaoundé
Pays Cameroun
bernardteti@yahoo.fr

** Département d'informatique
Université de Yaoundé I
Ville Yaoundé
Pays Cameroun
l_fotso@yahoo.com

RÉSUMÉ. Le problème du voyageur de commerce (TSP) est un problème classique d'optimisation combinatoire NP-complet. Des heuristiques ont été proposées permettant de trouver des solutions presque optimales. On distingue deux classes d'heuristiques: de construction et d'amélioration. Lin-Kernighan est la meilleure heuristique d'amélioration. L'heuristique de colonies de fourmis est une méthode composite de construction et d'amélioration. Nous proposons une hybridation de ces deux méthodes. L'étude expérimentale sur des instances de TSP symétriques de nombre de villes variant entre 17 et 2392 montre que cette hybridation est meilleure que les deux autres approches hybrides de la littérature à savoir (Algorithme Génétique-Lin-Kernighan) et (Colonie de fourmis-3-*opt*).

ABSTRACT. The travelling salesman problem (TSP) is NP-hard classical optimisation problem. There are heuristics that give near optimal solutions. These heuristics are classified into two categories: for construction and for improvement. Lin-Kernighan is known as the best construction heuristic while Colony of Ants is composite mixing construction and improvement. We propose to hybridize these two methods Experimental study on symmetric TSP instances from the TSPLIB containing between 17 and 2392 cities shows that our hybrid heuristic gives better solutions (nearer to the optimum solution) than two others hybrid heuristics: Genetic Algorithm with Lin-Kergnighan and Colony of Ants with 3-*opt*.

MOTS-CLÉS : TSP, heuristique, Lin-Kernighan, colonie de fourmis, algorithme génétique

KEYWORDS : TSP, heuristic, Lin-Kernighan, colony of ants, genetic algorithm

1. Introduction

Un voyageur de commerce désire visiter un certain nombre de villes, débutant et finissant son parcours dans la même ville en visitant chacune des autres villes une et une seule fois. Il désire sélectionner la tournée qui minimise la distance totale parcourue. Ce problème est connu sous le nom du problème du voyageur de commerce (en anglais *travelling salesman problem* : TSP) et est NP-Complet. La complexité en temps des algorithmes exacts proposés croît exponentiellement avec n (la taille du problème ou le nombre de villes). Plusieurs méthodes d'approximation (heuristiques [1]) ont été proposées qui approchent en temps raisonnable la solution optimale.

Les heuristiques peuvent être réparties en trois classes : les heuristiques de construction, les heuristiques d'amélioration et les algorithmes composés (qui combinent les deux premières). Les heuristiques de construction élaborent graduellement la tournée en ajoutant une ville (noeud) à chaque étape. Elles s'arrêtent dès que la solution est trouvée et n'essayent pas de l'améliorer. Dans cette catégorie il y a l'heuristique du plus proche voisin, l'algorithme glouton, l'algorithme de Christofides. Les heuristiques d'amélioration 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* [6] et 3-*opt* [6] sont des exemples les plus communément utilisés. Dans cette catégorie, il y a également l'algorithme de Lin-Kernighan [5], la recherche tabou [4], le recuit simulé [8] et les algorithmes génétiques [3]. L'heuristique de colonie de fourmis combine les caractéristiques des deux classes. Il existe des hybridations de ces heuristiques parmi lesquelles l'hybridation de Freisleben et Mertz [2] et de Dorigo et Gambardella [1] qui combinent respectivement algorithme génétique avec Lin-Kernighan et colonie de fourmis avec 3-*opt*. Nous effectuons une étude comparative de certaines de ces heuristiques que nous avons implémentés : recuit simulé, algorithme génétique, algorithme de colonies de fourmis, algorithme de Lin-Kernighan, les deux algorithmes hybrides ci-dessus cités et l'algorithme hybride de colonies de fourmis avec l'algorithme de Lin-Kernighan que nous proposons.

Le reste de l'article se présente comme suit : à la section 2 nous définissons le problème ; à la section 3 nous présentons quelques heuristiques ; notre heuristique hybride est présenté à la section 4 avec les résultats expérimentaux ; la section 5 conclut.

2. Définition du TSP

Un voyageur de commerce doit visiter n villes données en passant par chaque ville exactement une fois. Il commence par une ville quelconque et termine en retournant à la ville de départ. Les distances entre les villes sont connues. Il faut trouver le chemin qui minimise la distance parcourue. La notion de distance peut-être remplacée par d'autres notions comme le temps qu'il met ou l'argent qu'il dépense.

Formellement, à partir d'une matrice $C = (c_{ij})$ où c_{ij} représente le coût du déplacement de la ville i à la ville j $(1 \leq i, j \leq n)$, il faut trouver une permutation

1. Une heuristique est une méthode, conçue pour un problème d'optimisation donné, qui produit une solution non nécessairement optimale lorsqu'on lui fournit une instance de ce problème.

$\begin{pmatrix} 1 & 2 & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(n) \end{pmatrix}$ qui minimise la somme $\sum_{i=1}^{n-1} c_{\sigma(i)\sigma(i+1)} + c_{\sigma(n)\sigma(1)}$ (le coût d'une tournée est égale à la somme des coûts de tous les arcs appartenant à la tournée). Autrement dit, il faut trouver un circuit (respectivement cycle) Hamiltonien de longueur minimale dans un graphe orienté (respectivement non orienté) valué complet. Si $c_{ij} = c_{ji}$ pour tout $i \in \{1, \dots, n\}$ le problème est dit symétrique sinon il est asymétrique.

Le TSP peut être formulé comme un problème de programmation linéaire en nombre entier. Il a des applications directes dans le transport, la logistique, etc. Par exemple trouver le chemin le plus court pour les bus de ramassage scolaire ou, pour le camion de ramassage des ordures. Le TSP présente plusieurs caractéristiques communes aux problèmes d'optimisation combinatoire et fournit ainsi une plate-forme idéale pour étudier les méthodes générales applicables à un grand nombre de ces problèmes.

3. Méthodes de résolution

Les algorithmes de résolution du TSP peuvent être répartis en deux classes :

– Les algorithmes exacts permettent de trouver la solution optimale, mais leur complexité est exponentielle. Les algorithmes les plus efficaces sont "*cutting-plane*", "*facet-finding*" et "*branch and bound*".

– Les algorithmes d'approximation (heuristiques) obtiennent de bonnes solutions mais ne donnent aucune garantie sur l'optimalité de la solution trouvée. Nous présentons quelques un de ces heuristiques.

3.1. Algorithme de Lin-Kernighan

L'algorithme commence avec une tournée admissible donnée ; cherche ensuite dans le voisinage de la solution courante défini par l'opération λ-*opt move* toute tournée améliorant la configuration courante. A chaque étape de l'itération, l'algorithme examine, pour des valeurs croissantes de λ (à partir de 2) si l'échange de λ liens produit une tournée plus courte. L'algorithme continue ainsi jusqu'à ce qu'aucune amélioration ne soit plus possible. L'opération λ-*opt move* consiste à supprimer λ liens (arêtes) et à reconnecter les segments restants par de nouveaux liens, en renversant si possible le sens de parcours d'un ou de plusieurs de ces segments. Plus la valeur de λ est grande, plus la solution finale est proche de l'optimum et plus le temps d'exécution devient élevé. En général on utilise des valeurs entières de $\lambda \in \{2, 3, 4, 5\}$. La Figure 3.1 illustre l'opération *2-opt move*.

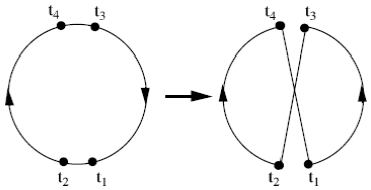

Figure 3.1 : *2-opt move*

3.2. Les algorithmes génétiques (*Genetics algorithms* : GA)

On part d'une population (ensemble) de N permutations de villes solutions du problème représentées par des individus (tournées). Cette population choisie aléatoirement est appelée population *parent*. Le degré d'adaptation (*Fitness* ou coût) d'un individu à l'environnement est exprimé par la valeur de la fonction de coût $f(x) = \sum_{i=1}^{n-1} c_{x(i)x(i+1)} + c_{x(n)x(1)}$, où x est la solution que l'individu représente. Un individu est d'autant mieux adapté à son environnement que le coût de la solution qu'il représente est plus faible. Au sein de cette population, intervient alors la *sélection* au hasard d'un ou de deux parents, qui produisent une nouvelle solution, à travers les *opérateurs génétiques*, tels que le *croisement* et la *mutation*. La nouvelle population, obtenue par le choix de N individus parmi les populations *parent* et *enfant*, est appelée génération suivante. En itérant ce processus, on produit une population plus riche en individus mieux adaptés.

3.3. Les colonies de fourmis (*Ant colonies system* : ACS)

Un ensemble d'agents, appelés fourmis, recherchent en parallèle une meilleure solution au TSP en coopérant indirectement à travers les phéromones qu'ils déposent sur le graphe du TSP. Dans le système de colonies de fourmis, m fourmis (agents) sont initialement positionnées sur n villes choisies par une règle d'initialisation (e.g., aléatoirement). Chaque fourmi construit une tournée (i.e., une solution admissible du TSP) en choisissant les villes à visiter par la règle de transition d'état. Pendant la construction de sa tournée, la fourmi modifie également la quantité de phéromone sur l'arc visité en appliquant la règle de mise à jour locale. Une fois que toutes les fourmis ont terminé leurs tournées, une certaine quantité de phéromones est déposée sur la plus courte tournée par la règle de mise à jour globale.

4. Proposition d'une heuristique hybride

En général, les heuristiques d'amélioration produisent de meilleurs résultats que les heuristiques de construction. L'approche générale d'hybridation consiste à utiliser une heuristique de construction pour générer une solution et appliquer ensuite une heuristique d'amélioration sur cette solution pour l'optimiser.

4.1. Justification

Il a été montré [7] qu'il est plus efficace d'alterner une heuristique d'amélioration avec des mutations de la dernière (ou meilleure) solution produite. Cette mutation consiste généralement à utiliser une heuristique d'optimisation locale. Les travaux de Freisleben et Merz [2] sont un exemple d'application de cette stratégie. L'algorithme génétique génère de nouvelles solutions qui sont localement améliorées par l'algorithme de Lin-Kernighan.

ACS comme l'algorithme génétique de Freisleben et Merz, est une heuristique d'amélioration qui, produit un ensemble de solutions qui sont des mutations de la précédente meilleure solution après chaque itération. L'étude expérimentale montre que ACS est plus efficace que GA et le recuit simulé. Il est donc raisonnable de penser que l'addition d'une heuristique d'optimisation locale à ACS peut le rendre plus compétitif que l'hybridation de Freisleben et Merz. D'où l'idée de M. Dorigo et L. Gambardella [1] qui ont proposé

une approche hybride qui consiste à appliquer l'heuristique d'optimisation locale 3-*opt* aux tournées produites par chaque fourmi avant l'application de la règle de mise à jour globale. Dans le but d'appliquer cette hybridation à la fois aux TSP symétriques et asymétriques, ils n'ont utilisé qu'une version restrictive de 3-*opt* (celle qui ne modifie pas le sens du parcours de la tournée après échange des arcs).

Etant donné que Lin-Kernighan est la meilleure méthode d'optimisation locale, une hybridation de ACS et Lin-Kernighan serait plus efficace que l'hybridation de Dorigo et Gambardella pour des problèmes symétriques.

4.2. Principe de ACS-LK

L'on dispose de m agents. Chaque agent k utilise ACS pour produire une tournée. Une fois que tous les agents ont construit leurs tournées, l'algorithme de Lin-Kernighan est appliqué à chacune de ces tournées en l'améliorant par des opérations de λ-*opt move* $\lambda \in \{2, 3, 4, 5\}$ jusqu'à ce qu'aucune amélioration ne soit plus possible.

4.3. Algorithme (ACS-LK)

1. Initialisation
2. **Répéter**
   - Chaque fourmi est positionnée à un nœud (ville de départ)
   - **Répéter**
     i. Chaque fourmi applique la règle de transition d'état pour se déplacer d'une ville à l'autre et construit ainsi une solution.
     ii. Chaque fourmi applique également la règle de mise à jour locale
   - **Jusqu'à** ce que chaque fourmi achève sa tournée
   - Appliquer la procédure de Lin-Kernighan aux solutions(tournées) obtenues par chaque fourmi.
   - Appliquer la règle de mise à jour globale
3. **Jusqu'à** la condition finale

4.4. Résultats expérimentaux

Nous avons implémenté les algorithmes ci-dessus en C et exécuté sur plusieurs(10) instances de TSP de tailles variant entre 17 et 2392 appartenant à la bibliothèque TSPLIB [9]. Le tableau 4.1 est un extrait de 5 de ces exemples. Chaque problème a été exécuté 10 fois sur chaque algorithme, les exécutions étant indépendantes. La condition d'arrêt de chaque exécution est un nombre de répétition (NBEssai) fixé à l'avance ou bien la valeur optimale du problème. Pour LK, NBEssai représente le nombre de répétitions de l'algorithme avec des tournées initiales différentes, pour GA et GA-LK, NBEssai représente le nombre de générations et pour ACS, ACS-3opt et ACS-LK, NBEssai représente le nombre de tournées effectuées par chaque fourmis.

Nous avons été guidés par les expérimentions effectuées dans la littérature (ACS[1], LK[5])pour choisir les valeurs des paramètre des heuristiques. Nous avons ensuite associé les meilleures valeurs de paramètre de ACS à ceux de LK pour obtenir les paramètres de l'heuristique ACS-LK. L'expérimentation avec d'autres paramètres ne donne pas de meilleures résultats. Nous avons donc retenu pour notre hybride ACS-LK les valeurs des paramètres suivantes : $\lambda \in \{2, 3, 4, 5\}$, NBEssai$= 500$, $m = 10$, $\tau_0 = (n.L_{nn})^{-1}$, $\beta = 2$,

$\alpha = \rho = 0.1$ et $q_0 = 0.9$. m représente le nombre de fourmis, τ_0 la quantité initiale de phéromones déposées sur tous les arcs, L_{nn} la longueur d'une tournée obtenue par une heuristique de construction, n le nombre de villes, β l'importance relative de la fonction heuristique sur les phéromones, $\alpha \in]0,1[$ le paramètre d'évaporation des phéromones dans la règle de mise à jour globale, $\rho \in]0,1[$ le paramètre d'évaporation des phéromones dans la règle de mise à jour locale et $q_0 \in [0,1]$ l'importance relative de l'exploitation par rapport à l'exploration dans la règle de transition d'état.

Problème	algorithme	Succès	Coût moyen	Tps moyen
lin318 (42029)	GA-LK	4	42097.4	1.6
	ACS-3*opt*	10	42029.0	1.0
	ACS-LK	10	42029.0	1.2
	LK	9	42040.4	1.3
att532 (27686)	GA-LK	10	27686	5.4
	ACS-3*opt*	2	27699.9	33.8
	ACS-LK	10	27686	4.6
	LK	9	27687.7	3.2
rat783 (8806)	GA-LK	10	8806	2.0
	ACS-3*opt*	10	8806	9.8
	ACS-LK	10	8806	1.1
	LK	10	8806	0.2
pr1002 (259045)	GA-LK	5	259046.5	17.1
	ACS-3*opt*	10	259045	8.5
	ACS-LK	10	259045	9.4
	LK	10	259045	2.8
pr2392 (378032)	GA-LK	8	378034.8	1114.1
	ACS-3*opt*	10	378032	212.9
	ACS-LK	10	378032	239.5
	LK	10	378032	65.8

Tableau 4.1 : Comparaison de ACS-LK, ACS-3*opt*, GA-LK et LK

La première colonne du tableau 4.1 contient le nom du problème dans TSPLIB avec entre parenthèses le coût optimal, la deuxième l'algorithme, la troisième le nombre de fois que l'algorithme aboutit à l'optimum sur 10 exécutions effectuées, la quatrième le coût moyen obtenu et la cinquième le temps moyen en utilisant un micro ordinateur de processeur Intel Pentium 4, 1.7 MHz avec 256MO de RAM.

Le temps pris par la recherche locale dans la méthode hybride ACS-LK représente 69,23%, 93,84%, 63,63%, 92,68% et 96,25% du temps d'exécution total pour lin318, att532, rat783, pr1002 et pr2392 respectivement.

Nous constatons que notre approche ACS-LK aboutit toujours à l'optimum (Succès =10/10) par contre pour le problème att532, ACS-3*opt* n'est arrivé à l'optimum que 2 fois sur 10 et pour les problèmes lin318, pr1002 et pr2392, GA-LK n'aboutit à l'optimum que 4 fois sur 10, 5 fois sur 10 et 8 fois sur 10 respectivement. Ceci nous permet de conclure que notre hybride produit une qualité de solution meilleure que les deux autres hybrides. Pour les problèmes considérés, le temps d'exécution de ACS-LK est meilleur par rapport à celui de GA-LK. ACS-LK aboutit toujours à l'optimum ; ce qui n'est pas le cas pour LK (9/10) pour les problèmes lin318 et att532.

La figure 4.1 illustre la qualité de la solution en fonction du temps des 3 algorithmes hybrides pour le problème pr2392. On constate qu'au démarrage de l'algorithme, GA-LK présente des solutions de meilleure qualité par rapport à ACS-LK et ACS-3*opt* mais prend trop de temps par la suite pour les améliorer. Sur le tableau 4.1 nous avons l'impression que ACS-3opt et ACS-LK se valent globalement mais après une comparaison en temps constant, nous constatons (exemple de la figure 4.1) qu'en général la qualité des solutions de ACS-LK est meilleure par rapport á celle des solutions de ACS-3*opt*. Par conséquent nous pouvons conclure qu'en général, ACS-LK est meilleur que les deux autres hybrides ACS-3*opt* et GA-LK pour les problèmes de taille inférieure à 2392.

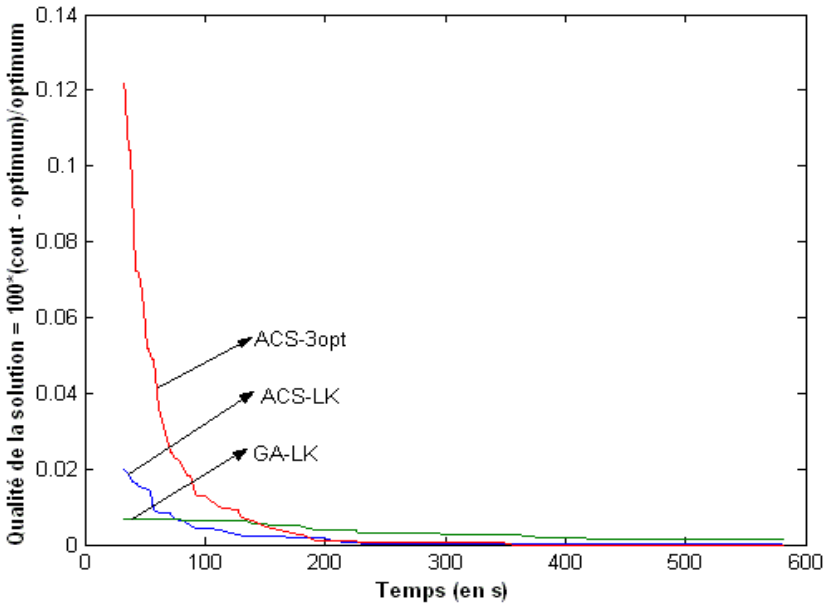

Figure 4.1 : Comparaison de ACS-LK, ACS-3*opt* et GA-LK sur le problème pr2392

La grande différence de qualité au début de l'algorithme qui existe entre ACS-3*opt* et ACS-LK est due à l'efficacité de LK par rapport à 3-*opt*. LK a la possibilité d'effectuer des échanges de 2, 3, 4 voir 5 liens alors que 3-*opt* se limite à 3 liens. Par ailleurs la convergence lente de GA-LK s'explique par le fait que dès la première itération, l'application de LK aux individus produit une population d'individus presque identiques et il devient difficile par la suite d'avoir une population assez divergente.

5. Conclusion

Nous avons présenté plusieurs heuristiques pour résoudre le problème du voyageur de commerce (TSP). Nous avons implémenté celles que nous avons jugées les plus efficaces (SA, GA, ACS, LK, ACS-LK, ACS-3*opt*, GA-LK) pour une étude comparative sur 10 instances de TSP symétriques. Le tableau 4.1 illustre 5 de ces problèmes. Les résultats expérimentaux nous ont permis de confirmer la supériorité de performance de LK sur les autres heuristiques d'optimisation locale.

ACS présente une meilleure performance par rapport à SA et GA, mais reste de loin inférieure à LK en termes de qualité de la solution et du temps d'exécution. Ce constat nous a motivé à hybrider ACS et LK (ACS-LK). Les résultats exépérimentaux sur des problèmes de la bibliothèque TSPLIB de taille comprise entre 318 et 2392 montrent que notre méthode hybride donne des résultats plus proches de l'optimum que ceux de Freisleben et Merz résultant de l'hybridation de GA avec LK et ceux de Dorigo et Gambardella résultant de l'hybridation de ACS avec 3-*opt*.

En perspective il est logique de penser qu'en parallélisant ces heuristiques on améliorerait leurs temps d'exécution. Pour les colonies de fourmis par exemple, on pourrait attribuer la tâche de chaque fourmi à un processeur.

6. Bibliographie

[1] M. DORIGO, L. M. GAMBARDELLA, « Ant Colony System : A cooperative learning approach to the traveling salesman problem », *IEEE Transactions on Evolutionary Computation*, n° 1(1) :53-66, 1997.

[2] B. FREISLEBEN, P. MERZ, « New genetic local search operators for the traveling salesman problem », *Proceedings of PPSN IV-Fourth International Conference on Parallel Problem Solving From Nature, H.-M. Voigt, W. Ebeling, I. Rechenberg and H.-S. Schwefel (Eds.), Springer-Verlag, Berlin*, 160-168, 1996.

[3] J. GREFENSTETTE, R. GOPAL, B. ROSIMAITA, D. V. GUCHT, « Genetic Algorithms for the Traveling Salesman Problem », *Proceedings of an International Conference on Genetic Algorithms and their Applications, Carnegie Mellon publishers*, 160-168, 1985.

[4] F.GLOVER, « Tabu Search-Part II », *ORSA Journal on Computing*, n° 2(1) :4-32, 1990.

[5] KELD HELSGAUN, « An effective implementation of the Lin-Kernighan traveling salesman heuristic », *European Journal of Operations Research*, n° 12 :106-130, 2000.

[6] DAVID S. JOHNSON, LYLE A. MCGEOCH, E.E. ROTHBERG, « Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound », *in Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, Atlanta, Georgia, January 28-30*, 341-350, 1996.

[7] DAVID S. JOHNSON, LYLE A. MCGEOCH, « The Travelling Salesman Problem : A Case Study in Local Optimization », *in E.H.L. Aarts and J.K. Lenstra (eds.), "Local Search in Combinatorial Optimization", Wiley, New York*, 1997.

[8] S. KIRKPATRICK, C. D. GELATT, M.P. VECCHI, « Optimization by Simulated Annealing », *Science*, n° 671-680, 1983.

[9] http ://www.iwr.uni-heidelberg.de/iwr/comopt/ software/TSPLIB95
Février 2005