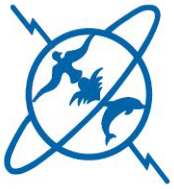
Université Mohammed V de Rabat

Laboratoire de Recherche en Informatique et Télécommunications



Faculté des Sciences

Thèse de Doctorat

présentée par

Mehdi El Krari

Discipline: Sciences de l'ingénieur

Spécialité: Informatique

Adaptation de Métaheuristiques pour résoudre des Problèmes d'Optimisation Combinatoire liés aux Transports

---

Soutenue le 02/03/2019, devant le jury composé de:

Président:

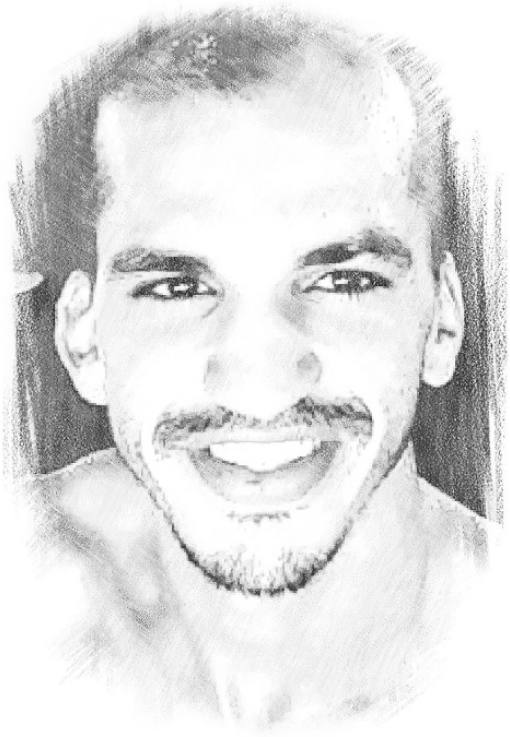
Ahmed HAMMOUCH	Université Mohammed V de Rabat

Rapporteurs:

Belaïd AHIOD	Université Mohammed V de Rabat
Salma MOULINE	Université Mohammed V de Rabat
Sébastien VEREL	Université du Littoral Côte d'Opale, France

Examinateurs:

Mohamed EL HASSOUNI	Université Mohammed V de Rabat
John Robert WOODWARD	Queen Mary University of London, UK



À jamais dans nos cœurs

Avant propos

Cette thèse a été préparée au sein du Laboratoire de Recherche en Informatique et Télécommunications (LRIT) de l'université Mohammed V de Rabat, sous la direction du Professeur **Belaïd AHIOD**.

Je remercie donc, tout d'abord, Monsieur **Belaïd AHIOD** d'avoir accepté de m'encadrer durant les trois cycles universitaires (LMD); surtout ce dernier où j'ai appris à ses côtés les fondements de l'optimisation combinatoire et les principes du calcul évolutionnaire. Le Professeur AHIOD a été fondamental sur le plan théorique, pratique, mais aussi éthique.

Un grand merci à Monsieur **Ahmed HAMMOUCH**, chercheur au LRIT, Professeur de l'Enseignement Supérieur à L'École Normale Supérieure de L'Enseignement Technique de Rabat (ENSET) et Directeur de la Recherche Scientifique et de l'Innovation au Département de l'Enseignement Supérieur et de la Recherche Scientifique du Ministère de l'Éducation Nationale, de la Formation Professionnelle, de l'Enseignement Supérieur et de la Recherche Scientifique, d'avoir accepté de présider ma soutenance.

Mes remerciements vont aussi à Madame **Salma MOULINE**, membre permanente du LRIT et Professeur de l'Enseignement Supérieur (PES) à la Faculté des Sciences de Rabat, pour avoir rapporté ma thèse ainsi que pour ses remarques pertinentes qui ont aidé à améliorer ce mémoire.

Je voudrais aussi exprimer ma plus grande gratitude à Monsieur **Sébastien VEREL**, Professeur au département informatique de l'université du littoral Côte d'Opale (ULCO) et chercheur au Laboratoire d'Informatique Signal et Image de la Côte d'Opale (LISIC), d'avoir accepté de rapporter ma thèse et d'être parmi les membres du jury.

Je tiens à remercier infiniment Monsieur **Mohamed EL HASSOUNI**, Professeur Habilité à la Faculté des Lettres et des Sciences Humaines de Rabat et membre permanent du LRIT, d'avoir accepté de faire parti des membres du jury et examiner mes travaux de recherche.

Je suis également reconnaissant envers Monsieur **John Robert WOODWARD**, chercheur et chef de l'équipe "*Recherche Opérationnelle*" du département Génie Électrique et Informatique de l'université Queen Mary University of London, pour avoir fait le déplacement pour assister à ma soutenance et examiner ce travail.

Présente à mes côtés dans chaque moment de ma vie, ce manuscrit, même rempli de *merci*, n'aurait pas suffit. Tout le monde prône avoir la meilleure maman au monde, seules trois personnes peuvent se vanter de l'avoir: mon frère (Faïçal) et ma sœur (Basma). Votre soutien était fondamental. Je vous aime tous les trois.

On ne choisit pas sa famille certes, mais j'ai été choyé par le bon dieu d'avoir des proches extraordinaires. Merci à tous ceux qui ont été là durant ces cinq ans pour m'encourager, ne serait-ce que par un petit mot. Sans citer personne pour ne pas vexer ceux que je pourrait oublier, mais sans oublier Omar: « Tu resteras éternellement inoubliable ».

Amine, Asmae, Brahim, Hicham, Ismail, Issam, Maya, Mohamed, Nabil, Nadir, Saad, Safae, Yaf, Yassine... Vous étiez ma deuxième famille. Merci pour votre support.

Merci aux différents établissements et organisations d'avoir soutenu mes travaux de recherche:

- Bourse d'excellence du Centre National pour la Recherche Scientifique et Technique (CNRST)
- Bourse de mobilité de l'Agence Universitaire de la Francophonie pour assister à la conférence ISDA 2016 à Porto, Portugal
- Subvention de voyage (Travel grant) de la part de ACM pour assister à GECCO 2017 à Berlin, Allemagne

Résumé

En plein essor, l'industrie des transports engendre des revenus attrayants et une rude concurrence. Certains problèmes d'optimisation combinatoire ont été modélisés afin de répondre aux besoins de cette industrie et faire face à ses différentes contraintes. Des métaheuristiques de différentes classes ont été proposées pour résoudre ces problèmes. Cette thèse s'intéresse aux métaheuristiques basées sur la recherche locale pour résoudre le problème du voyageur de commerce (TSP) et celui généralisé (GTSP). Le premier travail a été l'adaptation d'une métaheuristique basée sur la recherche locale itérée pour résoudre le TSP, puis la proposition d'une version mémétique basée sur cette métaheuristique pour résoudre le GTSP, tout en apportant de nouvelles contributions. Ayant fait face au problème de dimensionnalité, une méthode de réduction pour le GTSP a été proposée pour diminuer la taille des instances du problème et rendre les solveurs plus rapides. D'autres travaux ont été effectués dans le cadre de cette thèse. Le premier est une étude empirique sur la méthode de construction Multi-Fragment. Le deuxième est une observation sur les barycentres des clusters du GTSP qui peuvent servir de différentes façons.

Mots-clés: Recherche locale, Optimisation combinatoire, Métaheuristiques, Transports, Réduction de la dimensionnalité.

Abstract

The transportation industry is growing and generates attractive revenues and stiff competition. Some combinatorial optimization problems have been modelled to meet the needs of this industry and face its different constraints. Metaheuristics of different classes have been proposed to solve these kinds of problems. This thesis focused on metaheuristics based on local search to solve the travelling salesman problem (TSP) and the generalized one (GTSP). The first work was the adaptation of a metaheuristic based on the iterated local search framework to solve the TSP, then a memetic version based on this approach to solve the GTSP, while making new contributions. Having faced the problem of dimensionality, a reduction method for the GTSP has been proposed to reduce the size of the instances of the problem and make the solvers faster. Other works have been done as part of this thesis. The first is an empirical study of the Multi-Fragment construction heuristic. The second is an observation on the cluster barycenters of the GTSP which that can be used in different ways.

Keywords: Local search, Combinatorial optimisation, Metaheuristics, Transportation, Dimensionality reduction.

TABLE DES MATIÈRES

Liste des tableaux

Liste des figures

Liste des abréviations:

AG	Algorithme Génétique
AM	Algorithme Mémétique
BKS	Best Known Solution (Meilleure Solution Connue)
BLS	Breakout Local Search
CBLS	Cluster-Based Local Search (Recherche Local Basée sur les Clusters)
CO	Cluster Optimization Algorithm
GLKH	Generalized Lin Kernighan Helsgaun
GTSP	Generalized Travelling Salesman Problem (Problème du Voyageur de Commerce Généralisé)
ILS	Iterated Local Search (Recherche Locale Itérée)
MBLS	Memetic Breakout Local Search
MF	Multi-Fragment
NENLS	Neighbourhood Node Exchange Local Search
NN2C	Nearst Nodes to Clusters
POC	Problème d'Optimisation Combinatoire
TSP	Travelling Salesman Problem (Problème du Voyageur de Commerce)
VND	Variable Neighbourhood Descent (Descente par Voisinage Variable)
VNS	Variable Neighbourhood Search (Recherche par Voisinage Variable)

Introduction générale

L'industrie du transport s'est fortement développée depuis la révolution industrielle. Dans une économie où les besoins en consommation sont de plus en plus grandissants, son importance est devenue indiscutable aussi bien dans le secteur public que dans le privé. Les revenus qu'elle peut dégager ainsi que les frais qu'elle occasionne imposent la mise en place d'une politique et d'un système tout sauf arbitraire.

Pour en tirer les meilleurs bénéfices, une stratégie visant à maximiser les revenus et minimiser les frais doit être développée et optimisée. Des modèles mathématiques inspirés de problèmes réels sont conçus pour permettre à la communauté scientifique dans le domaine de l'optimisation de mettre en œuvre des méthodes de différentes catégories.

Le problème du voyageur de commerce (TSP) et ses variantes sont des problèmes d'optimisation combinatoire (POC) $\mathcal{NP}-$complets qui contribuent considérablement à la résolution de divers problèmes réels, y compris ceux des domaines du transport. D'autre part, le TSP fait partie des premiers problèmes auxquels s'attaquent la plupart des nouvelles métaheuristiques afin de valider la méthode proposée, les instances du TSP étant de différentes topologies et difficultés.

Différentes méthodes d'optimisation permettent de résoudre les POC. Les approches exactes apportent la solution optimale dans un temps exponentiel, ce qui les rend pratiquement stériles face aux instances larges. Les métaheuristiques sont par contre beaucoup plus rapides, mais ne garantissent pas d'atteindre l'optimalité. Plusieurs travaux ont cependant émergé dans la littérature fournissant des solutions de plus en plus satisfaisantes.

Cette classe d'algorithmes permet d'avoir des solutions dans un temps assez court, ce qui lui permet d'être praticable pour les instances larges contrairement aux méthodes exactes. Mais d'un autre côté, elle ne donne aucune garantie sur la qualité de la solution (par rapport à l'optimale). Cette contrainte a suscité l'intérêt des chercheurs afin de proposer des heuristiques pouvant fournir de

bonnes solutions (voire même les optimales) avec une probabilité très élevée indépendamment de la topologie de l'instance.

Parmi les différentes classes de métaheuristiques proposées dans la littérature, la recherche locale fait partie des plus sollicitées. Sa capacité à exploiter un ensemble de solutions pour en tirer la meilleure a fait en sorte de constituer la base ou d'être une composante majeure de plusieurs travaux de recherche couronnés de succès dans divers problèmes.

La recherche locale itérée (ILS), une métaheuristique basée sur le principe de la recherche locale, se base sur la simple idée de fuir un optimum local en lui appliquant une modification pour obtenir une solution qui n'appartient pas au dernier bassin d'attraction visité. L'algorithme, simple mais dont la contribution est notable, a vu de nouvelles métaheuristiques qui en découlent proposant des améliorations à une de ses composantes ou en le combinant dans une approche hybride.

Cette thèse aborde en première partie *Breakout Local Search* (BLS), une de ces récentes métaheuristiques basées sur ILS, afin de résoudre le TSP et une de ses variantes. L'idée apportée par BLS et les bons résultats obtenus pour de nombreux POC ont été les motivations principales pour s'intéresser à cette métaheuristique.

Le deuxième chapitre présente la première contribution de cette thèse, une adaptation de BLS pour le problème du voyageur de commerce (TSP), où des perturbations variables en types de mouvements ont été proposées afin de mieux échapper aux optima locaux de forte attraction. Les résultats comparatifs ont montré une nette amélioration par rapport à ILS, mais ont en même temps décelé une lenteur contraignante sur les instances difficiles, où BLS peine parfois à échapper les bassins d'attractions.

Pour augmenter les chances de tomber sur des optimums locaux meilleurs, une combinaison entre BLS et un algorithme génétique (AG) appliquée au TSP généralisé (GTSP) a été proposée dans le troisième chapitre comme deuxième travail de cette première partie de thèse. Connus pour leurs aspects explorateurs, les

AG permettront à BLS de se positionner dans plusieurs points de l'espace de recherche en même temps. Ce travail était aussi l'occasion de proposer une nouvelle amélioration afin de réduire considérablement le temps nécessaire à BLS pour améliorer un individu.

La deuxième partie et quatrième chapitre de cette thèse propose une méthode de réduction pour le GTSP. Le nombre de villes d'une instance de ce problème peut être réduit étant donné qu'une solution faisable ne les visite pas toutes. La méthode présentée dans ce chapitre offre des taux de réduction assez conséquents et supérieurs par rapport à l'unique approche qui existe dans la littérature pour le GTSP. Les instances réduites fournissent des solutions beaucoup plus rapidement et ne perdent pas trop en qualité par rapport aux solutions retournées par les instances complètes.

D'autres travaux ont été accomplis dans le cadre de cette thèse. Le premier est une étude empirique de la méthode Multi-Fragment, une heuristique de construction pour le TSP. Le but était de fournir plus de détails puisqu'il en manquait dans la littérature. Le deuxième travail est une introduction au principe du barycentre sur les clusters des instances du GTSP. Les barycentres ont été présentés comme étant un moyen de représenter (virtuellement) leurs clusters respectifs, ce qui peut servir par exemple pour situer un cluster par rapport aux autres ou pour construire des solutions de départ.

La présente thèse sera mise en contexte dans le premier chapitre avant de discuter les contributions de citées ci-dessus. Les POC abordés y seront définis avant de dresser un état de l'art des différentes méthodes d'optimisation ayant contribué le plus dans le domaine.

Chapitre 1: Les métaheuristiques pour les problèmes d'optimisation combinatoire

1.1 Introduction

L'optimisation des réseaux de transport est un challenge de taille face auquel l'industrie tente de trouver les meilleures solutions possibles. Face à des réseaux de taille considérable, les possibilités et les combinaisons se multiplient et rendent la recherche de bonnes solutions de plus en plus difficile.

L'optimisation combinatoire regorge de modèles mathématiques pouvant s'adapter aux différents problèmes réels et à travers lesquels l'obtention de solutions satisfaisantes pour les décideurs devient possible. Diverses méthodes d'optimisation ont vu le jour et permettent aujourd'hui de fournir des solutions acceptables certes, mais toujours améliorables, que ce soit en terme de qualité de la solution fournie ou du temps nécessaire pour l'avoir, ou les deux.

Le premier chapitre de cette thèse fera l'objet d'un état de l'art des méthodes d'optimisation les plus populaires dans l'optimisation combinatoire. Il va de soi alors de définir d'abord quelques-uns de ces problèmes auxquels s'attaquent ces méthodes, notamment les problèmes abordés dans cette thèse.

1.2 Les Problèmes d'optimisation combinatoire et les transports

1.2.1 Les transports: un centre d'intérêt dans l'optimisation combinatoire

Les transports sont définis comme étant le transfert de personnes ou d'objets en moyennant un véhicule ou tout autre engin, entre plusieurs endroits géographiquement séparés. Les transports ont toujours été un facteur important dans la société dont il fallait satisfaire les besoins et les attentes, d'où l'importance d'analyser les modèles mathématiques des réseaux de transport.

Plusieurs recherches avancées sur les modèles de transports ont été effectuées. On peut trouver les bases des modèles utilisés dans les réseaux de transport modernes dans des travaux comme ceux de Overgaard [1] ou de Steenbrink [2], des progrès récents basés sur ces modèles ont été réalisés par Hensher et Button [3] et Hoogendoom et Bovy [4].

Le déplacement d'un point A vers un un autre B est toujours accompagné par un bénéfice ou objectif (nécessité de passer par B, récolter des objets,...) et un ou plusieurs coûts (temps de déplacement, distance parcourue,...). Chaque personne ou entité utilisant un réseau de transport pour son propre besoin a l'objectif de maximiser la différence entre les bénéfices et les coûts comme l'est indiqué dans la formule 1.1.

$$\max_{m,p,B} \ (u^{AB} - t^{mpAB}) \tag{1 1}$$

- u^{AB} sont les profits ou objectifs atteints en se déplaçant de A vers B.
- t^{mpAB} sont les coûts appliqués lors du déplacement d'un engin m de A vers B en empruntant le chemin p.

Or un réseau de transport ne s'agit pas que de deux points, mais plusieurs. Optimiser la formule 1.1 pour un réseau devient plus difficile pour un réseau plus grand, le nombre de solutions réalisables s'amplifie considérablement compliquant la découverte de la meilleure solution. Trouver la meilleure solution dans ensemble fini (mais souvent trop large) de possibilités entre dans le cadre de l'optimisation combinatoire, appelé aussi mathématiques discrètes ou théorie des graphes.

Afin de prendre les décisions nécessaires pour répondre aux besoins d'une entreprise, d'une mairie ou d'un gouvernement, des techniques d'optimisation, dont certaines seront abordées dans ce chapitre, ont été proposées et explorées au fil des ans. L'élargissement des réseaux de transport et le besoin continu d'améliorer les solutions existantes poussent les ingénieurs et les chercheurs du domaine à concevoir des méthodes plus performantes que celles qui les ont précédées. Certaines

sont génériques et peuvent être appliquées à n'importe quel modèle, d'autres sont par contre spécifique au problème étudié.

Plusieurs problèmes, dits Problèmes d'Optimisation Combinatoire (POC), ont été formulés pour répondre aux besoins affichés dans le domaine des transports. Un POC est un problème où il faut (souvent) minimiser une certaine fonction (dite de coût ou de fitness) sur un ensemble fini de configurations. Beaucoup de ces problèmes peuvent être représentés sous forme de graphe. Un graphe est composé d'un ensemble fini de points et d'un certain nombre de liens (orientés ou non) reliant ces points. À chacun de ces liens peut être associé un poids. Par exemple, on peut considérer le graphe suivant: les points sont les aéroports du monde entier, les liens sont les liaisons qui existent entre ces aéroports et les poids peuvent être le temps de vol, la distance ou le prix du voyage.

Le problème de tournées de véhicules (*Vehicle Routing Problem* (VRP)) [5], le problème de tournées sur les arcs avec contraintes de capacité (*Capacited Arc Routing Problem* (CARP))[6], le problème d'affectation quadratique (Quadratic Assignment Problem (QAP))[7], ou encore le problème du voyageur de commerce (*Travelling Salesman Problem* (TSP))[8] ainsi que leurs variantes font parti des POC les plus sollicités et les plus cités dans la littérature. Ce dernier (le TSP) en plus d'une de ses extensions seront abordés dans cette thèse. Une définition de ces problèmes ainsi que les principales motivations (applications, difficultés,...) derrière ce choix seront fournis dans les prochaines sections.

1.2.2Le Problème du Voyageur de Commerce (TSP)

Le problème du voyageur de commerce (TSP) est l'un des problèmes les plus étudiés en optimisation combinatoire [8], [9]. Il a été introduit pour la première fois par William Rowan Hamilton (1859). Formellement, le problème peut être défini comme suit: Étant donné un nombre fini de villes et un coût de voyage (distance, temps, énergie,...) entre chaque paire, trouvez le tour le moins coûteux pour visiter toutes les villes et de revenir au point de départ.

Plus en détails, à partir de n villes et une matrice $D = (d_{ij})_{n \times n}$ de distances entre chaque couple de villes, le TSP vise à trouver le plus court tour

fermé (i.e. cycle Hamiltonien) qui visite chaque ville une seule fois. Chaque tour peut être représenté par une permutation $\pi = (\pi(1), \pi(2), ..., \pi(n))$ d'entiers allant de 1 à n et où $j = \pi(i)$ indique que la ville j est la i$^{\text{ème}}$ ville à visiter dans le tour π. Le but est donc de trouver une permutation π qui minimise la longueur du tour calculé par la formule 1.2 suivante

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \qquad (1\ 2)$$

où d_{ij} est la distance entre les villes i et j, f est appelé fonction de fitness.

Outre le domaine des transports, le TSP est un problème applicable aussi pour des modèles d'industrie diverses. En électronique, l'intégration à très grande échelle (Very-Large-Scale Integration (VLSI)) est très proche du modèle TSP [8]. VLSI est une technologie utilisée dans les circuits intégrés permettant d'inclure des milliers de composants électronique sur une seule puce, le challenge est donc de percer plusieurs points sur une carte électronique le plus rapidement possible puis interconnecter ces composants pour former un circuit. La cristallographie aux rayons X [10] est aussi un champ d'application très cité dans la littérature.

Le TSP est très sollicité par les chercheurs dans le domaine de l'optimisation. Beaucoup de nouvelles idées ont été présentées dans la littérature en s'attaquant à ce problème, il est considéré par la communauté comme un "*banc d'essai*" permettant de valider la méthode proposée. Cette particularité revient à la diversité de ses instances qui sont de difficultés et de topologies variées. L'obtention de bonnes solutions pour un large ensemble d'instances est souvent un signe d'efficience de l'algorithme.

Le TSP dispose de plusieurs variantes et extensions. La variante la plus populaire est celle définit ci-dessus, où les distances entre les villes sont symétriques $d_{\pi(i),\pi(j)} = d_{\pi(j),\pi(i)}$, puisqu'il existe une autre variante asymétrique (ATSP) où le nombre de solution faisables est le double par rapport au TSP pour deux instances de même taille. Les extensions sont nombreuses, le TSP généralisé (Generalized TSP (GTSP)) [11], [12], le TSP avec fenêtres de temps (TSP with Time Windows

(TSPTW)) [13] et le TSP dynamique (Dynamic TSP (DTSP)) [14] sont parmi tant d'autres [15]. La première citée sera présentée en détails dans la prochaine section, tandis que les autres seront introduites brièvement dans la section qui la suit.

1.2.3Le Problème du Voyageur de Commerce Généralisé (GTSP)

Le Problème du Voyageur de Commerce Généralisé (GTSP) est, comme son nom l'indique, une généralisation du TSP proposée par Srivastava et al. [11] et Henry-Labordere [12] en 1969. Un ensemble V de n nœuds (ou villes) est réparti sur m clusters (C_1, C_2, ..., C_m), un seul nœud de chaque cluster C_i doit être visité et former un cycle Hamiltonien en revenant au cluster/nœud de départ.

Plus en détails, le GTSP est défini à l'aide d'un graphe $G=(V,E,W)$ pondéré et complet. $V=(v_1, v_2, ..., v_n)$ est l'ensemble des nœuds/villes, E représente les arêtes/chemins (v_i, v_j) (pour $i,j = 1, 2, ..., n$) et W est la fonction de fitness qui détermine la distance pour chaque arête. Le GTSP à n clusters est plus difficile à résoudre que le TSP à n nœuds [16]. En effet, une instance TSP peut être considérée comme une instance GTSP avec un seul nœud dans chaque cluster.

Une solution faisable pour le GTSP est un sous-graphe $S=(P,\pi,W)$ de G où $P = p_1,p_2,... p_m$ est l'ensemble des villes tel que chaque cluster C_k a exactement un seul nœud p_t pour $k,t = 1, 2, ..., m$ ($\mathrm{p}_i \neq p_j, p_i \in C_x, p_j \in C_y, \Rightarrow C_x \neq C_y$). π est un sous-ensemble de connections appartenant à E et formées par P dont la taille est de m complétant un cycle Hamiltonien dont la fonction de fitness est donné par la formule 1.3 suivante:

$$W(\pi) = c(p_m, p_1) + \sum_{i=1}^{m-1} c(p_i, p_{i+1}) \qquad (1\ 3)$$

De nombreuses applications réelles peuvent être modélisées en tant que GTSP. Le routage des avions [17], la distribution du courrier [18], le routage des véhicules [18], la sélection de boîtes postales [18], la planification des client par les organismes de protection sociale [19] et le séquencement de fichiers informatiques

[12] sont quelques exemples de nombreux problèmes réels d'optimisation combinatoire pouvant être modélisé comme un GTSP.

La transformation du GTSP en TSP a fait l'objet de plusieurs travaux [20]–[23], certains d'entre eux fournissent une solution optimale pour le TSP transformé qui peut être convertie en une solution optimale pour le GTSP [23]. Un problème majeur avec cette transformation est que la solution obtenue pour le TSP peut ne pas être réalisable pour le GTSP si elle n'est pas optimale.

Certaines références dans la littérature définissent le GTSP comme étant un problème dont la solution est un tour qui doit passer par exactement une ville de chaque cluster, comme il a été définit dans cette section, d'autres par contre caractérisent la solution à ce problème par un tour passant par au moins une ville de chaque ensemble C_k. Les précurseurs de cette seconde définition mentionnent et dénomment la première par le *Equality*-GTSP (E-GTSP).

1.2.3.1 Un problème à deux niveaux

Le GTSP est considéré comme un problème d'optimisation à deux niveaux (*bi-level optimisation problem*) [24], [25]. Une solution $s = (\pi, \tau)$ faisable est en effet représentée en deux parties: (i) l'ensemble des nœuds couvrants $\pi = \pi_1, \pi_2, ...\pi_m$ où $\pi_i \in C_i$, et (ii) une permutation des clusters $\tau = \{\tau_1, ..., \tau_m\}$ qui forme un cycle Hamiltonien de taille m.

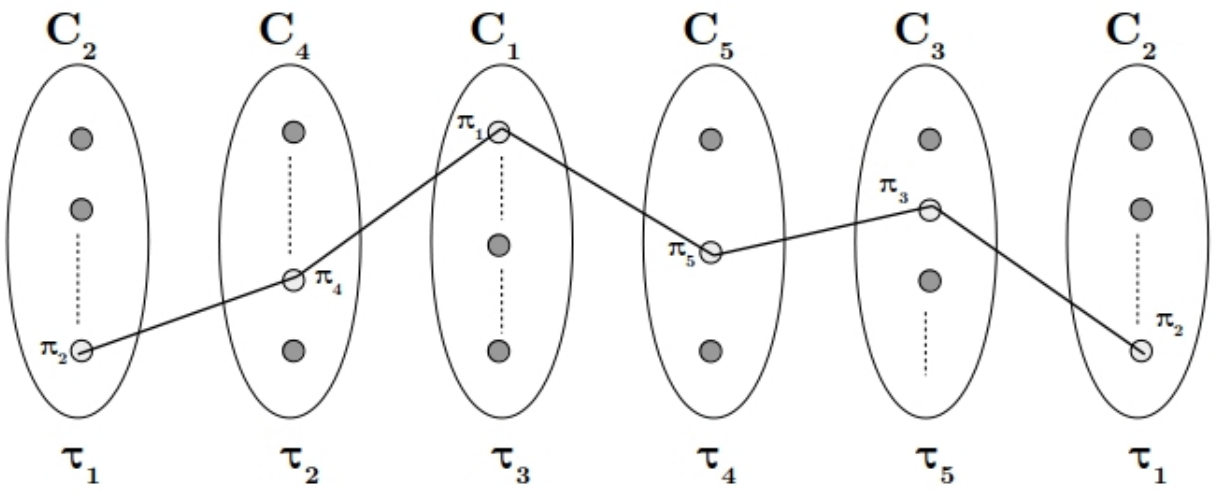


Figure 1.1: Exemple de représentation d'une solution d'une instance GTSP

Hu et Raidl [24] ont proposé deux approches basées sur cette représentation pour résoudre le GTSP. La première est basée sur les clusters: la couche supérieure de l'algorithme cherche la meilleure permutation des clusters, tandis que la couche inférieure retourne l'ensemble optimale des nœuds couvrants. La deuxième approche est basée sur les nœuds, elle procède dans le sens inverse de la première: la couche supérieure construit l'ensemble des nœuds couvrant puis en déduit le meilleur tour dans la couche inférieure. Les deux approches ainsi qu'une troisième hybride (VNS) seront discutées dans le troisième chapitre de la thèse.

1.2.3.2 Les barycentres d'une instance du GTSP

Pour trouver une bonne permutation de clusters dans la première approche, l'utilisation des barycentres a été proposée [26] afin de pouvoir représenter chaque cluster par un point fictif et unique, tout en tenant compte de la position de leurs villes respectives. Le barycentre B d'un cluster C est calculé selon l'équation 1.4 ci-dessous, où $k = |C|$.

$$B(x,y) = \begin{cases} x = \sum_{i=1}^{k} (x_i)/k \\ y = \sum_{i=1}^{k} (y_i)/k \end{cases} \qquad (1\ 4)$$

1.2.4 D'autres extensions du TSP

Les extensions du TSP sont nombreuses et diverses, elles se distinguent entre elles par les objectifs à atteindre et/ou par les contraintes à prendre en compte, à l'image des extensions citées ci-dessous.

- *Le TSP groupé* (the Clustered TSP (CTSP)) [27]: Comme dans le GTSP, les villes sont réparties en clusters, sauf que le voyageur doit cette fois-ci visiter toutes les villes de chaque cluster visité avant de le quitter. Le problème a de nombreuses applications, notamment la défragmentation de disque, le routage de véhicules et la planification de temps [28].
- *Le TSP en groupes ordonnés* (the Ordered Cluster TSP (OCTSP)) [29]: Une version simplifié du CTSP où l'ordre de visite des clusters est prédéfini. Le premier cluster ne contient qu'un seul nœud qui représente le point de départ

de la tournée. Des applications bien connues dans le routage automatisé d'entrepôts [27], [30] et la planification de la production [30] peuvent être résolues par le modèle du OCTSP.

- *Le TSP avec fenêtres de temps* (TSP with Time Windows (TSPTW)) [13], [31], [32]: Pour chaque arête (i,j), c_{ij} est le coût (distance) et t_{ij} est le temps de voyage entre i et j. Pour chaque nœud/ville i il existe un triplet $\{a_i, b_i, s_i\}$ tel que $a_i \leq b_i$ et $0 \leq s_i \leq b_i - a_i$. s_i est une quantité qui définit le temps de service dans la ville i, où une tâche doit être complétée dans intervalle $[a_i,b_i]$. Le voyageur doit attendre l'instant a_i s'il arrive avant au nœud i pour effectuer sa tâche. L'objectif du TSPTW est de trouver un tour de coût minimal respectant les contraintes citées ci-dessus.

- *Le TSP dynamique* (the Dynamic TSP (DTSP)) [14]: C'est un modèle de TSP défini par une matrice de distance dynamique $D(t) = \{d_{ij}(t)\}_{n(t)\times n(t)}$ où $d_{ij}(t)$ est la distance entre les villes i et j au moment t. Le nombre de ville $n(t)$ dépend lui aussi de t.

- *Le TSP noir et blanc* (the Black and White TSP (BWTSP)) [33]: L'ensemble des nœuds sont répartis en deux catégories, les blancs et les noirs. Un tour est dit faisable s'il passe par tous les nœuds blancs et noirs et si (i) le nombre de nœuds blancs entre deux noirs consécutifs ne dépasse pas un entier positif I, et (ii) la distance entre deux noirs consécutifs ne dépasse pas un nombre réel positif R. Le but du BWTSP est un tour faisable avec un coût minimal.

- *Le problème du voyageur voleur* (the Travelling Thief Problem (TTP)) [34], [35]: Un problème récent multi-composantes qui combine entre le TSP et le problème du sac à dos [36]. Un voleur visite les villes exactement une fois, ramasse des objets dans les villes et remplit son sac. Le but est de trouver un tour qui visite toutes les villes une seule fois et qui retourne à la ville de départ, en optimisant la ou les fonctions objectives sans que le poids total du sac à dos ne soit enfreint. La ou les fonctions objectives peuvent être liées au moment du voyage et/ou à la valeur totale que le voleur gagne à partir des

objets choisis. Une étude du paysage du TTP a été effectuée en marge de cette thèse dans le cadre d'une collaboration internationale [37].

1.3 Espace de recherche et complexités

Les POC sont caractérisés par la croissance rapide du nombre de solutions faisables en fonction de la taille du problème (e.g. le nombre de villes dans le TSP). Ces solutions, regroupées dans un espace de recherche, deviennent difficiles à explorer afin de pouvoir tirer la solution optimale.

Il convient d'introduire les notions d'espace de recherche et de complexités avant de discuter les différentes méthodes de résolution des POC (à partir de la prochaine section). Dans le même contexte, des méthodes de prétraitement permettant de réduire la taille de l'espace de recherche seront abordées à la fin de cette section.

1.3.1 L'espace de recherche d'un POC

L'espace de recherche d'un POC, ou plus exactement d'une instance I de ce problème, regroupe toutes ses solutions faisables. La croissance de sa taille est souvent exponentielle ou factorielle par rapport à la taille (N) de I. Il devient alors difficile de trouver la solution optimale à partir d'un N très petit en faisant une recherche exhaustive dans l'espace. Avec 15 villes dans le TSP, la taille de l'espace de recherche dépasse 43 milliards, et passe à 654 milliards avec une ville de plus (16).

1.3.2 Complexités temporelle et spatiale

Les complexités sont, contrairement à l'espace de recherche, des notions liées à l'algorithme plutôt qu'au problème. Il existe néanmoins un lien entre les deux: la taille du problème et le codage adopté influencent respectivement la complexité temporelle et spatiale. Le tableau 1.1 [38] ci-dessous apporte une approximation sur la complexité maximale à prévoir pour quelconque méthode par rapport à la taille (N) du problème.

Tableau 1.1: Indication sur la complexité maximale en fonction de la taille du problème

Taille du problème	**Complexité maximale permise**
$\leq [10...11]$	$O(N!), O(N^6)$
$\leq [15...18]$	$O(2^N \times N^2)$
$\leq [18...22]$	$O(2^N \times N)$
≤ 100	$O(N^4)$
≤ 400	$O(N^3)$
$\leq 2k$	$O(N^2 \times logN)$
$\leq 10k$	$O(N^2)$
$\leq 1M$	$O(N \times logN)$
$\leq 100M$	$O(N), O(logN), O(1)$

Il est donc important de mettre en œuvre des méthodes de complexité réduite pour s'attaquer aux POC de taille importante pour répondre au besoin des applications réelles et fournir des solutions dans un laps de temps acceptable.

1.3.3Optimisation dans les espaces larges

Cette difficulté à éditer des algorithmes pouvant répondre à ce besoin a ouvert une la voie vers le traitement des données du problème et de ses variables de décisions afin de rendre l'exécution des algorithmes d'optimisation plus rapide. Ceci est examiné dans plusieurs travaux de la littérature dans le cadre de l'optimisation à grand échelle [39].

Partant du principe '*diviser pour régner*', certaines approches reposent sur la décomposition de l'instance du problème en plusieurs sous instances. Une récente (2013) proposition basée sur l'utilisation des vaccins artificiels a été formulée [40] puis appliquée au TSP [41]. Plus tôt, un algorithme qui combine entre la théorie de la résonance adaptative (réseaux de neuronnes) et une variante de l'algorithme d'optimisation de Lin–Kernighan a été présenté [42] pour résoudre de très grandes instances du TSP. Une autre démarche basée sur le partitionnement de graphes à plusieurs niveaux par l'algorithme k-mean [43].

D'autres approches réduisent la taille d'une instance en retirant quelques uns de ses éléments ou ses variables de décisions. Dans le cas du TSP, une instance peut être réduite en retirant quelques arêtes qui connectent les couples de villes pour obtenir un graphe moins dense. La triangulation de Delaunay [44] peut être appliquée pour les instances avec des coordonnées Euclidiennes en deux dimensions. Pour des dimensions supérieurs (k), les arbres kD (kD-trees) [45] peuvent être construits pour garder quelques arêtes de chaque quadrant. Plus récemment, Taillard et Helsgaun [46] ont proposé pour le TSP une construction de graphe réduit à partir de l'union des arêtes de plusieurs bonnes solutions.

Ces méthodes citées peuvent être aussi adoptées par le GTSP tout en prenant en compte ses contraintes afin de de ne pas vider l'espace de recherche. D'autres approches spécifiques au problème ont été proposées dans la littérature, elles seront discutées dans le chapitre 4.

1.4 Quelques méthodes de résolution des POC

La décomposition d'un POC en sous-problèmes ou la réduction de sa taille permet donc de le résoudre plus rapidement par rapport aux instances originales, mais ne fournissent pas de solutions. Ce sont des méthodes de pré-traitement qui viennent en amont d'une méthode d'optimisation quelle qu'elle soit.

À partir de cette section et jusqu'à la fin du chapitre, les différentes méthodes d'optimisation seront présentées, à commencer des méthodes exactes qui permettent d'obtenir la meilleure solution (optimale) puis les différentes approches approximatives, tout en dressant un état de l'art des différents travaux consacrés aux TSP et GTSP.

1.4.1 Algorithmes exactes

Les méthodes exactes fournissent la solution optimale pour n'importe quelle instance. Plusieurs approches sont proposées pour le TSP [47], la recherche naïve (ou exhaustive) qui parcourt l'espace de recherche en entier en fait partie. Les algorithmes *Branch-and-Bound* [48], [49] et *Branch-and-Cut* [50] sont les plus citées dans la littérature pour plusieurs POC y compris le TSP [51]–[53] et le GTSP [54].

Ces méthodes sont connues pour leurs complexités assez larges, factorielles ou exponentielles. Les Problèmes tels que le TSP étant NP-difficiles, ne peuvent être résolus à l'optimalité en temps polynomial. Les approches exactes demandent donc un temps d'exécution trop important qui devient impraticable pour les instances larges ou même moyennes. Padberg et Rinaldi [53] ont proposé une approche basée sur le *Branch-and-Cut* pour résoudre les instances larges du TSP. Burkhovetskiy et al. [55] ont récemment publié un algorithme exacte et parallèle pouvant résoudre une instances de dix mille villes en quelques minutes.

Le solveur ayant le plus de succès pour le TSP est *Concorde*[1], un programme capable de fournir les solutions optimales dans un temps correct. Il reste tout de même limité comme tous les autres approches, les instances de deux mille villes par exemple nécessitent plus de cinq jours pour être résolues.

1.4.2Approches approximatives

L'application d'un algorithme exacte pour résoudre un POC étant très limitée, l'utilisation d'un algorithme approximatif ou d'une heuristique est une alternative pour obtenir des solutions dans des délais corrects. Le résultat de l'algorithme peut être évalué par $C/C* \leq k$. C étant le fitness de la solution obtenue par l'algorithme approximatif, C^* est celui de la solution optimale, et k est la limite supérieure du rapport entre C et C^* dans les pires conditions. La valeur de $k > 1.0$. Plus il se rapproche de 1, meilleur est l'algorithme.

Ces approches sont de complexité polynomiale et permettent de tracer une borne inférieure (lower bound) qui sera nécessaire pour évaluer la qualité d'une solution en l'absence de la solution optimale que les méthodes exactes peuvent fournir. La meilleure borne inférieure atteinte pour le TSP est celle de Held & Karp [56], [57], elle est 1% plus grande que l'optimum pour les instances Euclidiennes générées aléatoirement, et à 2% des instances du benchmark TSPLIB [58].

1 http://www.math.uwaterloo.ca/tsp/concorde/index.html

1.4.3Méthodes de construction

La plupart des méthodes d'optimisation requièrent en entrée une solution initiale π_0 à améliorer, ils appelées aussi des méthodes d'améliorations. π_0 peut être construite aléatoirement, en sélectionnant itérativement une ville au hasard, ou par le biais d'une méthode dite de construction [59], qui construit des solutions à partir de une ou plusieurs règles bien définies, mais ne fait aucune amélioration une fois le circuit construit. En d'autres termes, la solution est construite de manière itérative sans apporter de modifications aux éléments déjà construits. Ci-dessous quelques unes des méthodes de construction pour le TSP les plus fréquemment utilisées dans la littérature, dont une ayant fait l'objet d'une étude empirique [60] dans cette thèse.

Le plus proche voisin: La méthode du plus proche voisin [61] est la méthode la plus simple dédiée au TSP. Le voyageur part d'une ville choisie arbitrairement et se dirige vers la ville la plus proche, et ainsi de suite, toute en s'assurant qu'aucune des villes déjà visitées n'est revisitée. Lorsque toutes les n villes (taille de l'instance) sont visitées, le voyageur revient à la ville de départ.

La méthode d'insertion: Elle [61] est basée sur l'insertion d'une ville dans un tour partiel (V). Il existe plusieurs règles et critères d'insertion. La plus simple consiste à sélectionner une ville parmi celles non visitées (U) et à l'insérer à côté de la ville la plus proche (resp. la plus éloignée) de celles qui sont dans V. Une fois insérée, la ville sélectionnée est retirée de U. Une autre approche consiste à insérer la ville dans V de sorte à minimiser le fitness du nouveau sous-tour. Le processus est répété jusqu'à ce que U devienne vide.

La méthode Multi-Fragment (MF): Appelée aussi gloutonne, MF [61], [62] est une approche intéressante qui considère les arêtes comme le paramètre principal dans la construction du tour. L'idée est de sélectionner les arêtes en fonction de leurs distances respectives. L'objectif du TSP étant de trouver un tour dont le fitness est minime, MF vise à sélectionner les plus petites arêtes possibles.

Une étude détaillée sur MF [60] est effectuée dans le cadre de cette thèse. Les références antécédentes étant évasives sur l'algorithme et insuffisantes face aux questions de certains chercheurs[2].

1.5 La recherche locale

L'arrivée de la recherche local en optimisation combinatoire remonte à la fin des années 1950, quand les premiers algorithmes basés sur les mouvements et échanges des arrêtes ont été introduits pour le TSP [63]–[65]. Nicholson [66] a généralisé plus tard le principe des échanges en introduisant une classe de problèmes à permutations comme le routage de véhicules ou encore l'ordonnancement.

Avant d'entamer la discussion sur la recherche locale, la définition d'un problème et d'un voisinage (cf. définition 1.1) sont nécessaires pour l'assimilation de ce qui va suivre.

Définition 1.1: Soit **(S,f)** une instance d'un problème d'optimisation combinatoire. Une fonction de voisinage est une application $\mathbf{N} : \mathbf{S} \rightarrow \mathbf{2^S}$ qui définit pour chaque solution $\mathbf{i} \in \mathbf{S}$ un ensemble $\mathbf{N(i)} \subseteq \mathbf{S}$ de solutions qui sont en quelque sorte *proche de* **i**. L'ensemble $\mathbf{N(i)}$ est dit voisinage de la solution **i**, et chaque solution $\mathbf{j} \in \mathbf{N(i)}$ est un voisin de **i**.

En gros, la recherche locale démarre avec une solution initiale (π_0) et essaie de trouver une solution meilleure en cherchant continuellement dans le voisinage de π_0. La version basique de la recherche locale est dite l'*amélioration itérée* (iterative improvement), elle cherche dans le voisinage d'une solution initiale π_0 une solution de meilleur coût π. π, s'il existe, remplace la solution courante pour continuer la recherche. Sinon, l'algorithme retourne la solution courante, qui sera donc un optimum local tel qu'il est définit ci-dessous.

Définition 1.2: Soit **(S,f)** une instance d'un problème d'optimisation combinatoire, et **N** une fonction de voisinage. Une solution $\mathbf{i'} \in \mathbf{S}$ est dite optimum local (minimal) par rapport à **N** si $\mathbf{f(i')} \leq \mathbf{f(j)} \forall \mathbf{j(i')}$.

2 https://cs.stackexchange.com/questions/65166/implement-multi-fragment-heuristics-for-the-traveling-salesman-problem/76720

1.5.1Les mouvements/voisinages

À noter que les optimums locaux dépendent de la fonction de voisinage utilisée. Les voisinages dépendent quand à eux du problème pris en considération. Trouver les bonnes fonctions de voisinage qui permettraient d'avoir des optimums locaux de bonne qualité est un challenge dans le domaine de la recherche locale. On trouve dans la littérature plusieurs exemples de fonctions de voisinage pour chaque problème d'optimisation combinatoire [67].

Le TSP en fait partie. Les plus simples voisinages sont basés sur l'échange (swap) des positions de k ($k \leq 2$) villes, ou l'insertion de k ($k \leq 1$) ville(s) dans k différente(s) positon(s). Les plus populaires sont les voisinages à k ($k \leq 2$) échanges, ou k-opt, qui consistent en le découpage d'une solution en k sections en supprimant k arrêtes, puis en reconnectant les sections de sorte à avoir une nouvelle solution faisable et différente de la première.

1.5.2Avantages et Inconvénients

La recherche locale met en avant deux grands avantages permettant d'obtenir des résultats en un temps constant ($\boldsymbol{O}(1)$) quelque soit la méthode de voisinage utilisée:

- Coût de génération de la solution voisine: en échangeant seulement les nœuds impliqués dans l'échange à partir de la solution initiale.
- Coût d'évaluation d'une solution voisine: en utilisant la formule Δ qui correspond à chaque voisinage. En effet, une solution peut être évaluée à partir de sa voisine étant donné que le nombre d'arêtes qui différent de l'une à l'autre dépend de k.

La recherche locale présente aussi des inconvénients, ceci est dû principalement à la répartition de l'espace des solutions:

- Évolution exponentielle du nombre de minima locaux en fonction de la taille de l'instance.

- Une vision très courte sur l'espace des solutions. Le nombre de voisins pour chaque solution est très petit par rapport à la taille de l'espace.
- Problème d'optimalité locale (général)

En effet, la recherche locale explore une portion relativement très petite (quelque soit la fonction de voisinage) par rapport à la taille de l'espace de recherche, ce qui condamne rapidement la recherche à un optimum local π. Pour explorer d'autres voisinages, la recherche locale doit recommencer à partir de nouvelles solutions initiales (π_0) différentes de celle(s) déjà sollicitée(s).

Plusieurs métaheuristiques basées sur la recherche locale ont été proposées pour pallier au problème d'optimalité locale et éviter de redémarrer la recherche d'une nouvelle solution initiale. La recherche locale itérée [68], la recherche à voisinage variable [69] et le recuit simulé [70] font partie des multiples métaheuristiques fondées autour du principe de la recherche locale et/ou de celui du voisinage.

1.6 Les métaheuristiques

En plus des métaheuristiques basées sur la recherche locale, d'autres approches seront présentées dans cette section. Plusieurs classifications existent pour les métaheuristiques afin de différencier entre chacune d'elles.

1.6.1 Différentes classifications des métaheuristiques

Méthodes à solution unique / à population: Cette classification divise les métaheuristiques selon le nombre de solutions optimisées en parallèle. Les approches à solution unique améliorent une solution depuis la première itération en la faisant évoluer dans l'espace de recherche selon l'algorithme adopté. Cette évolution repose essentiellement sur la notion de mouvement vu précédemment. Le recuit simulé, la recherche avec tabous ou encore la recherche à voisinage variable sont des méthodes sollicitant une solution à la fois.

L'autre classe et celle des méthodes fondées sur la notion de population: plusieurs solutions sont améliorées en même temps et en une seule

itération. Les algorithmes génétiques, l'optimisation par les colonies de fourmis ou par essaims particulaires appartiennent à cette classe.

Métaheuristiques avec / sans mémoire: Certaines méthodes utilisent une mémoire pour enregistrer des informations sur la recherche pendant les itérations antécédentes. Elle peut être de court terme, pour enregistrer par exemple les t derniers mouvements sollicités ou les dernières solutions visités; ou bien de long terme comme les variables décrivant l'état de la recherche (la température dans le recuit simulé).

Fonction objectif statique / dynamique: La majorité des métaheuristiques disposent d'une fonction objectif statique, elle reste intacte durant le processus de l'optimisation. Certains algorithmes, comme la recherche locale guidée, altèrent la représentation du problème en introduisant l'information collectée durant la recherche,directement au sein de la fonction objectif.

D'autres classifications peuvent être relevées dans la littérature. Les métaheuristiques sont classées aussi selon l'inspiration de la nature (inspirées ou non inspirées) ou par le nombre de voisinages sollicités (un seul voisinage ou plusieurs). La figure 1.2 ci-dessous est un diagramme proposé par Johann Dréo [71] qui '*tente de présenter où se placent quelques-unes des méthodes les plus connues*'.

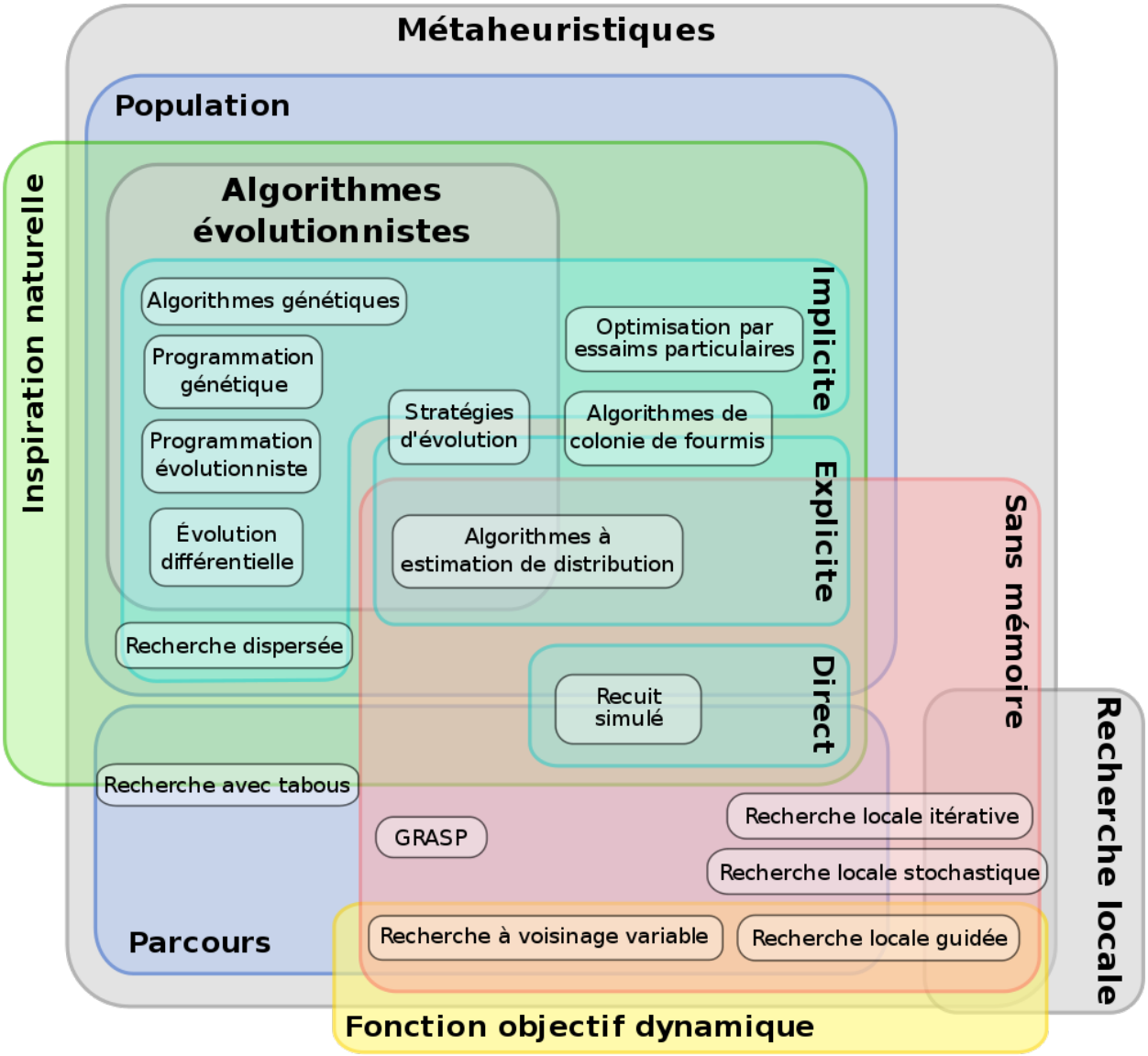


Figure 1.2: Classification des métaheuristiques

1.6.2Les métaheuristiques basées sur la recherche locale

Le problème d'optimalité locale discuté précédemment a été taclé dans plusieurs travaux. Plusieurs travaux ont été proposés pour continuer la recherche à partir du minimum local trouvé.

1.6.2.1 La recherche locale itérée (ILS)

La recherche locale itérée (Iterated Local Search (ILS)) [68] échappe de l'optimum local en lui appliquant une modification puis relance la recherche depuis cette nouvelle solution. ILS est une idée très simple reprise dans plusieurs travaux de recherches. Elle est définie, comme il est montré sur l'algorithme 1.1, par trois composantes: RechercheLocale, Modifier et Acceptation. La première effectue une descente avec une fonction de voisinage N, la deuxième perturbe l'optimum local obtenue par RechercheLocale, tandis que la dernière accepte/rejette le nouvel optimum local.

Algorithme 1.1: Iterated Local Search

générer solution alétoire (ou gloutonne) s
s = RechercheLocale(s)
$s_{best} = s$
tant que (coditions d'arrêt non rencontrés) **faire**
    s' = Modifier(s, *historique*)
    s'' = RechercheLocale(s')
    si (f(s'')<f(s_{best})) **alors**
        $s_{best} = s''$
    s = Acceptation(s,s'',*historique*)
    fin si
fin tant que
retourner s_{best}
fin

Étant la base de l'une des contributions de cette thèse, elle sera détaillée au début du prochain chapitre avant d'aborder une des métaheuristiques dont ILS est la base.

1.6.2.2 La recherche à voisinage variable (VNS)

La recherche à voisinage variable (Variable Neighborhood Search (VNS)) proposé par Mladenović et Hansen [69] change de structure de voisinage une fois un optimum atteint durant la recherche. L'idée est de changer la fonction de voisinage qui a mené vers l'optimum local courant vers une nouvelle fonction, étant donné qu'un optimum local est défini par rapport à une certaine structure de

voisinage et ne l'est pas forcément pour une autre structure. Sauf l'optimum global qui est optimum local pour n'importe quelle structure. Le changement de structure de voisinage $(\mathcal{N}_k, k = 1, ..., n)$ peut se faire de différentes manières:

- La première, présentée dans l'algorithme 1.2, utilise la même structure $(\mathcal{N}_1)$ dans la recherche locale et fait intervenir les autres $(\mathcal{N}_k, k = 2, ..., n)$ dans des perturbations, dans le même principe que ILS. La sélection du mouvement est faite aléatoirement dans $\mathcal{N}_k$. L'enchaînement des structures ne suit pas forcément une relation d'ordre (la taille du voisinage par exemple).

Algorithme 1.2: Recherche à Voisinage Variable
Entrées: Structures de voisinage N_k, k=1,...,n
générer solution initiale s_0
k = 2
s = RechercheLocale(s_0)
$s_{best} = s$
tant que (conditions d'arrêt non atteintes) **faire**
    s' = SolutionAleatoire($N_k(s_{best})$)
    s'' = RechercheLocale(s')
    si ($f(s'') < f(s_{best})$) **alors**
        $s_{best} = s''$
        k = 2
    sinon
        k = k+1
    fin si
fin tant que
retourner s_{best}
fin

- La deuxième approche fait changer la fonction de voisinage dans le processus de la recherche locale. Cette variante est appelée Descente à Voisinage Variable (Variable Neighborhood Descent (VND)). La solution finale est optimum local pour toutes les structures $\mathcal{N}_k, k = 1, ..., n$ sollicitées et qui sont triées par taille de voisinage du plus petit au plus grand.

Algorithme 1.3: Descente à Voisinage Variable
Entrées: Structures de voisinage N_k, k=1,...,n

```
s = SolutionInitiale()
k = 1
tant que (k ≤ n) faire
        s' = RechercheLocale(s)
        si (f(s')<f(s)) alors
                s = s'
                k = 1
        sinon
                k = k+1
        fin si
fin tant que
retourner s
fin
```

1.6.2.3 La recherche tabou (TS)

Après plusieurs itérations de descentes, la recherche locale se confronte au risque d'atteindre des solutions déjà visitées. L'exploration de l'espace de recherche deviendra alors condamné dans un circuit fermé et ne donnera plus aucun nouvel optimum local. La recherche tabou (Tabu Search (TS)) [72], [73] est une métaheuristique qui vient pour suppléer à ce risque. Une mémoire à court terme enregistre les t derniers mouvements pour empêcher la recherche de revenir à ces solutions, seules celles absentes de la liste des tabous pourront être choisies.

Algorithme 1.4: Recherche Tabou

```
Entrées: taille de la liste tabou (t), liste tabou (LT) vide
s_b = SolutionInitiale()
tant que (conditions d'arrêt non atteintes) faire
        s' = s ∈ N(s_b) ∧ f(s) ≤ f(s_n) ∀s_n ∈ N(s_b) ∧ s ∉ LT
        si (f(s')<f(s_b)) alors
                s_b = s'
                ajouter s_b à LT
                tant que (taille(LT)>t) faire
                        retirer de LT la plus ancienne solution
                fin tant que
        fin si
```

fin tant que
retourner s_b

1.6.2.4 Le recuit simulé (SA)

Le recuit simulé (Simulated Annealing (SA)) est inspiré du processus de recuit en métallurgie mis en place par Kirkpatrick et al. [70]. Dans ce processus naturel, un matériau est chauffé puis refroidi lentement dans des conditions maîtrisées pour augmenter la taille des cristaux dans le matériau et réduire leurs défauts. Cela a pour effet d'améliorer la résistance et la durabilité du matériau. La chaleur augmente l'énergie des atomes, ce qui leur permet de se déplacer librement, et le programme de refroidissement lent permet de découvrir et d'exploiter une nouvelle configuration à faible consommation d'énergie.

Chaque configuration d'une solution dans l'espace de recherche représente une énergie interne différente du système. Le chauffage du système entraîne un assouplissement des critères d'acceptation des solutions voisines de la solution courante. Lorsque le système est refroidi, les critères d'acceptation sur le voisinage sont réduits pour se concentrer sur l'amélioration des mouvements.

1.6.2.5 La recherche locale par acceptation tardive (LAHC)

La recherche locale par acceptation tardive (Late Acceptance Hill Climbing (LAHC)) a été récemment introduite par Burke & Bykov [74]–[76]. Elle retarde la comparaison entre les solutions voisines et compare à partir de l'itération T chaque nouvelle solution candidate avec une explorée depuis T d'itérations. Ce nombre est le seul paramètre de cette métaheuristique, il influence directement le temps d'exécution.

1.6.3Les algorithmes génétiques

Un algorithme génétique (AG) est une métaheuristique inspirée de la nature, plus exactement du processus de sélection. Il fait partie de la classe des algorithmes évolutionnaires qui inclut entre autres la programmation génétique, les stratégies d'évolution et les algorithmes culturels.

Les AG sont des méthodes stochastiques dont les modèles de recherche suivent le phénomène naturel de l'héritage génétique ainsi que la théorie Darwinienne de l'évolution. L'idée est de faire ce que fait la nature, mimer une population d'individus du même genre mais qui se distinguent par leurs caractéristiques et performances. Le vocabulaire des AG est lui aussi emprunté à la génétique naturelle. Une population est ainsi constituée de chromosomes, chacun représente une solution du problème, et chaque chromosome se compose de gênes. Dans le cas du TSP, une population de chromosomes est un ensemble de tours, chaque gêne est un nœud ou une connexion (arête) entre deux nœuds.

La première phase est dans un AG est l'initialisation. Une population (S) est créée puis évaluée à travers la fonction de fitness du problème en question. Une population de parents (P) est ensuite constituée à partir de S dans la phase de **sélection** où les chromosomes de S ayant les meilleurs fitness ont plus de chances d'être tirés et faire partie de P. Vient après la phase de **reproduction** où des certains individus de la population subissent un **croisement** et/ou une **mutation**.

Le croisement produit une population d'enfants (O) à partir de la recombinaison des solutions de P en déployant un ou plusieurs opérateurs de croisement. Les enfants hériteront de certaines propriétés de leurs parents, en fonction du croisement choisi. Plusieurs opérateurs de croisement peuvent être recensés dans la littérature, parmi les plus cités figurent le croisement à un point (*One point Crossover (1X)*), le croisement uniforme (*Uniform Crossover (UX)*), ou encore le croisement par partitionnement (*Partition Crossover (PX)*). Une amélioration de ce dernier a été proposé récemment par Chicano et al. [77].

La mutation est un opérateur unaire, il crée de nouveaux individus en modifiant des chromosomes de la population. Le but est de perturber la population à travers ses individus afin de conserver sa diversité et explorer une nouvelle région de l'espace de recherche. Les opérateurs de mutation sont similaires aux mouvements présentés précédemment dans la recherche locale, une solution s peut être mutée en s' par un certain mouvement mvt et inversement implique que s et s' sont voisines dans le voisinage structurée par mvt.

La phase de remplacement est la dernière. Une nouvelle population est construite par la sélection de nouveaux individus produits par le croisement et la mutation et d'autres individus de la dernière génération. D'autres générations seront produites tant que la condition d'arrêt n'est pas atteinte. La figure 1.3 ci-dessous décrit les différentes phases du processus d'une génération.

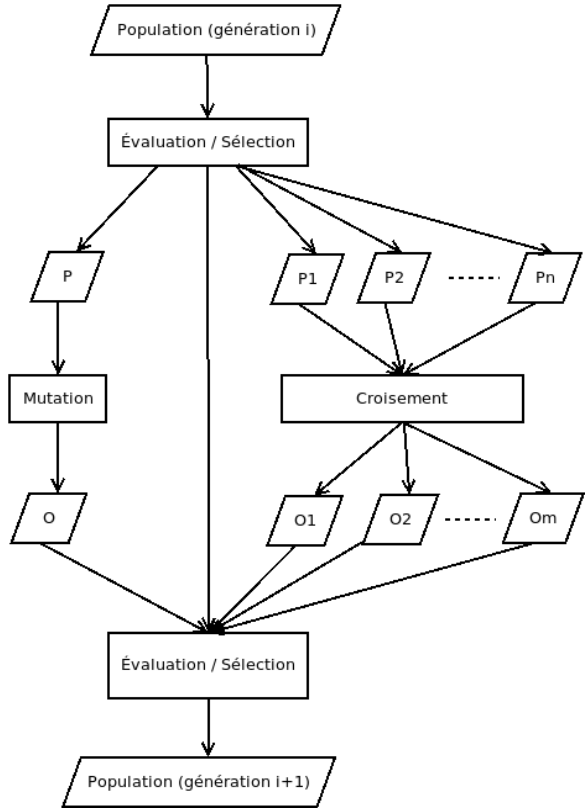


Figure 1.3: Processus d'évolution d'une population

1.6.4Autres métaheuristiques

Le recuit simulé (Simulated Annealing (SA)): Le recuit simulé est inspiré du processus de recuit en métallurgie mis en place par Kirkpatrick et al. [70]. Dans ce processus naturel, un matériau est chauffé puis refroidi lentement dans des conditions maîtrisées pour augmenter la taille des cristaux dans le matériau et réduire leurs défauts. Cela a pour effet d'améliorer la résistance et la durabilité du matériau. La chaleur augmente l'énergie des atomes, ce qui leur permet de se déplacer librement, et le programme de refroidissement lent permet de découvrir et d'exploiter une nouvelle configuration à faible consommation d'énergie.

Chaque configuration d'une solution dans l'espace de recherche représente une énergie interne différente du système. Le chauffage du système entraîne un assouplissement des critères d'acceptation des solutions voisines de la solution courante. Lorsque le système est refroidi, les critères d'acceptation sur le voisinage sont réduits pour se concentrer sur l'amélioration des mouvements.

Optimisation par essaims particulaires (Particle swarm optimization (PSO)): Une technique d'optimisation stochastique basée sur la population. Elle est développée par Eberhart et Kennedy [78], [79], inspirée par le comportement social du flocage d'oiseaux ou de la pisciculture.

PSO partage de nombreuses similarités avec les AG. Le système est initialisé avec une population de solutions aléatoires et cherche les optimums en mettant à jour les générations. Contrairement aux AG, PSO n'a pas d'opérateurs génétiques tel que le croisement ou la mutation. Dans PSO, les solutions potentielles, appelées particules, traversent l'espace de recherche en suivant les meilleures particules actuelles.

Algorithme de colonies de fourmis (Ant Colony Optimization (ACO)): ACO étudie des systèmes artificiels qui s'inspirent du comportement des colonies de fourmis et qui sont utilisés pour résoudre des problèmes d'optimisation discrète.

Le premier système ACO a été introduit par Marco Dorigo dans sa thèse de doctorat [80] et s'appelait à l'époque « *Ant System* » (AS). Il a été initialement appliqué au TSP [81] et au QAP [82]. Dorigo, Gambardella et Stützle ont travaillé sur différentes extensions du paradigme AS. Dorigo et Gambardella ont proposé le système de colonie de fourmis (Ant Colony System (ACS)) [83], tandis que Stützle et Hoos ont proposé le système de fourmis MAX-MIN (MAX-MIN Ant System (MMAS)) [84]. Les deux ont été appliqués au TSP symétrique et asymétrique et ont donné de très bons résultats. Dorigo, Gambardella et Stützle ont également proposé de nouvelles versions hybrides d'optimisation de colonies de fourmis avec recherche locale [85], [86].

1.6.5Les approches hybrides

L'hybridation de différentes métaheuristiques est aujourd'hui très répandue [87], notamment en ce qui concerne l'utilisation de méthodes de recherche locales dans les méthodes basées à population comme les AG, appelées algorithmes mémétiques [88], [89]. Des approches comme le calcul évolutionnaire et ACO font souvent appel à des procédures de recherche locale pour affiner les solutions qui sont générés au cours du processus de recherche.

Cela peut être attribué au fait que ces méthodes inspirées par la nature sont utiles pour l'exploration de l'espace de recherche et l'identification de régions avec des solutions de bonne qualité. Lors de l'initialisation, ils essaient généralement d'avoir une image globale de l'espace de recherche. Ensuite, au cours du processus de recherche, ils concentrent successivement la recherche sur des régions plus prometteuses de l'espace de recherche. Cependant, ils ne sont généralement pas aussi efficaces en ce qui concerne l'exploitation de l'espace de recherche, c'est-à-dire la recherche des meilleures solutions dans ces sous-espaces de haute qualité. D'autre part, la force de la recherche locale réside dans la capacité à trouver rapidement de meilleures solutions à proximité de solutions de départ données.

Une métaheuristique peut aussi être hybridée avec une méthode d'optimisation autre qu'une métaheuristique. La programmation par contraintes a été ainsi combinée avec la recherche locale (Constraint Programming-based Large Neighborhood Search) [90] et aussi avec ACO [91]. *Iterated dynasearch* est un autre exemple d'hybridation, il combine entre ILS et la programmation dynamique qui sert à l'exploration du voisinage [92].

1.7 Conclusion

Résoudre un POC passe par le choix d'une méthode d'optimisation parmi toutes celles disponibles dans la littérature, ou la conception d'une nouvelle. Les méthodes exactes et les algorithmes approximatifs sont deux approches qui assurent de résoudre le problème avec un écart bornée de la solution optimale, et qui est nul pour les approches exactes. L'ambition de trouver des solution encore

meilleures, voir même l'optimalité, a fait en sorte l'apparition des métaheuristiques qui peuvent donner de très bon résultats sans pour autant les garantir pour n'importe quelle instance.

De nouvelles métaheuristiques sont publiées continuellement dans la littérature, la plupart du temps se basant sur des métaheuristiques antécédentes. Elles essaient de proposer une nouvelle contribution et s'affirmer comme étant une amélioration par rapport à leurs prédécesseurs respectifs. Une de ces métaheuristiques sera le sujet des deux prochains chapitres, où seront présentées de nouvelles contributions et adaptations pour le TSP puis le GTSP.

Chapitre 2: Adaptation d'une métaheuristique basée sur ILS pour le TSP

2.1 Introduction

La recherche locale itérée (Iterated Local Search (ILS)) fait partie des métaheuristiques les plus reprises dans la littérature. L'idée de base et la simplicité de l'approche ont conduit les chercheurs à proposer de nouvelles (méta)heuristiques en la sollicitant dans des approches hybrides et/ou en améliorant certaines de ses composantes.

U. Benlic et J. Hao se sont eux aussi intéressés récemment à ce framework, proposant une métaheuristique avec une nouvelle stratégie de perturbation pour fuir les optimums locaux. Mise à l'essai et appliquée sur plusieurs POC, l'idée apportée dans la phase de perturbations ainsi que les bons résultats fournis étaient la motivation principale derrière l'intérêt porté à cette métaheuristique dans cette thèse.

Ce deuxième chapitre présente une adaptation de cette métaheuristique au TSP. ILS, présentée brièvement dans le chapitre précédent, est développée dans la prochaine section avant d'introduire la nouvelle approche et exposer la contribution apportée pour résoudre (au mieux possible) le TSP.

2.2 La Recherche Locale Itérée

Pour continuer la recherche locale sans redémarrage, ILS propose d'appliquer une modification sur l'optimum local π en utilisant une fonction de voisinage ***f'*** plus large que celle sollicitée pendant la recherche locale. Un tel mouvement provoqué par ***f'*** est censé produire une nouvelle solution π' qui se situera (en général) en dehors du voisinage de π et qui pourra mener vers un nouvel optimum local. La recherche locale itérée (Iterated Local Search) est développée autour de cette idée afin de résoudre les problèmes d'optimisation combinatoire.

ILS[93]–[95] est une métaheuristique simple, mais puissante, destinée à améliorer les performances des algorithmes de la recherche locale. La simplicité vient du fait que seules quelques lignes de (pseudo-)code doivent être ajoutées à une procédure de recherche locale déjà existante pour implémenter un algorithme ILS. On peut également présumer les bonnes performances de la recherche locale itérée par rapport au redémarrage de la recherche locale à partir d'une nouvelle solution générée au hasard. Ceci est appuyé par le fait que les algorithmes basés sur la métaheuristique ILS sont parmi les méthodes les plus performantes pour de nombreux problèmes d'optimisation combinatoire comme le problème du voyageur de commerce[93], [96].

Pour appliquer ILS à un problème donné, trois composantes doivent être définies. L'une d'entre elles est "**Modifier**", qui perturbe la solution actuelle π (qui est généralement un optimum local) conduisant vers une solution intermédiaire π'. La perturbation est aussi souvent citée en tant que *saut*, les deux termes seront utilisés dans la suite de ce mémoire. Ensuite, la "**RechercheLocale**" est appliqué sur π' pour atteindre un minimum local π''. Enfin, il faut décider quelle solution doit être choisie pour la prochaine étape de modification. Cette décision est prise en fonction d'un critère d'"**Acceptation**" qui prend en compte les solutions précédentes π, la nouvelle solution candidate π'' et éventuellement l'historique des recherches.

La perturbation introduite par la procédure **Modifier** est utilisée avec l'objectif de quitter la solution locale optimale actuelle. Souvent, il s'agit d'un mouvement choisi au hasard dans un voisinage d'un ordre plus élevé que celui utilisé dans la recherche local, ou bien un mouvement que la recherche locale ne peut inverser en une itération. La *puissance* du saut et sa *direction* peuvent être choisis en fonction de l'historique de la recherche locale. Le critère d'acceptation est utilisé pour décider quelle solution est perturbée à la suite. Le choix du critère d'acceptation est important, car il contrôle l'équilibre entre l'exploitation et l'exploration. L'exploitation est réalisée, par exemple, en acceptant seulement le meilleur optimum local dans la méthode **Modifier**. En fait, un tel critère d'acceptation est appliqué dans la plupart des implémentations basées sur ILS. Un

tel choix peut être préférable si des solutions encore meilleures peuvent être trouvées près de l'optimum local actuel. L'exploration quand à elle peut être réalisée, dans le cas extrême, en acceptant à chaque itération le nouvel optimum local π'', ce qui peut paraître comme un déplacement aléatoire dans l'espace de recherche.

L'algorithme 2.1 ci-dessous décrit le framework ILS. À noter que ce dernier est présenté dans la littérature sous différentes appellations: *large step Markov chains* [94], *chained local optimization* [93], *simple variable neighborhood search* [97], *iterated Lin-Kernighan* [96].

Algorithme 2.1: Recherche Locale Itérée

```
générer solution alétoire (ou gloutonne) s
s = RechercheLocale(s)
s_best = s
tant que (coditions d'arrêt non rencontrés) faire
        s' = Modifier(s, historique)
        s'' = RechercheLocale(s')
        si (f(s'')<f(s_best)) alors
                s_best = s''
        s = Acceptation(s,s'',historique)
        fin si
fin tant que
retourner s_best
fin
```

Comme expliqué précédemment, la recherche locale commence par une solution initiale π_0 et se dirige vers un optimum local π tout en passant par des solutions intermédiaires π'_i au fil des itérations. En effet, l'optimum local π trouvé durant la recherche peut être atteint non seulement par π_0 ou les π'_i (qui peuvent être elles aussi des solutions initiales), mais aussi par d'autres solutions dans l'espace de recherche.

Chaque optimum local a un *rayon* (r) de solutions qui convergent vers lui par le biais de la même recherche locale et la même fonction de voisinage. r peut être évalué par le nombre de solutions qui convergent vers π ou par la taille du sous espace de recherche (S_k), appelé bassin d'attraction. Ce dernier varie d'un optimum

local à un autre, plus il est grand plus l'optimum local est attractif. La figure 2.1[3] ci-dessous est une représentation d'un espace de recherche composé de huit optimums locaux (de 'a' à 'h'), chacun ayant son propre bassin d'attraction.

L'objectif de la recherche locale itérée est de passer d'un optimum local à un nouveau bassin d'attraction qui n'appartient pas à celui-ci (afin de trouver un nouvel optimum local), ce qui n'est pas toujours aussi simple. En effet, la taille de certains bassins d'attraction rendent insuffisant les modifications proposés par les versions basiques du framework ILS.

Plusieurs métaheuristiques basées sur ILS ont fait apparition depuis la naissance du framework, attaquant un bon nombre de problèmes d'optimisation combinatoires tout en essayant d'apporter une contribution notable à la recherche locale itérée. L'une des plus récentes est "*Breakout Local Search*", proposée par Una Benlic et Jin-Kao Hao [98] en 2012.

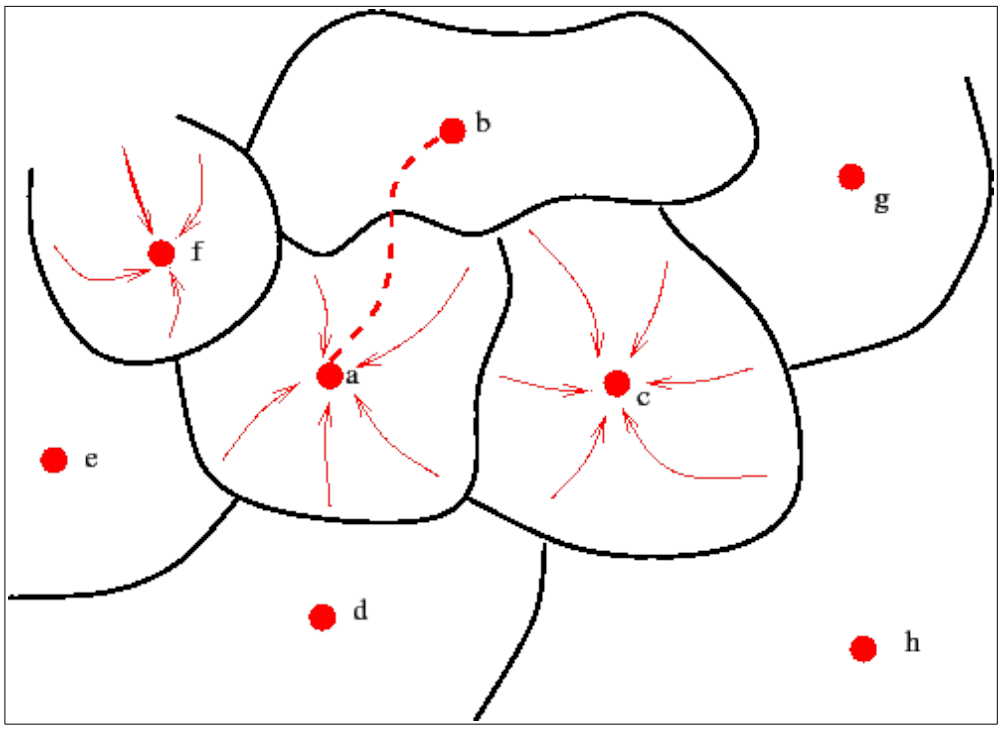


Figure 2.1: Bassins d'attraction des optimums locaux

3 Attraction basins around locally optimal points - Roberto Battiti (licence CC BY-SA 3.0)

2.3 La Recherche Locale d'Évasion

La Recherche Locale d'Évasion (Breakout Local Search (BLS)) est une métaheuristique récente (par rapport à la date de début cette thèse: 2014) ayant attaquée beaucoup de POC. Le premier ayant été résolu par BLS est le *Minimum Sum Coloring Problem* [99], puis le *Quadratic Assignment Problem* [100] et d'autres problèmes traitant la théorie des graphes [101]–[103]. Les bons résultats affichés sur ces travaux on conduit d'autres chercheurs à résoudre d'autres problèmes par BLS en adaptant la métaheuristique selon les spécificités du problème [104]–[106].

BLS alterne itérativement entre la recherche locale pour exploiter un voisinage et une stratégie de diversification spécifique pour se déplacer d'un bassin d'attraction vers un autre dans l'espace de recherche. L'originalité de BLS demeure dans son processus de diversification qui élit d'une façon adaptative entre deux types de perturbation (dirigée ou aléatoire) et règle en temps réel la force des perturbations en fonction de l'état courant de la recherche. On rappelle que BLS suit le schéma général du framework ILS et utilise le principe de la recherche tabou (Tabu Search) pour toute perturbation non aléatoire. À la différence de la recherche tabou [72], [73], BLS ne sollicite pas la liste tabou pendant les phases de recherche locale mais uniquement dans la phase de perturbation.

2.3.1 L'idée Générale

L'approche globale de BLS est de se déplacer d'un optimum local d'un voisinage à un autre en appliquant des sauts dont le nombre et les candidats impliqués sont déterminés de manière adaptative. BLS commence à partir d'une solution initiale π et effectue une recherche locale pour atteindre un nouvel optimum local. La recherche locale dans BLS est effectuée par le principe de la meilleure amélioration (*best improvement*), pour atteindre un optimum local. BLS tente après d'y échapper à cette attraction pour se déplacer vers un nouveau voisinage en appliquant une série (L) de mouvements (dont le nombre est borné par un minimum (L_0) et un maximum) sur l'optimum local actuel (π) qui devient perturbé et sert comme point de départ pour la prochaine itération de la procédure de recherche locale. Lorsque la procédure de recherche locale renvoie la même solution π, BLS la

perturbe plus fortement en incrémentant le nombre de sauts (L) pour appliquer la prochaine perturbation, et en choisissant des mouvements plus *explorateurs*. Après avoir visité un certain nombre (T) de voisinages sans améliorer la meilleure solution trouvée jusqu'ici (π_{best}), BLS effectue une perturbation beaucoup plus forte pour diriger une fois pour toute la recherche dans une nouvelle région plus distante dans l'espace de recherche. L'algorithme 2.2 ci-dessous présente l'idée générale de cette métaheuristique.

Algorithme 2.2: Breakout Local Search

```
π = SolutionInitiale()
π_best = π
π_p = π
ω = 0
L = L_0
tant que (critères d'arrêt non atteints) faire
        π = RechercheLocale(π)
        si f(π)<f(π_best) alors
                π_best = π; ω = 0
        sinon
                ω = ω+1
        fin si
        si ω>T alors
                π = perturbationForte(π)
                ω = 0
        sinon
                si (π==π_p) alors
                        L = L+1
                sinon
                        L = L_0
                fin si
                π = perturbationAdaptative(π)
        fin si
        π_p = π
fin tant que
retourner π_best
```

2.3.2Stratégie de diversification adaptative

Le processus de diversification adaptative a un rôle important sur les performances de BLS. Ce mécanisme combine (au minimum) deux perturbations pour orienter la recherche locale vers des régions inexplorées dans l'espace de recherche. La diversification adaptative sélectionne dynamiquement entre des mouvements/perturbations[4] dirigés et aléatoires selon l'état actuel de la recherche et la région dans laquelle elle se produit.

- La perturbation dirigée élit des mouvements qui n'ont pas été sollicités durant les γ dernières itérations (liste tabou) et qui dégradent au minimum la solution courante π.
- La perturbation aléatoire tire des mouvements de façon arbitraire, ce qui oriente la recherche locale dans une région inconnue. Elle apporte donc plus de diversité et est utile quand la recherche n'arrive pas à améliorer la meilleure solution trouvée.

BLS applique donc le plus souvent (avec une grande probabilité P) la perturbation dirigée (la plus faible) lorsque la recherche se trouve dans une région avec de très bonnes solutions. Quand la zone de recherche baisse de qualité, la probabilité P diminue progressivement pour introduire une diversification plus forte par des mouvements aléatoires. Ce changement adaptatif de probabilité P est illustré par l'équation 2.1, où ω est le nombre d'itérations consécutives n'ayant pas améliorées la meilleure solution trouvée (π_{best}). La probabilité P est borné par un minimum P_0 pour permettre à la perturbation dirigée d'être exécutée avec un minimum de chances.

$$P = \begin{cases} e^{-\omega/T}, & si(e^{-\omega/T} > P_0) \\ P_0, & sinon \end{cases} \qquad (2\ 1)$$

4 . D'autres perturbations peuvent être introduites pour améliorer la diversification adaptative, nous le verrons lors de l'adaptation pour le TSP.

2.3.3Les différents paramètres de BLS

BLS comprend plusieurs paramètres, Certains permettent de savoir la force des perturbations à effectuer, la taille de la liste tabou,... D'autres influencent sur le choix de la perturbation

- $\mathbf{L_0}$: Le nombre initial et minimal de sauts à effectuer durant une perturbation. Sa valeur est souvent assez petite pour ne pas diversifier la recherche dès le départ.
- $\mathbf{L_{max}}$: Le nombre de sauts à effectuer durant la perturbation forte. La grandeur de L_{max} dépend du mouvement à effectuer durant cette perturbation.
- **T**: Maximum d'itérations consécutives sans améliorer la meilleure solution trouvée (π_{best}). Une fois T atteint, la perturbation forte est déclenchée.
- $\boldsymbol{\gamma}$: La taille de la liste tabou. Elle est souvent choisie en fonction de la fonction de voisinage et dépend de la taille de l'instance [107].
- $\mathbf{P_0}$: La plus petite probabilité pour effectuer la perturbation dirigée.

2.4 Adaptation de BLS pour le TSP

Le TSP est, comme déjà cité dans le chapitre précédent, un problème référence pour présenter et valider les nouvelles approches algorithmiques. BLS ayant été adaptée et appliquée sur un grand nombre de POC, elle n'avait pas fait jusqu'alors l'objet d'une application pour le TSP.

L'adaptation proposée [108] a introduit des changements notables sur l'algorithme, sans pour autant changer son mécanisme de base.

2.4.1Fonction de voisinage

L'étude du paysage (*landscape*) du TSP [109] (ou de n'importe quel POC) permet d'identifier les fonctions de voisinage qui exploitent le mieux l'espace de recherche du problème en dépit des différentes topologies des instances. Cette étude est liée à deux notions primaires, la fonction de fitness et la corrélation.

La fonction de fitness détermine la qualité d'une solution, elle est mesurée dans le cas du TSP par la longueur du tour obtenu. Étant un problème de minimisation, une solution est meilleure avec un fitness plus petit. L'étude du paysage est toujours liée à cette notion, d'où le concept du paysage de fitness (*fitness landscape*).

Quant à la notion de corrélation [110], [111], elle a été introduite pour mesurer la difficulté du problème prenant en compte certains opérateurs. Elle détermine le degré de connectivité d'une solution à ses voisines.

Des travaux antérieurs sur le TSP [111]–[113] ont montré que le paysage est mieux corrélé pour l'opérateur *2Opt* que pour les autres. Nous l'avons donc sollicité dans ce travail, surtout dans la phase de la recherche locale.

2.4.2Perturbation forte

L'opérateur choisi pour la perturbation forte est le *Double Bridge* [114], il retire quatre arêtes (aucune n'est adjacente à l'autre), et construit le nouveau tour à partir des quatre tronçons en connectant la fin d'un tronçon avec le début du tronçon placé juste avant dans la solution précédente. La figure 2.3 ci-dessous illustre un exemple du mouvement *Double Bridge*. Ce mouvement est effectué L_{max} fois durant cette perturbation.

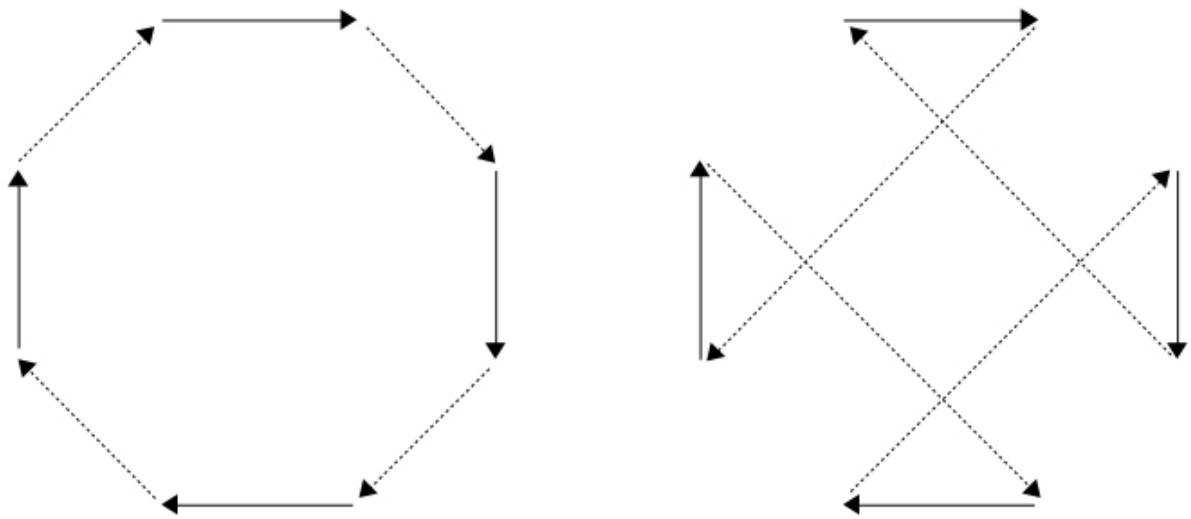


Figure 2.2: Mouvement Double Bridge sur une instance du TSP

2.4.3Perturbations dynamiques

Comme expliqué précédemment, BLS adopte une perturbation adaptative afin de choisir les meilleurs mouvements/perturbations selon l'état courant de la recherche. Cette stratégie a été améliorée dans cette adaptation en rajoutant une couche supérieur à la perturbation adaptative, en faisant varier le type de mouvement sollicité pour perturber l'optimum local courant.

Trois types de mouvements sont exécutés (tant que la recherche est bloquée dans le même bassin) à tour de rôle. L'ordre d'exécution dépend du nombre d'arêtes modifié par le type de mouvement:

1. Le premier est le mouvement 2Opt, caractérisé par l'échange de deux arêtes non adjacentes.
2. L'insertion est le deuxième mouvement, il déplace un nœud vers une nouvelle position. L'exécution de ce mouvement fait changer trois arêtes.
3. Finalement le mouvement d'échange, il s'agit d'échanger la position d'un nœud/ville avec un autre. Quatre arêtes sont impliquées dans ce mouvement.

Cette succession de mouvements est inspirée de la recherche à voisinage variable (*Variable Neighbourhood Search* (VNS)) [97] discutée dans le chapitre précédent.

Les algorithmes 2.3 et 2.4 ci-dessous exposent le mécanisme de la perturbation dynamique.

Algorithme 2.3: perturbationDynamique
Entrées: Optimum local **π**, nombre de sauts à effectuer **L**, indicateur du type de perturbation **Lw**, matrice d'historique **H**, nombre de mouvements effectués **Iter**, stratégie de perturbation **St**
Sortie: Une solution perturbée **π**

```
si Lw mod 3 == 0 alors
        π = perturbation(π,L,H,Iter,St,2Opt)
sinon si Lw mod 3 == 1 alors
        π = perturbation(π,L,H,Iter,St,insertion)
```

sinon Lw mod 3 == 2 **alors**
    π = perturbation(π,L,H,Iter,St,échange)
fin si
retourner π

Algorithme 2.4: perturbation
Entrées: Optimum local **π**, nombre de sauts à effectuer **L**, matrice d'historique **H**, nombre de mouvements effectués **Iter**, stratégie de perturbation **St**, type de mouvement **mvt**
Sortie: Une solution perturbée **π**
pour i allant de 1 à L **faire**
    construire l'ensemble (E) des mouvements candidats selon St
    sélectionner aléatoirement un mouvement x,y dans E
    π=π⊕**mvt**(x,y)
    C = C + delta**mvt**(π,x,y)
    Mettre à jour $H_{x,y}$
    Iter=Iter+1
    si C<C_{best} **alors**
        mettre à jour C_{best} et π_{best}
        ω = 0
        Desc = Desc x 1/2
    fin si
fin pour
retourner π

2.4.4Stratégie de diversification adaptative

En plus des deux scénarios de sélection définies dans 2.3.3, un troisième est introduit dans cette adaptation. Il privilégie les mouvements les moins récemment sollicités depuis le début de la recherche.

La stratégie de diversification dans le cas du TSP sélectionne les mouvements candidats selon ces trois scénarios de sélection.

A) **La perturbation dirigée**: Elle vise à construire un ensemble de candidats avec le minimum de dégradation possible durant la perturbation. Ces candidats ne doivent pas avoir été sollicités durant les γ derniers mouvements. Une exception est faite lorsque ce mouvement, alors tabou, mène à une solution qui améliore la meilleure solution trouvée jusqu'alors.
La perturbation dirigée est basée à la fois sur le principe de la liste tabou [72], [73] ainsi que la qualité des mouvements à appliquer. Les candidats éligibles durant cette perturbation sont définis par l'ensemble A suivant:

$$\begin{aligned} A = &\{2Opt(u,v) | min\{delta2Opt(\pi,u,v)\}, \\ &(H_{u,v}+\gamma) < Iter \vee (delta2Opt(\pi,u,v)+c) < c_{best}, u \neq v\} \end{aligned} \quad (2\ 2)$$

2Opt(x,y) étant l'opérateur de mouvement 2Opt, *delta2Opt(π,x,y)* la différence entre le fitness de π et le fitness du résultat de π perturbé par *2Opt(x,y)*. $H_{x,y}$ est la dernière itération dans laquelle *2Opt(x,y)* a été effectué, H est la matrice d'historique représentant tous les mouvements possibles.

B) **La perturbation basée sur la récence** construit l'ensemble des candidats en utilisant uniquement la matrice d'historique H. Les mouvements sont ceux ayant été le moins récemment (ou jamais) utilisés. L'ensemble est défini selon l'ensemble B suivant

$$B = \{2Opt(u,v) | min\{H_{u,v}\}, u \neq v\} \quad (2\ 3)$$

C) **La perturbation aléatoire** exécute des mouvements sélectionnés arbitrairement, aucune critère n'intervient dans le choix de ces mouvements.

$$C = \{2Opt(u,v), u \neq v\} \quad (2\ 4)$$

Quelque soit la perturbation appliquée, le mouvement choisi doit produire une solution faisable, sinon il est écarté.

Nous avons pris comme exemple le mouvement 2Opt dans les formules 2.2,2.3 et 2.4. Les mêmes s'appliquent aux mouvements d'insertion et d'échange.

L'algorithme 2.5 explique le fonctionnement de la perturbation adaptative. BLS sélectionne une de ces trois perturbations d'une façon pseudo-

aléatoire, en fonction de l'état de la recherche. Cet état est déterminé par le paramètre ω qui indique le nombre d'itérations consécutives n'ayant pas améliorées la meilleure solution trouvée (π_{best}). L'objectif est de donner la priorité (avec une plus grande probabilité) à la perturbation dirigée au début de la recherche, quand ω est assez petit, et réduire ses chances quand le voisinage courant s'avère trop attractif. Dans ce cas les autres perturbations seront préférées à la perturbation dirigée pour favoriser la diversification dans la recherche.

La probabilité (P) pour appliquer la perturbation dirigée est la même définie dans la formule (2.1), les perturbations basée sur la récence et aléatoire sont respectivement définies avec les probabilités $(1-P)\times Q$ et $(1-P)\times(1-Q)$ où Q est une probabilité constante.

Algorithme 2.5: perturbationAdaptative
Entrées: Optimum local **π**, nombre de sauts à effectuer **L**, indicateur du type de perturbation **Lw**, matrice d'historique **H**, nombre de mouvements effectués **Iter**, nombre de blocage consécutifs dans un voisinage **ω**
Sortie: Une solution perturbée **π**
Définir la probabilité *P* selon la formule (2.1)
Avec une probabilité P
    π = perturbationDynamique(π,L,Lw,H,Iter,A)
Avec une probabilité (1 – P) x Q
    π = perturbationDynamique(π,L,Lw,H,Iter,B)
Avec une probabilité (1 – P) x (1 – Q)
    π = perturbationDynamique(π,L,Lw,H,Iter,C)
retourner π

2.4.5Variation des sauts et des perturbations

Tant que le blocage persiste, BLS varie lors des perturbations le type de mouvements et/ou le nombre de sauts à effectuer. Ces variations sont effectuées dans l'ordre suivant: avec le même nombre de sauts L (i) le mouvement 2Opt puis (ii) l'insertion et finalement (iii) l'échange. Le nombre de sauts est incrémenté de 1 quand les trois perturbations précédentes ne conduisent pas vers un nouveau voisinage avant de les réexécuter dans le même ordre.

Dans le pire des cas, après T série de perturbations successives (ω=T) sans avoir changer de voisinage, la forte perturbation est lancée.

2.4.6Condition d'arrêt

La condition d'arrêt dans cette adaptation est définie par un nombre maximal de descentes ($Desc_{max}$) à effectuer. $Desc_{max}$ est défini en fonction de la taille de l'instance.

Le nombre de descentes courant ($Desc$) est réduit de de moitié si BLS améliore la meilleure solution trouvée (π_{best}), et d'un huitième lors de l'exécution de la perturbation forte. Ces réductions ont pour objectif de fournir plus d'itérations à la recherche qui, dans ces deux cas de figures, commence à explorer une nouvelle région de l'espace de recherche.

2.4.7L'algorithme de l'adaptation

Après avoir détaillé toutes les composantes de la métaheuristique BLS, l'algorithme 2.6 ci-dessous représente l'intégralité de l'algorithme et dont les résultats seront exposés dans la section suivante.

Algorithme 2.6: BLS pour le TSP

Entrées: Nombre maximal de descentes $\mathbf{Desc_{max}}$, nombre initial de sauts $\mathbf{L_0}$, nombre maximal de visites consecutives de l'optimum local sans amélioration **T**, nombre de sauts lors de la forte perturbation $\mathbf{L_{max}}$.

Sortie: Une solution $\boldsymbol{\pi}_{best}$

π = SolutionInitiale() /*Plus proche voisin*/
C = fitness(π)
Mettre à jour π_{best} et C_{best}
ω = 0
L = L_0
Desc = 0
Lw = 0 /*Indique le mouvement à exécuter dans la perturbation*/
C_p = C /*Fitness trouvé dans la dernière descente*/
Iter = 0 /*Nombre de mouvements effectués*/
tant que Desc<$Desc_{max}$ **faire**

tant que ∃2Opt(x,y)tq(C+delta2Opt(π,x,y)<C) **faire**
  π=π⊕2Opt(x,y)
  Mettre à jour $H_{x,y}$
  Iter=Iter+1
fin tant que
si f(π)<f(π_{best})**alors**
  Mettre à jour π_{best} et C_{best}
  ω = 0
  Desc = Desc x 1/2
sinon
  ω = ω+1
fin si
si (ω>T) **alors**
  π = perturbationForte(π,L_{max})
  ω = 0
  Desc = Desc x 7/8
sinon si (C==C_p) **alors**
  Lw = Lw+1
  L = L_0 + Lw/3
sinon
  Lw = 0
fin si
C_p = C
si la perturbation forte n'est pas exécutée **alors**
  π = perturbationAdaptative(π,L,Lw,H,Iter,ω)
fin si
fin tant que
retourner π_{best}

2.5 Résultats expérimentaux

2.5.1 Environnement

L'adaptation/implémentation de BLS pour le TSP est faite en langage Java (idem pour toutes les implémentations de cette thèse), elle est appliquée sur 41 instances du benchmark TSPLIB[58] dont les tailles varient de 14 à 442 villes.

Chacune de ces instances est exécutée 20 fois avec les paramètres donnés dans le tableau 2.1 ci-dessous. Le choix des valeurs des paramètres est fait après une pile de tests préliminaires discutés dans la section suivante.

Tableau 2.1: Valeurs des principaux paramètres de BLS

Paramètre	**Valeur**	**Explication**
$\mathbf{Desc_{max}}$	50n, 25n	Nombre maximale de descentes. 50N si n<200, sinon 25n
$\mathbf{L_0}$	1	Nombre initial de sauts
$\mathbf{L_{max}}$	n/2	Nombre de saut durant la forte perturbation
$\boldsymbol{\Upsilon}$	n	Durée de vie des tabous
$\mathbf{P_0}$	0,75	La plus petite probabilité pour faire la perturbation dirigée
Q	0,7	Probabilité de la perturbation basée sur la récence par rapport à l'aléatoire
T	(Desc-1) $\times(1-\frac{7}{8})$	Limite du nombre consécutif de blocages au minimum local

Il a été observé (dans le pire des cas) que BLS ne puisse jamais atteindre les conditions d'arrêts si T est prédéfini à une valeur assez petite. Ce blocage est causé par le réduction répétitive du nombre courant des descentes ($Desc$) qui empêche d'atteindre $Desc_{max}$. T doit être supérieur à (Desc-1) $\times(1-\frac{7}{8})$. T est donc égale à $\mathrm{Desc}_{max} - 1_{\overline{8+1}}$.

2.5.2Justification des valeurs des paramètres

La qualité des résultats obtenus par BLS est due en partie au choix des paramètres (tableau 2.1), chacun est justifié par son rôle et son influence sur BLS. Les plus influents d'entre eux seront validés par des tests comparatifs sur 3 instances de la TSPLIB (eil51, eil76 et eil101). Les résultats sont représentés par deux graphes superposés. (i) Le premier est un diagramme à barres empilées,

chaque pile donne la meilleure solution, la moyenne et la pire. Il affiche seulement deux barres quand la meilleure solution trouvé est égale à la meilleure solution connu (best known solution (*bks*)). (ii) Le second est un graphe linéaire montrant l'évolution du temps d'exécution de BLS.

Les premiers paramètres présentés ci-dessous sont en fonction de la taille de l'instance (N).

Le nombre maximal de descentes ($Desc_{max}$)

$Desc_{max}$ décide combien de fois la recherche locale doit être effectuée, plus le nombre de descentes est grand, meilleurs seront les résultats: Chaque descente est une nouvelle occasion pour trouvé un optimum local meilleur ou pour échapper à l'attraction d'un bassin. En contrepartie, le temps d'exécution sera plus conséquent comme il est montré dans la figure 2.3. Il a été noté durant les tests qu'après un certain nombre de descentes, les résultats deviennent assez satisfaisants; les descentes qui suivent augmentent le temps d'exécution sans pour autant apporter une amélioration notable à la qualité de la solution.

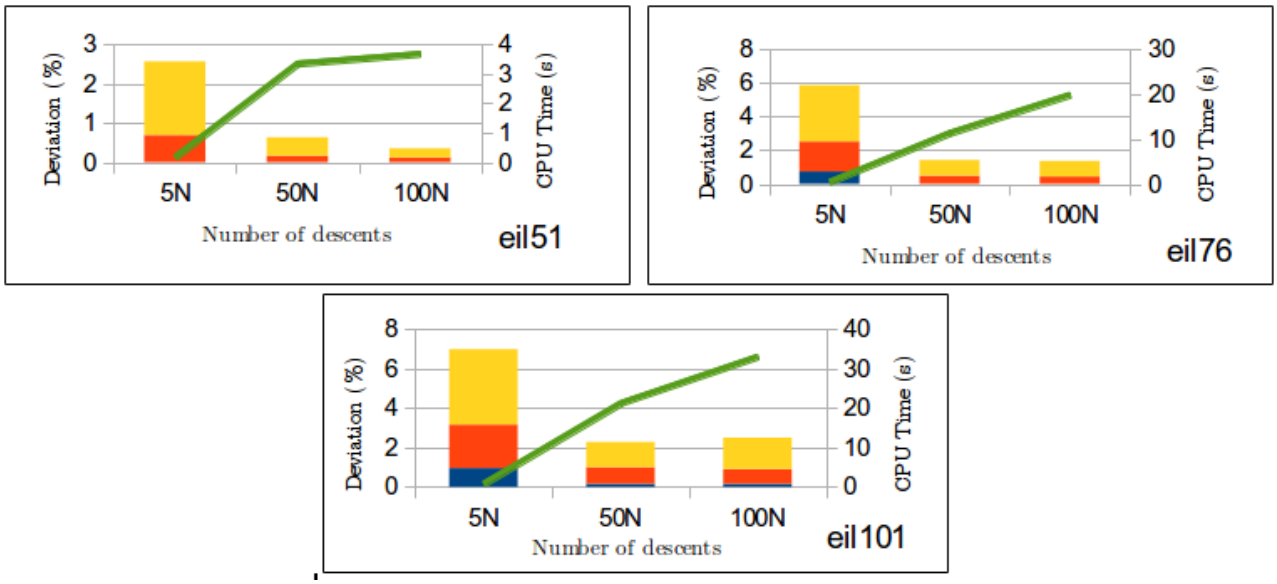


Figure 2.3:Définition du nombre maximale de descentes pour les instances *eil*

Nombre initial de mouvements/sauts (L_0)

Ce nombre indique le minimum de mouvements à effectuer pour chacun des trois types de perturbations. Il a un impact direct sur le changement du voisinage et donc sur les chances de déblocage. Plus grand il est, plus il est important le changement dans l'espace de recherche, ce qui peut rendre parfois la descente qui suit totalement indépendante de ce qui a précédé et donner l'impression d'un nouveau démarrage.

Les meilleurs résultats sont obtenus comme il est montré dans la figure 2.4 avec un démarrage à un seul saut, offrant une grande efficacité aux perturbations.

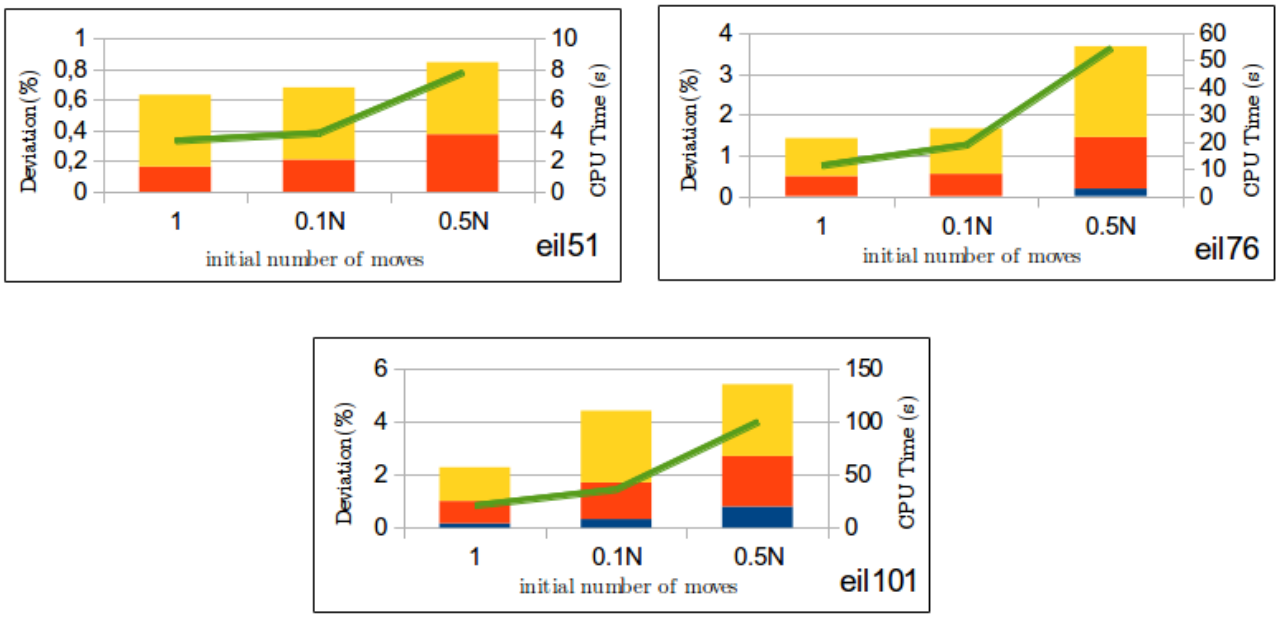


Figure 2.4: Définition du nombre initial de sauts pour les instances *eil*

Nombre maximal d'échec consécutif à l'échappée du voisinage (*T*)

Cette variable indique à quel moment la forte perturbation doit être exécutée. Plus T est petit, plus grand est le nombre de fortes perturbations, ce qui augmente les chances d'échapper à l'attraction du bassin courant. Comme expliqué auparavant, T doit être plus grand que $\text{Desc}_{max} - 1\frac{}{8+1}$ pour garantir l'atteinte du critère d'arrêt. Les tests ci-dessous affichés sur la figure 2.5 sont effectués en variant

T trois fois: (i) En choisissant d'abord la valeur minimale pour chaque instance, (ii) puis en doublant cette valeur et (iii) en la triplant.

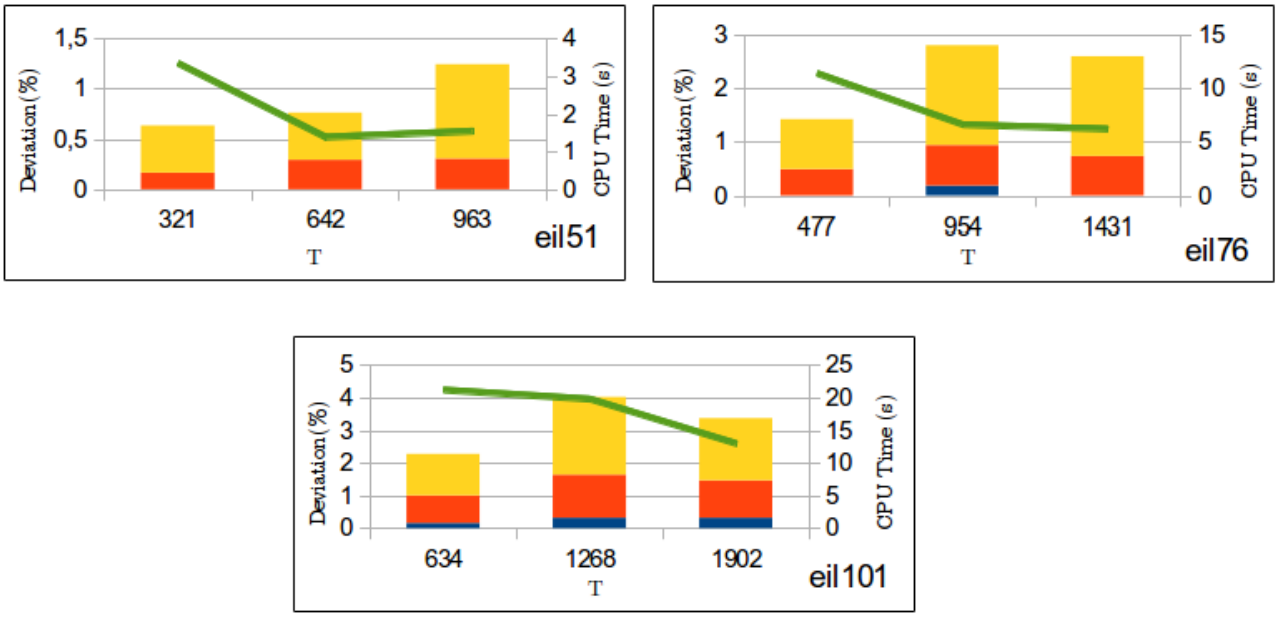


Figure 2.5: Définition du nombre maximal d'échec consécutif à l'échappée du voisinage pour les instances *eil*

2.5.3Performances et étude comparative

Pour chaque instance, l'analyse des performances de BLS est basée sur quatre mesures:

1. δ: L'écart entre la solution obtenue et la meilleure solution connue

$$\delta = 100 \times (\bar{c} - bks)/bks[\%] \qquad (2\ 5)$$

   où $\bar{c}$ est la moyenne des fitness obtenus des 20 exécutions, et bks est la valeur de la meilleure solution connue disponible sur la libraire TSPLIB[58];

2. $C_{1\%}$: Le nombre de solutions dont l'écart est en dessous de 1% (sur les 20 exécutions);

3. C_{opt}: Le nombre de solutions dont le fitness est égal à bks;

4. Temps (CPU) d'exécution en secondes.

Les performances de BLS sont comparés avec ceux de (i) la recherche locale standard en utilisant le voisinage 2Opt[115] (LS 2Opt) tirés depuis la littérature, ainsi que (ii) l'algorithme ILS en utilisant le voisinage 2Opt dans la phase de la recherche locale et le mouvement Double Bridge dans la perturbation. La comparaison reportée dans le tableau 2.2 est basée sur la recherche locale et perturbation(s) utilisés dans cette adaptation de BLS.

Tableau 2.2: Résultats comparatifs entre BLS, la recherche locale 2Opt et ILS

Instance	**taille**	**bks**	**LS 2Opt**	**ILS**		**BLS**		
				δ	$C_{1\%}/C_{opt}$	δ	$C_{1\%}/C_{opt}$	**Temps CPU (sec)**
burma14	14	3323	--	**0**		**0**		0
ulysses16	16	6859	--	**0**		**0**		0
ulysses22	16	7013	--	**0**		**0**		0
eil51	51	426	--	0.469	14/5	0.199	20/8	0.58
berlin52	52	7542	--	0.106	19/19	**0**		0
st70	70	675	--	0.741	13/3	**0**		0.97
eil76	76	538	--	0.929	11/3	0.492	18/5	2.67
pr76	76	108159	--	0.518	20/7	**0**		1.41
gr96	96	55209	0.997	0.466	18/4	0.215	20/5	6
rat99	99	1211	0.614	1.652	7/0	0.057	20/12	5.72
kroA100	100	21282	0.073	0.282	20/8	**0**		4.02
kroB100	100	22141	0.379	0.632	14/5	0.012	20/18	6.58
kroC100	100	20749	0.546	0.882	13/2	**0**		4.38
kroD100	100	21294	1.538	0.977	13/0	0.053	20/14	5.16
kroE100	100	22068	0.983	0.77	15/1	0.164	20/5	5.89
rd100	100	7910	0.961	0.619	14/3	0.01	20/16	10
eil101	101	629	1.657	1.749	3/0	1.017	11/0	6.07
lin105	105	14379	0.642	0.285	19/7	**0**		2.1

Instance	**taille**	**bks**	**LS 2Opt**	**ILS**		**BLS**		
				δ	$C_{1\%}/C_{opt}$	δ	$C_{1\%}/C_{opt}$	**Temps CPU (sec)**
pr107	107	44303	0.093	0.95	9/0	0.042	20/11	5.89
pr124	124	59030	0.953	0.281	19/7	**0**		7.91
bier127	127	118282	0.649	0.686	15/1	0.139	20/4	31
ch130	130	6110	0.999	1.244	7/0	0.324	20/2	14
pr136	136	96772	--	1.775	7/0	0.675	20/0	15
gr137	137	69853	0.824	1.266	10/0	0.347	20/0	20
pr144	144	58537	--	0.091	20/7	**0**		4.39
ch150	150	6528	--	1.241	8/0	0.412	20/2	18
kroA150	150	26524	--	1.421	6/0	0.374	20/0	26
kroB150	150	26130	--	1.309	8/0	0.237	20/1	22
pr152	152	73682	--	0.415	19/1	0.03	20/8	15
u159	159	42080	--	1.694	5/1	**0**		25
rat195	195	2323	--	2.927	0/0	1.157	1/0	40
d198	198	15780	--	0.754	16/0	0.581	20/0	46
gr202	202	40160	--	1.805	0/0	1.314	5/0	35
ts225	225	126643	--	0.706	16/15	0.11	20/10	50
gr229	229	134602	0.911	1.306	6/0	1.403	2/0	49
gil262	262	2378	1.099	2.397	0/0	1.345	6/0	70
a280	280	2579	--	3.373	1/0	0.866	12/0	68
lin318	318	42029	1.202	2.253	0/0	1.384	2/0	152
rd400	400	15281	1.543	3.03	0/0	2.493	0/0	280
gr431	431	171414	3.045	2.357	0/0	2.243	0/0	294
pcb442	442	50778	5.185	3.096	0/0	2.315	0/0	205
moyenne				1.157		0.488		

Les résultats ci-dessus montrent la bonne contribution de BLS par rapport à la recherche locale standard et au framework ILS dans sa version basique. 38 des 41 instances ont un écart inférieur à 1%, au moins une fois sur les 20 exécutions, 30 instances le sont pour toutes les exécutions. 20 instances atteignent au moins une fois la le meilleur fitness connu, dont 12 l'obtiennent toujours.

2.5.4Le mouvement 3Opt en phase de recherche locale

BLS peut être amélioré en optant pour une fonction de voisinage plus large dans la recherche local comme le mouvement 3Opt, le nombre d'arêtes affectés à chaque mouvement est plus grand dans ce cas. Ce changement aura un impact sur les performances et le comportement de BLS: l'historique des mouvement est plus large/diversifié que celui construit par le mouvement 2Opt, ce qui permet d'avoir des perturbations plus diversifiés.

Le tableau 2.3 ci-dessous montre les résultats de BLS en appliquant le mouvement 3Opt (BLS 3Opt), tout en le comparant avec la recherche locale standard en utilisant 3Opt comme fonction de voisinage (LS 3Opt)[115], et aussi avec la version de BLS vue précédemment (BLS 2Opt).

Tableau 2.2: Résultats comparatifs entre BLS et la recherche locale 3Opt

Instance	**Taille**	**bks**	**LS 3Opt**	**BLS 2Opt**			**BLS 3Opt**		
				δ	$C_{1\%}/C_{opt}$	**Temps CPU (sec)**	δ	$C_{1\%}/C_{opt}$	**Temps CPU (sec)**
burma14	14	3323	--	**0**		0	**0**		0
ulysses16	16	6859	--	**0**		0	**0**		0
ulysses22	16	7013	--	**0**		0	**0**		0
eil51	51	426	--	0.199	20/8	0.58	**0**		2.69
berlin52	52	7542	--	**0**		0	**0**		0.47
st70	70	675	--	**0**		0.97	**0**		11
eil76	76	538	--	0.492	18/5	2.67	**0**		8.55

Instance	**Taille**	**bks**	**LS 3Opt**	**BLS 2Opt**			**BLS 3Opt**		
				δ	$C_{1\%}/C_{opt}$	**Temps CPU (sec)**	δ	$C_{1\%}/C_{opt}$	**Temps CPU (sec)**
pr76	76	108159	--	**0**		1.41	**0**		21
gr96	96	55209	0.997	0.215	20/5	6	**0**		23
rat99	99	1211	0.614	0.057	20/12	5.72	0.033	20/12	16
kroA100	100	21282	0.073	**0**		4.02	**0**		18
kroB100	100	22141	0.379	0.012	20/18	6.58	**0**		62
kroC100	100	20749	0.546	**0**		4.38	**0**		25
kroD100	100	21294	1.538	0.053	20/14	5.16	**0**		35
kroE100	100	22068	0.983	0.164	20/5	5.89	138	20/10	52
rd100	100	7910	0.961	0.01	20/16	10	**0**		54
eil101	101	629	1.657	1.017	11/0	6.07	0.063	20/12	39
lin105	105	14379	0.642	**0**		2.1	**0**		29
pr107	107	44303	0.093	0.042	20/11	5.89	**0**		36
pr124	124	59030	0.953	**0**		7.91	**0**		21
bier127	127	118282	0.649	0.139	20/4	31	0.102	20/9	247
ch130	130	6110	0.999	0.324	20/2	14	0.216	20/7	196
pr136	136	96772	--	0.675	20/0	15	0.374	20/0	244
pr144	144	58537	--	**0**		4.39	**0**		12
ch150	150	6528	--	0.412	20/2	18	0.117	20/14	284
kroA150	150	26524	--	0.374	20/0	26	0.162	20/0	400
kroB150	150	26130	--	0.237	20/1	22	0.185	18/0	626
pr152	152	73682	--	0.03	20/8	15	0.041	20/6	374
u159	159	42080	--	**0**		25	**0**		287
rat195	195	2323	--	1.157	1/0	40	0.499	20/0	624

Le voisinage 3Opt apporte beaucoup d'améliorations à BLS. Toutes les instances testées ne dépassent pas 1% d'écart avec *bks*, la plus mauvaise moyenne est inférieure à 0.5%. Seules 4 instances des 30 n'ont pu atteindre *bks*, tandis que 21 l'obtiennent toujours.

2.6 Discussion des résultats

En s'inspirant du framework ILS, BLS utilise la recherche locale pour exploiter une région et avoir son optimum, puis des perturbations pour échapper à cette région et continuer à explorer l'espace de recherche. BLS se distingue cependant de ILS par sa combinaison de stratégies de perturbations diversifiée de différentes puissances, dont chacune est exécutée selon l'état de la recherche. Comme expliqué précédemment, BLS utilise des perturbations de faible diversification avec une grande probabilité P tant que la recherche mène vers de nouveaux optima locaux meilleurs que les précédents.

En négligeant le critère d'arrêt dans cette adaptation, BLS réussit toujours à trouver *bks*. En effet, l'espace de recherche couvert par BLS s'étend après chaque itération jusqu'à tomber sur le voisinage où se trouve *bks*. Le tableau 2.4 ci-dessous donne le meilleur et la moyenne des temps d'exécution (en secondes) pour trouver *bks* pour quelques instances quand BLS est exécuté sans critères d'arrêts.

Tableau 2.3: Temps d'exécution requis par BLS pour trouver *bks*

Instance	**Taille**	**bks**	**Meilleur temps**	**Temps moyen**
eil51	51	426	0	2.2
berlin52	52	7542	0	0.3
st70	70	675	0.6	13
eil76	76	538	1.2	9.4
pr76	76	108159	0.3	6.9
rat99	99	1211	2.1	960
kroA100	100	21282	0.5	35

Instance	**Taille**	**bks**	**Meilleur temps**	**Temps moyen**
kroB100	100	22141	8.1	95
kroC100	100	20749	1.4	20
kroD100	100	21294	24	433
kroE100	100	22068	1.19	47
rd100	100	7910	1.5	55
eil101	101	629	2	257
lin105	105	14379	0.6	41
pr107	107	44303	0.6	78
pr124	124	59030	1.3	27

2.7 Conclusion

Ce chapitre présente une adaptation de la métaheuristique *Breakout Local Search* (BLS) pour résoudre le TSP. Elle est basée sur le framework de la recherche locale itérée (ILS) et améliore ses perturbations avec une stratégie de diversification adaptative afin de réussir le passage d'un bassin d'attraction vers un nouveau. BLS utilise la forte descente dans la recherche locale avec le 2Opt comme fonction de voisinage. Pour visiter un optimum local de fitness meilleur, les sauts (en nombre et en nature) sont contrôlés d'une manière adaptative selon l'état de la recherche. Chaque perturbation est effectuée en variant le type de mouvement puis le nombre de sauts, et en sélectionnant les mouvements les plus appropriés après chaque descente.

La qualité des résultats de BLS dans cette adaptation sont assez satisfaisants par rapport aux algorithmes cités dans la comparaison. La construction de l'ensemble des candidats à chaque saut (formules 2.2, 2.3, 2.4) rend BLS trop lent par rapport à ses concurrents. Pour un bon compromis entre la qualité des résultats et le temps d'exécution, le voisinage *2Opt* a été préféré au *3Opt* dans la

phase de la recherche locale , le deuxième étant plus lent que le premier comme il a été présenté dans le tableau comparatif 2.3.

D'autres stratégies de sélection/construction des candidats à la perturbation peuvent être envisagées afin de remédier à cette lenteur. Les résultats de l'adaptation ont montré aussi quelques lacunes sur certaines instances difficiles du benchmark dont certains bassins sont trop attractifs et donc deviennent difficiles à y échapper. BLS peut être renforcé en introduisant des méthodes à populations pour mieux explorer l'espace de recherche et augmenter les chances de tomber sur des optima locaux meilleurs, voir même la *bks*.

Chapitre 3: Une approche mémétique basée sur BLS pour le Problème du Voyageur de Commerce Généralisé

3.1 Introduction

Les améliorations proposées à la fin du chapitre précédent seront abordées dans celui-ci, tout en s'attaquant à un autre problème, celui du Voyageur de Commerce Généralisé (GTSP): une variante du TSP plus difficile (cf. chapitre 1).

Ce chapitre traite donc un algorithme mémétique basé sur BLS amélioré selon les suggestions présentés précédemment: proposer une nouvelle méthode de sélection des candidats à la perturbation plus rapide que le parcours exhaustif de tous les mouvements possibles.

3.2 Les Algorithmes Mémétiques

Les algorithmes génétiques (AG) simples (cf. chapitre 1) se sont avérés inefficaces pour trouver des solutions quasi optimales. Quoiqu'ils proposent une diversification élargie (selon les paramètres choisis) dans l'espace de recherche, ces algorithmes manquent de stratégie d'amélioration des individus sélectionnés. Plusieurs chercheurs ont donc proposé l'introduction de la recherche locale dans les AG tels que l'ensemble des individus de la population deviennent des optima locaux, donnant ainsi des solutions meilleurs et donc des algorithmes plus efficaces, connus sous le nom d'algorithmes mémétiques (AM) [88], [116].

Les AM combinent donc entre le point fort des AG, celui de l'exploration, et la recherche locale caractérisée par l'exploitation d'un sous espace de recherche. Les AM peuvent être présentés en cinq procédures ou étapes:

1. **Initialisation**: Construire une population initiale de taille P en utilisant des heuristiques simples et rapides.

2. **Amélioration**: Appliquer la recherche locale aux P solutions pour obtenir une population d'optima locaux. De toutes les solutions similaires on ne gardera qu'une seule. On obtient dans cette première génération une population de taille $p \leq P$.
3. **Production d'une nouvelle génération**: Produire une nouvelle génération d'individus de taille P' en utilisant les opérateurs génétiques de sélection, croisement et mutation.
4. **Amélioration de la nouvelle génération**: Utiliser la recherche locale pour remplacer les P' individus par P' optima locaux. Là encore, de toutes les solutions similaires on ne gardera qu'une seule. On obtient une génération d'individus de taille $p' \leq P'$.
5. **Évolution**: Répéter les étapes 3 et 4 jusqu'à l'atteinte des conditions d'arrêt.

Les AM ont été appliqués sur différents problèmes d'optimisation combinatoires tel que le problème d'affectation quadratique[117], le problème de programmation quadratique binaire[118], le partitionnement des graphes[119], le problème du voyageur de commerce (TSP)[120]–[125], et le GTSP [126]–[128].

3.3 La recherche locale adaptée au GTSP

Deux approches de résolution du GTSP ont été proposées par Hu et Raidl [24], elles se basent sur la répartition du problème en deux niveaux/couches. La première approche est basée sur les clusters, la deuxième est basée sur les nœuds (cf. section 1.2.3.1). Des méthodes de recherche locale sont implémentées à partir de ces deux approches.

3.3.1 La recherche locale basée sur les clusters

Dans l'approche basée sur les clusters, la définition d'une permutation de clusters se fait dans la couche supérieure, la sélection des nœuds se fait dans la couche inférieure.

La recherche locale basée sur les clusters (*Cluster-Based local search (CBLS)*) cherche une solution voisine à la solution courante $s = (\pi, \tau)$ par le voisinage 2Opt appliqué à la permutation de clusters $\tau = \{\tau_1, ..., \tau_m\}$ comme il est défini dans la formule 3.1, appelé aussi 2Opt généralisé. Pour chaque permutation τ' voisine à τ, la couche inférieure cherche l'ensemble optimale de nœuds couvrants. Hu et Raidl [24] ont utilisé pour cette couche une méthode basée sur le calcul bidirectionnel et incrémentiel pour trouver chemin le plus court (*incremental bidirectional shortest path calculation*), Karapetyan et Gutin [129] ont par la suite proposé une amélioration de l'algorithme de Fischetti et al. [54] basé sur la programmation dynamique. L'algorithme 3.1 ci-dessous décrit le processus CBLS, la méthode noeudsCouvrants(τ) désigne la recherche de l'ensemble optimale de nœuds couvrants de la couche inférieure à partir de la permutation de clusters τ.

$$N_a(\tau) = \{\tau' | \tau' = (\tau_1, ..., \tau_{i-1}, \tau_j, \tau_{j-1}, ..., \tau_i, \tau_{j+1}, ..., \tau_m), 1 \leq i \leq j \leq m\} \quad (3\ 1)$$

Algorithme 3.1: Recherche locale basée sur les clusters (CBLS)

Entrée: solution initiale **s = (π,τ)**
pour chaque τ' ∈ N_a(τ) **faire**
    π' = noeudsCouvrants(τ')
    si (f(s')<f(s)) **alors** /* s' = (π',τ') */
        s = s'
        recommencer CBLS
    fin si
fin pour
retourner s = (π,τ)

3.3.2La recherche locale basée sur les nœuds

Dans l'approche basée sur les nœuds, la sélection des nœuds couvrants se fait dans la couche supérieure, l'algorithme de la couche inférieure trouve le meilleure tour à partir de ces nœuds sélectionnés.

Dans la recherche locale basée sur les nœuds (Node-Based local search), l'algorithme de la couche supérieure effectue une recherche locale par le voisinage d'échange de nœuds défini dans la formule 3.2 par le remplacement d'un nœud couvrant π_i par un autre nœud appartenant au même cluster de π_i. La résolution de la couche inférieure (trouver le meilleure tour parcourant tous les nœuds couvrants) revient à résoudre une instance du TSP, une heuristique ou une méthode exacte, si le nombre de clusters le permets, peut être appliquée aux nœuds couvrants (π) pour trouver une bonne permutation. Le 2Opt généralisé peut être aussi appliqué dans la couche inférieure pour améliorer la permutation de clusters (τ). Les algorithmes 3.2 et 3.3 présentent la recherche locale basée sur le voisinage par échange de nœuds (*Node Exchange Neighborhood Local Search (NENLS)*). Le premier (NENLS1) utilise la résolution de la couche inférieure par une méthode d'optimisation (TSPopt()) sur les nœuds couvrants sélectionnés, le deuxième (NENLS2) utilise la recherche locale 2Opt généralisé (G2Opt()) sur la permutation des clusters obtenue à partir de l'ordre des nœuds couvrants déjà établi.

$$N_b(\pi) = \{\pi' | \pi' = \{\pi_1, ..., \pi_{i-1}, \pi'_i, \pi_{i+1}, ..., \pi_m\}, \pi'_i \in C_i \setminus \{\pi_i\}, 1 \leq i \leq m\} \quad (3\ 2)$$

Algorithme 3.2: Recherche locale basée sur les nœuds (NENLS1)

Entrée: solution initiale **s = (π,τ)**

pour chaque π' ∈ N_b(π) **faire**
    π' = TSPopt(π')
    construire τ' tq s' = (π',τ')
    si (f(s')<f(s)) **alors**
        s = s'
        recommencer NENLS1
    fin si
fin pour
retourner s = (π,τ)

Algorithme 3.3: Recherche locale basée sur les nœuds (NENLS2)
Entrée: solution initiale **s = (π,τ)**
pour chaque π' ∈ N_b(π) **faire**
    construire τ' tq s' = (π',τ')
    τ' = G2Opt(τ')
    si (f(s')<f(s)) **alors**
        s = s'
        recommencer NENLS2
    fin si
fin pour
retourner s = (π,τ)

3.3.3La recherche par voisinage variable

Hu et Raidl [24] ont introduit aussi une approche hybride des deux recherches locales définies précédemment, appelée "la recherche par voisinage variable" (*Variable Neighbourhood Search (VNS)*). Elle alterne entre CBLS et NENLS2, cette dernière est sollicitée quand la première atteint l'optimalité.

Deux voisinages sont établis à partir du voisinage 2Opt généralisé (N_a) et le voisinage par échange de nœuds (N_b), ils sont définis dans les formules 3.3 et 3.4 et utilisés en alternance comme le montre l'algorithme 3.4 ci-dessous.

$$\mathrm{N}_1(s) = \{s' = (\pi', \tau') \mid \tau' \in N_a(\tau),\ \pi' = noeudsCouvrants(\tau')\} \quad (3\ 3)$$

$$\mathrm{N}_2(S) = \{s' = (\pi', \tau') \mid \pi' \in N_b(\pi)\ ,\ \tau' = G2Opt(\tau')\} \quad (3\ 4)$$

Algorithme 3.4: Recherche par voisinage variable (VNS)
Entrées: solution initiale **s = (π,τ)**, voisinages **N_k ={1,2}**
k = 1
tant que $k \leq 2$ **faire**
    s_p = s
    pour chaque $s' \in N_1(s)$ **faire**

```
            si f(s')<f(s) alors
                s = s'
                k = 1
            fin si
        fin pour
        si (s_p == s) alors
            k = k+1
        fin si
    fin tant que
    retourner s
```

3.4 Breakout Local Search pour le GTSP

3.4.1 Rappel de l'idée de base

BLS est, comme décrit dans le chapitre précédent, une métaheuristique basée sur le framework de la recherche locale itérée (*Iterative Local Search (ILS)*). Elle apporte une contribution majeure dans la phase de perturbation en introduisant des perturbations adaptatives dont les mouvements candidats sont élus selon l'état de la recherche: Les mouvements causant le moins de dégradations et donnant la meilleure amélioration sont privilégiés quand BLS avance dans l'espace de recherche sans encombre. Tandis que les mouvements qui conduisent à des perturbations importantes - sans pour autant garantir une amélioration de la solution courante - sont davantage priorisés quand la recherche peine à quitter le bassin d'attraction courant.

La perturbation dans BLS est exécutée sous forme d'une série de sauts $L_0 \leq L \leq L_0 + T$. La recherche commence avec un petit nombre de sauts L_0 et l'incrémente au fur et à mesure du blocage dans le voisinage courant. Quand la recherche reste bloquée dans le même bassin après T descentes, une perturbation forte est déclenchée avec un nombre de saut L_{max} qui dépassent largement le nombre de sauts précédemment effectués.

À chaque saut, un nouvel ensemble de mouvements candidats à la perturbation est construit, étant donné que la matrice H d'historique a été altérée dans le saut précédent. Cette construction de l'ensemble des mouvements est caractérisée par une complexité carrée, puisque toute la matrice H est parcourue afin de construire les ensembles A et B définis dans les équations 2.2 et 2.3.

La contribution de BLS dans la phase de perturbation a nettement amélioré les résultats du framework ILS. La complexité des perturbations rend BLS très lent, ce qui représente un inconvénient majeur puisque ceci empêche la résolution des instances larges et/ou difficiles.

3.4.2Nouvelle contribution pour BLS

Dans cette nouvelle adaptation, une nouvelle contribution est apportée à BLS afin de réduire sa complexité, tout en essayant de ne pas endommager sa stratégie globale de perturbation afin de garder la qualité des résultats.

Au lieu de parcourir l'intégralité de la matrice H, cette version mémétique de BLS pour le GTSP sélectionne aléatoirement M mouvements et y applique les équations 2.2 et 2.3, la matrice de d'historique H de taille N^2 est réduite alors à un échantillon S de taille M. Ce changement réduit la complexité de chaque saut de la perturbation, en passant d'une complexité carrée ($O(m^2)$) à linéaire ($O(m)$) ce qui va baisser considérablement le temps d'exécution de BLS. Les équations 2.2 et 2.3 deviennent alors respectivement 3.5 et 3.6.

$$\begin{aligned} A =&\{swap(u,v)|min\{deltaSwap(\pi,u,v)\},\\ &(S_{u,v}+\gamma) < Iter \vee (deltaSwap(\pi,u,v)+c) < c_{best}, u \neq v\} \end{aligned} \tag{3 5}$$

$$\text{B=\{ swap(u, v) — min } \{S_{u,v}\}, u \neq v\} \tag{3 6}$$

$$C = \{swap(u,v), u \neq v\} \tag{3 7}$$

3.5 Algorithme mémétique basé sur BLS

Outre la réduction de la complexité, une hybridation avec les AG est effectuée dans ce travail afin d'obtenir un algorithme mémétique bénéficiant des

avantages de BLS et ceux des méthodes basées sur une population, évoluant selon les étapes définies précédemment dans 3.2.

3.5.1Le framework proposé

1. **Initialisation**: M/2 solutions sont générées pour former une population, M étant le nombre de clusters. Chaque solution est générée par la méthode de construction pseudo-aléatoire: Elle consiste en la génération aléatoire d'une permutation de clusters et sélectionner le meilleur nœud de chaque cluster. Ceci peut être effectué en utilisant l'algorithme *Cluster Optimization* (CO) de Fischetti et. Al [54] et amélioré par Karapetyan et Gutin [129]. CO utilise l'algorithme *s-t path* qui retourne le plus court chemin dans un graphe acyclique [130].
2. **Amélioration**: Effectuée avec une ou plusieurs méthodes de recherche locale. Dans le cas de cet AM, chaque solution de la population est améliorée par BLS pour obtenir une population avec des individus de bonne qualité.
3. **Production d'une nouvelle génération**: Les nouveaux individus de la population sont générés avec les opérateurs de croisement et mutation.
    1. *Croisement*: Le croisement uniforme [131] est utilisé dans cette adaptation, il produit deux enfants à partir de deux solutions (S1 et S2) sélectionnées de la population en utilisant la sélection par tournoi [132]. Chaque solution S_i est sélectionnée parmi trois solutions choisies aléatoirement.
    2. *Mutation*: Elle est appliquée sur une solution de la population tirée aléatoirement. Cette approche mémétique utilise le mouvement Double Bridge[114] (cf. Section 2.4.2) en tant qu'opérateur de mutation afin de réduire les chances de revenir à la même solution après l'étape de l'amélioration.

    Conditions d'arrêt: La population cesse d'évoluer une fois la meilleure solution connue (*bks*) pour l'instance **I** est trouvé. Sinon, Quand le nombre de

générations produites est égale au nombres de clusters (m) de $\mathbf{I}$. BLS retournera la meilleure solution de la dernière génération.

L'algorithme 3.5 représente succinctement la version mémétique basée sur la métaheuristique BLS dont les principaux modules ont été décrits ci-dessus.

Algorithme 3.5: Framework mémétique basée sur BLS

```
Entrée: meilleure solution connue bks,
S[M/2]                          /*population d'individus*/
p[2]                            /*parents*/
o[2]                            /*fils (offsprings)*/
pour i allant de 1 jusqu'à M/2 faire
        S[i] = SolutionInitiale()   /* pseudo aléatoire */
fin pour
pour i allant de 1 jusqu'à M/2 faire
        S[i] = BLS()                /* algorithme 3.6 */
fin pour
mettre à jour s_best
tant que (f(s_best)>bks ou k<M) faire
        p = SelectionTournoi(S)
        o = CroisementUniforme(p[1],p[2])
        i = aleatoire[1 , M/2]
        S[i] = DoubleBridgeMove()
        o[1] = BLS()
        o[2] = BLS()
        mettre à jour S
        mettre à jour s_best
        k = k+1
fin tant que
retourner s_best
```

3.5.2Nouvelle configuration de BLS

Mis à part la nouvelle contribution présentée précédemment, cette nouvelle version BLS doit être adaptée non seulement avec le GTSP, mais aussi avec le framework mémétique.

BLS sollicite dans la phase de descente la recherche locale basée sur les clusters (CBLS). Elle utilise le 2Opt généralisé dans la couche supérieure puis l'algorithme CO [54], [129] pour trouver le meilleure ensemble de nœuds couvrants dans la couche inférieure. L'historique de la recherche (H) conservera donc les mouvements des clusters

Coté perturbations, un seul type de mouvement est pris en considération, celui du *swap*. Idem pour la perturbation forte appliquée avec le même mouvement mais avec L_{max} sauts moyennant des mouvements aléatoires. Toutes les perturbations sont appliquées sur les permutations des clusters (τ).

Le nombre maximal de descentes $Desc_{max}$, critère d'arrêt de BLS, est cette fois-ci plus petit que lors de la précédente adaptation pour le TSP, il est réduit à 50 itérations (contre 50n appliquées au TSP). BLS étant appliquée sur une population, le temps d'exécution sera naturellement multiplié par le nombre d'individus. Il est donc impératif de réduire le nombre de descentes afin d'obtenir des résultats dans un temps raisonnable. L'algorithme 3.6 ci-dessous expose l'approche BLS sollicitée pour améliorer les individus de la population dans l'approche mémétique (algorithme 3.5).

Algorithme 3.6 Breakout Local Search (BLS)

Entrées: Nombre maximal de descentes $\mathbf{Desc_{max}}$, nombre initial de sauts $\mathbf{L_0}$, nombre maximal de visites consecutives de l'optimum local sans amélioration **T**, nombre de sauts lors de la forte perturbation $\mathbf{L_{max}}$, solution initiale $s = (\pi,\tau)$

$s_{best} = s$
$\omega = 0$
$L = L_0$
$s_p = s$
$Desc = 0$
$Iter = 0$
tant que ($Desc \langle Desc_{max}$) **faire**
    CBLS:

pour chaque $\tau' \in N_a(\tau)$ **faire**
    π' = clusterOptimization(τ')
    si (f(s')<f(s)) **alors**
        s = s'
        Mettre à jour H
        Iter = Iter+1
        aller à CBLS
    fin si
fin pour
si (f(s)<f(s_{best})) **alors**
    s_{best} = s
    $\omega = 0$
sinon
    $\omega = \omega+1$
fin si
si ($\omega \rangle$ T) **alors**
    s $\leftarrow$ PerturbationForte(s, L_{max}, H, Iter, ω)
    $\omega \leftarrow 0$
sinon si (s == s_p) **alors**
    L $\leftarrow$ L+1
sinon
    L $\leftarrow L_0$
fin si
$s_p \leftarrow$ s
si la perturbation forte n'est pas exécutée **alors**
    s $\leftarrow$ PerturbationAdaptative(s, L, H, Iter, ω)
fin si
fin tant que
retourner π_{best}

Algorithme 3.7: perturbationAdaptative
Entrées: Optimum local **s**, nombre de sauts à effectuer **L**, matrice d'historique **H**, nombre de mouvements effectués **Iter**, nombre de blocage consécutifs dans un voisinage **ω**
Sortie: Une solution perturbée **s**
Définir la probabilité *P* selon la formule (2.1)
Avec une probabilité P
    s = perturbation(s,L,H,Iter,A) /*perturbation dirigée*/
Avec une probabilité (1 – P) x Q //perturbation basée
    s = perturbation(s,L,H,Iter,B) //sur la récence
Avec une probabilité (1 – P) x (1 – Q)
    s = perturbation(s,L,H,Iter,C) /*perturbation aléatoire*/
retourner s

3.6 Résultats expérimentaux

Les instances sur lesquels ont été appliquées cette approche mémétique sont des instances de la TSPLIB, convertis en instances GTSP par la procédure de clusterisation de Fischetti et al. [54]. 39 dont la taille va de 10 clusters et 48 villes jusqu'à 89 clusters et 442 villes sont résolus, chacune est exécutée 20 fois.

L'algorithme mémétique basée sur BLS est comparé durant les tests avec deux approches évolutionnaires: un AM présenté par Bontoux et al. [127] basé sur un opérateur de croisement utilisant la recherche dans de larges voisinages. Et une approche hybride proposé par Snyder et Daskin [133], elle combine entre AG, clés aléatoires et recherche locale par voisinages 2Opt et swap. Ci-dessous sont présentés les mesures utilisées pour évaluer l'approche mémétique basée sur BLS (MBLS).

i. L'écart moyen entre les solutions obtenus et l'optimum global, noté *gap*.

$$gap = 100 \times (f_{avg} - opt)/opt[\%] \quad (3\ 8)$$

f_{avg} étant la moyenne des fitness des 20 solutions obtenus des 20 différentes exécutions, et *opt* l' optimum global de l'instance, atteint pour les instances de la GTSPLIB d'au plus 89 clusters et 442 nœuds [54].

ii. Le temps d'exécution (CPU) en secondes.

En plus des deux méthodes citées précédemment, le temps d'exécution obtenu par la méthode exacte du Branch-and-cut[54] figure aussi dans la comparaison. Elle permet aussi d'avoir l'optimum global des instances étudiés.

Tableau 3.1: Étude comparative entre l'approche mémétique basée sur BLS et d'autres AG

Instance	**Villes**	**Clusters**	***opt***	**Temps B&C**	**MBLS**		**Bontoux**		**Snyder**	
					gap	**CPU (s)**	***gap***	**CPU (s)**	***gap***	**CPU (s)**
10att48	48	10	5,394	2.1	-	-	**0.00**	0.57	**0.00**	0.00
10gr48	48	10	1,834	1.9	-	-	**0.00**	0.97	**0.00**	0.50
10hk48	48	10	6,386	3.8	-	-	**0.00**	0.58	**0.00**	0.20
11eil51	51	11	174	2.9	**0.00**	0.05	**0.00**	0.84	**0.00**	0.10
11berlin52	52	11	4,040	-	**0.00**	0.03	-	-	-	-
12brazil58	58	12	15,332	3.0	-	-	**0.00**	0.85	**0.00**	0.30
14st70	70	14	316	7.3	**0.00**	0.09	**0.00**	1.02	**0.00**	0.20
16eil76	76	16	209	9.4	**0.00**	0.42	**0.00**	1.18	**0.00**	0.20
16pr76	76	16	64,925	12.9	**0.00**	0.64	**0.00**	1.27	**0.00**	0.20
20gr96	96	20	29,440	19.4	**0.00**	1.32	-	-	-	-
20kroA100	100	20	9,711	18.4	**0.00**	0.57	**0.00**	1.98	**0.00**	0.40
20kroB100	100	20	10,328	22.2	**0.00**	0.68	**0.00**	2.01	**0.00**	0.40
20kroC100	100	20	9,554	14.4	**0.00**	0.60	**0.00**	1.84	**0.00**	0.30
20kroD100	100	20	9,450	14.3	**0.00**	0.56	**0.00**	2.93	**0.00**	0.40
20kroE100	100	20	9,523	13.0	**0.00**	1.21	**0.00**	1.87	**0.00**	0.80
20rat99	99	20	497	51.5	**0.00**	3.13	**0.00**	3.52	**0.00**	0.70
20rd100	100	20	3,650	16.6	**0.00**	2.12	**0.00**	2.93	**0.00**	0.30
21eil101	101	21	249	25.6	**0.00**	1.14	**0.00**	2.07	**0.00**	0.20

Instance	**Villes**	**Clusters**	***opt***	**Temps B&C**	**MBLS**		**Bontoux**		**Snyder**	
					gap	**CPU (s)**	***gap***	**CPU (s)**	***gap***	**CPU (s)**
21lin105	105	21	8,213	16.4	**0.00**	0.64	**0.00**	3.25	**0.00**	0.30
22pr107	107	22	27,898	7.4	**0.00**	0.68	**0.00**	4.67	**0.00**	0.40
24gr120	120	24	2,769	41.9	-	-	**0.00**	2.3	**0.00**	0.50
25pr124	124	25	36,605	25.9	**0.00**	2.65	**0.00**	2.89	**0.00**	0.60
26ch130	130	26	2,828	-	**0.00**	5.74	-	-	-	-
26bier127	127	26	72,418	23.6	**0.00**	1.33	**0.00**	3.02	**0.00**	0.50
28gr137	137	28	36,417	-	**0.00**	6.09	-	-	-	-
28pr136	136	28	42,570	43.0	**0.00**	6.34	**0.00**	4.17	**0.00**	0.50
29pr144	144	29	45,886	8.2	**0.00**	1.75	**0.00**	5.38	**0.00**	0.30
30ch150	150	30	2,750	-	**0.00**	9.49	-	-	-	-
30kroA150	150	30	11,018	100.3	**0.00**	15.77	**0.00**	5.27	**0.00**	1.30
30kroB150	150	30	12,196	60.6	**0.00**	10.21	**0.00**	4.61	**0.00**	1.00
31pr152	152	31	51,576	94.8	**0.00**	4.09	**0.00**	4.45	**0.00**	1.50
32u159	159	32	22,664	146.4	**0.00**	8.46	**0.00**	5.53	**0.00**	0.60
39rat195	195	39	854	245.9	**0.00**	9.55	**0.00**	10.42	**0.00**	0.70
40d198	198	40	10,557	763.1	**0.00**	34.00	**0.00**	8.72	**0.00**	1.20
40kroA200	200	40	13,406	187.4	**0.00**	45.00	**0.00**	6.74	**0.00**	2.70
40kroB200	200	40	13,111	268.5	**0.00**	71.00	**0.00**	8.78	**0.00**	1.40
41gr202	202	41	23,301	1,022	**0.00**	81.00	-	-	-	-
45ts225	225	45	68,340	37,875	**0.00**	141.00	0.04	35.31	**0.00**	2.40
46gr229	229	46	71,972	1,187	**0.00**	56.00	-	-	-	-
46pr226	226	46	64,007	106.9	-	-	**0.00**	6.92	**0.00**	1.00
53gil262	262	53	1,013	6,624	**0.09**	346.00	0.14	25.12	0.79	1.90
53pr264	264	53	29,549	337.0	-	-	**0.00**	16.64	**0.00**	1.30
56a280	280	56	1,079	-	**0.12**	3861.00	-	-	-	-
60pr299	299	60	22,615	812.8	-	-	**0.00**	20.19	**0.00**	6.10
64lin318	318	64	20,765	1,671	-	-	**0.00**	24.89	**0.00**	3.50

Instance	Villes	Clusters	*opt*	Temps B&C	MBLS		Bontoux		Snyder	
					gap	CPU (s)	*gap*	CPU (s)	*gap*	CPU (s)
80rd400	400	80	6,361	7,021	-	-	**0.42**	38.33	1.37	3.50
84fl417	417	84	9,651	16,719	0.17	4733.00	**0.00**	21.9	0.07	2.40
88pr439	439	88	60,099	5,422	**0.00**	2733.00	**0.00**	56.46	0.23	9.10
89pcb442	442	89	21,657	58,770	1.28	2437.00	**0.19**	76.64	1.31	10.10
Moyenne					0.04	375.21	0.02	10.46	0.09	1.46

Les résultats obtenus de ces tests sur l'approche mémétique basée sur BLS sur assez satisfaisants. L'approche réussi à atteindre la solution optimale 35 fois sur les 39 instances étudiées, tandis que l'écart moyen sur l'ensemble des instances est de 0.04% ce qui permet à cette approche d'être compétitive vis à vis des autres méthodes utilisées dans la comparaison. L'approche reste néanmoins gourmande en temps d'exécution, ce qui n'est pas étrange pour BLS. Rappelons que BLS a été améliorée dans ce travail coté complexité, et que sans cette contribution le temps d'exécution aurait été largement plus grand.

3.7 Conclusion

Une nouvelle approche mémétique a été présentée dans ce chapitre pour résoudre le GTSP, cette approche est caractérisée par l'adoption de BLS dans la phase d'amélioration. BLS est une métaheuristique performante ayant fait ses preuves sur plusieurs problèmes d'optimisation combinatoires. Les résultats rapportés sur tous les travaux précédents (y compris celui-ci) sont très convaincants, surtout en terme de fitness et d'écart de l'optimum global ou la meilleure solution connue.

Les résultats publiés dans la section précédente démontre la compétitivité de l'approche mémétique basée sur BLS en atteignant les solutions optimales pour la plupart des instances étudiées. L'étude comparative faite consolide les bonnes performances de cette approche, les deux méthodes comparées

étant connues parmi les meilleurs dans la littérature. La complexité de la métaheuristique reste contraignante à cause de certains composants "lourds", ce qui peut être considéré comme un inconvénient majeur, tout dépend des domaines applications. Une proposition d'amélioration sur le composant de perturbation a été apportée dans cette adaptation, en réduisant sa complexité de $O(m^2)$ à $O(m)$.

Durant les travaux effectués sur le GTSP dans cette thèse, et particulièrement cette adaptation, la possibilité de réduire la taille de l'instance a été observée, étant donné qu'une solution faisable ne passe que par une seule ville de chaque cluster et donc d'autres peuvent être retirées tant que le cluster possède plus d'une ville. Ceci permettrait non seulement à notre approche de tourner plus rapidement, mais à n'importe qu'elle autre solveur du GTSP.

Chapitre 4: NN2C: une nouvelle méthode de réduction pour le GTSP

4.1 Introduction

La difficulté de résoudre des instances de tailles assez larges a été discutée dans le premier chapitre de cette thèse. Que ça soit pour les méthodes exactes ou pour les heuristiques, les limites diffèrent selon leurs complexités temporelles et spatiales respectives mais existent toujours. Les méthodes de prétraitement visant à réduire la taille de l'espace de recherche s'avèrent dans certains cas très utiles puisqu'elles permettent d'obtenir des solutions de qualité proche ou même similaire de celles obtenus sans réduction.

Ce chapitre aborde comme le précédent le GTSP, qui est un des POC où la réduction de dimension peut être appliquée. Une solution à ce problème consiste à visiter un seul nœud de chaque cluster et retourner au nœud de départ. Puisque chaque cluster contient dans la plupart des cas plusieurs nœuds, certains de ces derniers peuvent être retirés tout en gardant des solutions faisables dans l'espace de recherche.

Une nouvelle approche de réduction de la taille des instances du GTSP sera introduite dans les prochaines sections. Elle consiste à garder les nœuds les plus proches entre eux et qui sont de différents clusters, puis retirer les autres nœuds considérés loin des autres clusters et donc ne mèneront probablement pas à de bonnes solutions.

L'approche proposée est testée sur un large ensemble d'instances de différents types et différentes tailles sélectionnées de deux librairies: GTSPLIB [58] et MOM [134]. Les instances issues de cette réduction, dont le taux atteint les 98%, fournissent de très bonnes solutions proches de l'optimum global en utilisant des solveurs connus dans la littérature. Les instances réduites donnent même des solutions meilleures que les réelles dans un environnement à budget de temps limité.

4.2 Revue de la littérature

L'approche proposée dans ce chapitre n'est pas la première consacrée au GTSP pour réduire la taille de ses instances, Gutin et Karapetyan [135] ont présenté auparavant l'approche suivante: Retirer un nœud r du cluster C si pour chaque paire de nœuds (x,y) appartenant à des clusters distincts et différents de C, il existe un nœud v de C$\setminus\{r\}$ tel que c(x,v)+c(v,y) $\leq c(x,r)+c(r,y)$. En d'autres termes, si pour chaque chemin xry il existe un autre xvy (v $\in C \setminus \{r\}$) ayant une distance inférieure ou égale, le nœud r peut être supprimé. La complexité de cette approche est de O($n^3 \times |n|$) avec |n| la taille moyenne des clusters. Cette réduction garantit de garder les nœuds qui constituent la solution optimale.

La triangulation de Delaunay [44], la réduction par la méthode POPMUSIC [46] ou encore la réduction par les vaccins artificiels [40], [41] sont des approches citées dans la littérature qui peuvent être appliquées au GTSP. Toutefois, ils n'ont pas été mis à l'essaie durant cette thèse.

4.3 NN2C: la nouvelle approche

La dimension d'une instance GTSP peut être réduite en retirant quelques nœuds étant donné qu'un seul doit être visité dans chaque cluster. La réduction peut donc supprimer de zéro jusqu'à $n - m$ nœuds, tout dépend de la stratégie adoptée.

Ce travail [136] propose une nouvelle stratégie de réduction nommée "Nearest Nodes to Clusters (NN2C)" (Les plus proches nœuds des clusters). Elle consiste à garder les nœuds les plus proches des autres clusters et retirer ceux qui sont assez loin. L'hypothèse derrière cette approche est que les nœuds sélectionnés sont les candidats légitimes pour former de bonnes solutions. Les tests expérimentaux de la prochaine section montreront que les meilleures solutions connues dans la littérature sont parfois atteintes avec les instances réduites avec NN2C.

L'algorithme NN2C parcourt toutes les paires de clusters (C_i,C_j) et sélectionne le couple de nœuds (v_x,v_y) appartenant à différents clusters et ayant la

plus petite distance. Le processus de sélection est répété jusqu'à ce que tous les clusters soient examinés. À noter qu'un nœud peut être sélectionné sur plusieurs itérations de l'algorithme, mais ne figurera qu'une seule fois dans l'ensemble des nœuds sélectionnés. L'algorithme 4.1 donne le processus détaillé de la méthode de réduction NN2C.

Algorithme 4.1: Nearest Nodes to Clusters (NN2C)
Entrées: une instance **I** ayant un ensemble **C** de *m* clusters et un ensemble **V** de *n* nœuds distribué sur **C**.
Sortie: I réduite avec un ensemble **C** de *m* clusters et un ensemble **V'⊂V** de *n'* (*n'*<*n*) nœuds distribué sur **C**.
pour chaque couple (Ci,Cj) de C **faire**
    sélectionner un nœud v_x de C_i et un nœud v_y de C_j tq $c(v_x,v_y)<c(v_a,v_b)$ $\forall$ ($v_a \in C_i$, $v_b \in C_j$)
    si ($v_x \notin V'$) **alors**
        ajouter v_x à V'
    fin si
    si ($v_y \notin V'$) **alors**
        ajouter v_y à V'
    fin si
fin pour
remplacer V par V'
retourner I

La complexité de NN2C est de $O(m^2|n|^2)$, |n| est la taille moyenne des cluster. La figure 4.1 montre un exemple de nœuds sélectionnés et ceux retirés dans une petite instance de quatre clusters. Les nœuds représentés par un cercle sont les nœuds (v_x,v_y) sélectionnés à chaque itération. Une fois toutes les paires parcourues, les nœuds non sélectionnés (illustrés par des croix) peuvent être supprimés de l'instance pour la réduire.

NN2C donne théoriquement un taux de réduction allant de 0.00% (aucun nœud supprimé) jusqu'à ((n-m)/n) $\times$100%, correspondant à la sélection d'un seul nœud de chaque cluster. Cette variance dans le taux de réduction dépend de la topologie et la configuration de l'instance, plusieurs paramètres interdépendants affectent ce taux:

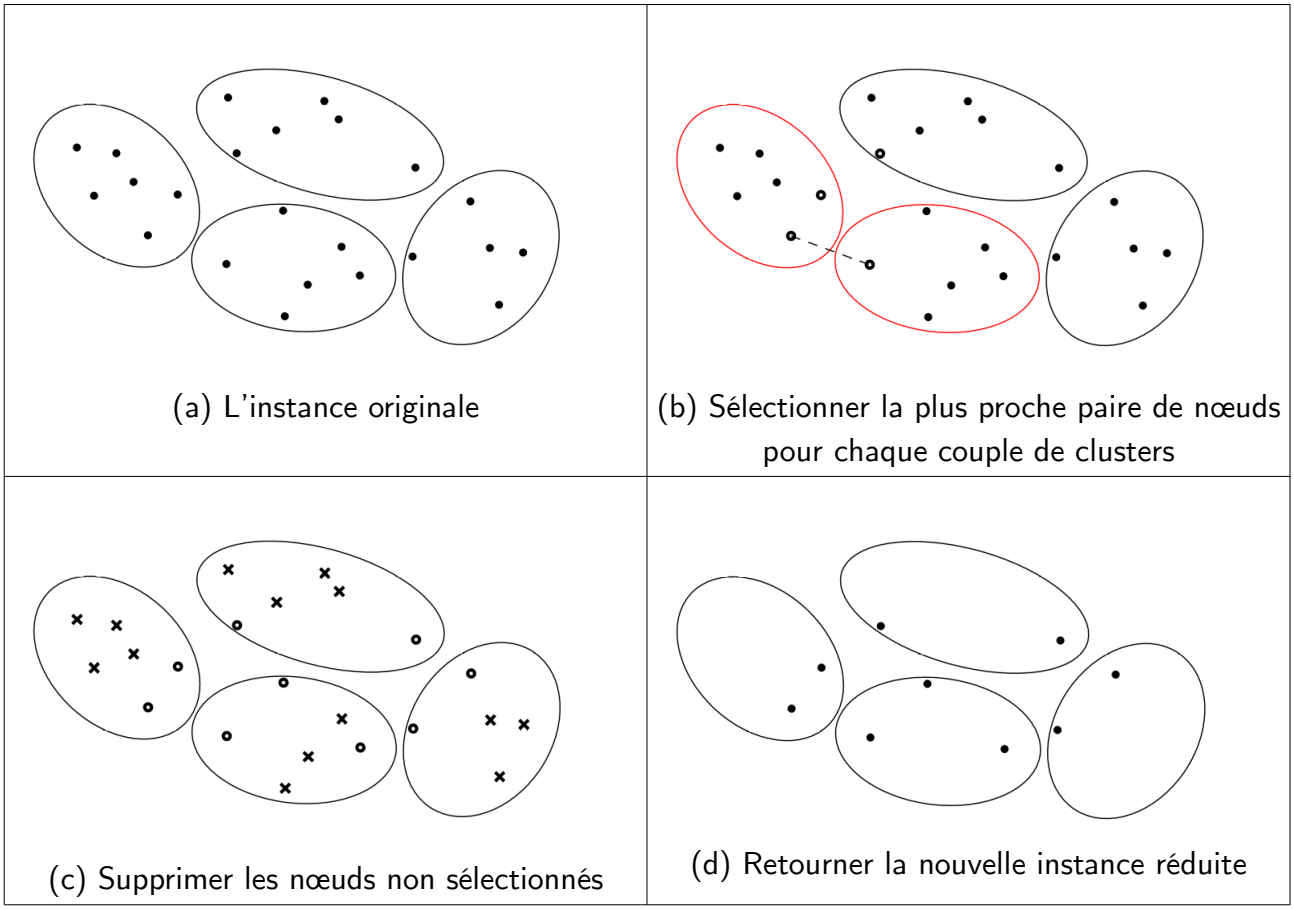


Figure 4.1: Application de l'approche NN2C sur une instance de 4 clusters

- **Le nombre de nœuds par cluster**: Un cluster contenant un grand nombre de nœuds implique une grande probabilité d'obtenir un bon taux de réduction sur ce cluster. NN2C sélectionne au maximum 1 – m nœuds pour chaque cluster, chacun ayant un grand nombre de nœuds sera soumis à un grand taux de réduction, contrairement aux clusters avec un nombre de nœuds plus petit. Le terme densité sera utilisé dans la suite de ce chapitre pour faire référence à ce paramètre.

- **La position géographique de chaque cluster**: La position d'un cluster par rapport aux autres est importante pour prédire le taux de réduction. Si un cluster C est entouré par d'autres, il est alors plus probable que NN2C sélectionne plusieurs nœuds de C étant donné que chacun sera plus proche d'un cluster. Tandis que si C est excentré dans l'espace géographique, certains nœuds seront sélectionné plusieurs fois puisqu'ils seront plus proches de plusieurs clusters à la fois.

- **La distribution des nœuds au sein du cluster**: La position des différents nœuds dans le cluster influence aussi le taux de réduction. Si les nœuds sont bien dispersés sur C, NN2C sélectionnera probablement un nœud différent pour chaque cluster. À la différence des nœuds qui sont isolés des autres et qui pourraient être les plus proches à plusieurs clusters en même temps.

Les figures 4.2 et 4.3 sont deux exemples d'instances simples montrant respectivement le taux de réduction minimale et maximale.

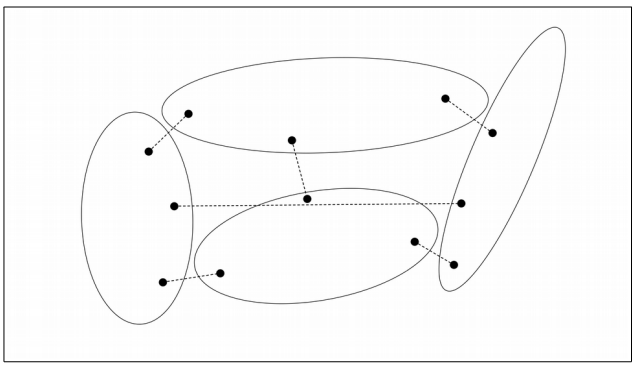


Figure 4.2: Réduction minimale, aucun nœud supprimé

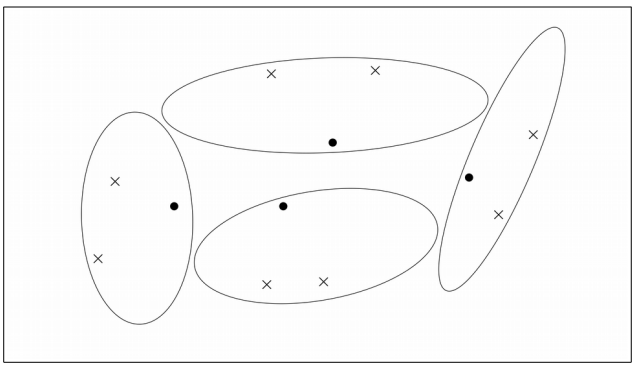


Figure 4.3: Réduction maximale, un seul nœud est sélectionné de chaque cluster

Les figures 4.4 et 4.5 montrent un exemple de réduction de l'instance *3burma14* disponible sur le benchmark TSPLIB (en tant que *burma14*) fournie par Reinelt [58] et convertie en une instance GTSP par la procédure de clustering standard [54]. NN2C a réduit la dimension de cette instance de 14 à 5. Les nœuds sont divisées en trois couleurs, chacune représentant un cluster. Les nœuds en croix sont ceux qui ont été supprimés par NN2C.

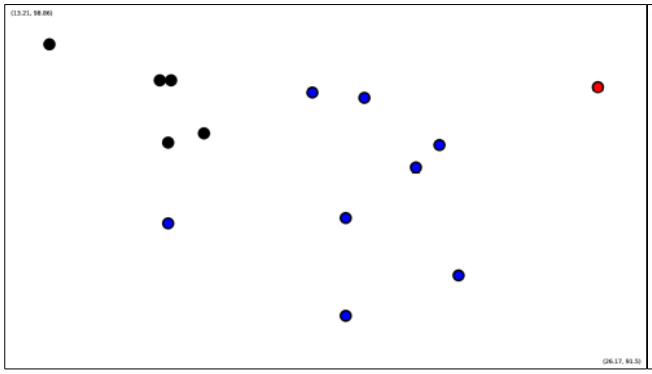


Figure 4.4: L'instance originale **3burma14**

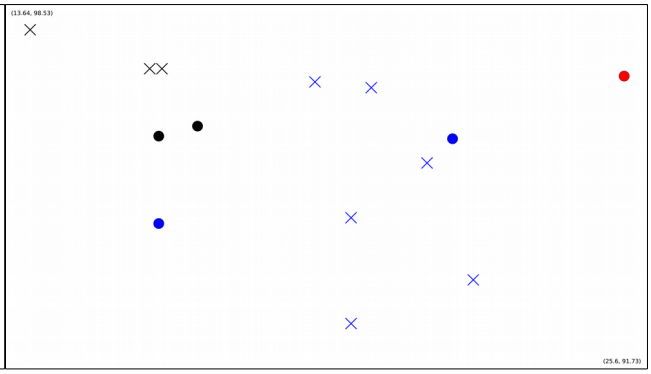


Figure 4.5: L'instance **3burma14** réduite par NN2C

4.4 Test expérimentaux et discussions

La méthode de réduction NN2C a été testé sur 88 instances de différents types de la GTSPLIB [58] dont les tailles varient de 14 nœuds et 3 clusters jusqu'à 24978 nœuds et 4996 clusters, et sur l'ensemble des instances de la librairie MOM [134],

La première partie de l'étude expérimentale expose les performances de la méthode de réduction NN2C et la compare avec la méthode de réduction de Gutin et Karapetyan [135]. La deuxième partie est dédiée à la résolution des instances originales et réduites en utilisant les solveurs Generalized Lin-Kernighan-Helsgaun (GLKH) [137] et GLNS [138] dans le but d'évaluer l'amélioration du temps d'exécution obtenu par NN2C d'un côté, et de l'autre la dégradation de la qualité des solutions causée par cette approche. Une généralisation de NN2C est proposée dans la troisième et dernière partie de ces tests, elle consiste à sélectionner $k \geq 1$ plus proches nœuds pour chaque couple de clusters.

4.4.1 Les taux de réduction obtenus par NN2C

L'algorithme de réduction NN2C a été testé sur différentes instances des librairies GTSPLIB et MOM. Ce test inclut différents types d'instances (Euclidien, pseudo-Euclidien et géométrique), petites, larges et très larges. Les tableaux 4.1 et 4.2 listent les résultats obtenus de toutes les instances étudiées, pour chacune sera noté le nombre de clusters et le nombre de nœuds, puis le nombre de

nœuds supprimés par NN2C, le taux de réduction ρ calculé par la formule 4.1, et le temps d'exécution en secondes.

$$\rho = \frac{|V'|-|V|}{|V|} \times 100 \quad (4\ 1)$$

Tableau 4.1: Performances de NN2C sur la GTSPLIB

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ **(%)**	**t (s)**
3burma14	3	14	9	64.29	0.00
4ulysses16	4	16	10	62.5	0.00
5ulysses22	5	22	12	54.55	0.00
10att48	10	48	23	47.92	0.00
11eil51	11	51	15	29.41	0.00
11berlin52	11	52	25	48.08	0.00
14st70	14	70	28	40	0.00
16eil76	16	76	27	35.53	0.00
16pr76	16	76	27	35.53	0.00
20gr96	20	96	39	40.63	0.01
20rat99	20	99	30	30.3	0.01
20kroA100	20	100	35	35	0.01
20kroB100	20	100	30	30	0.00
20kroC100	20	100	42	42	0.01
20kroD100	20	100	40	40	0.00
20kroE100	20	100	41	41	0.00
20rd100	20	100	34	34	0.00
21eil101	21	101	37	36.63	0.01
21lin105	21	105	45	42.86	0.01
22pr107	22	107	32	29.91	0.00
25pr124	25	124	47	37.9	0.01
26bier127	26	127	53	41.73	0.01

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
26ch130	26	130	43	33.08	0.01
28pr136	28	136	40	29.41	0.01
28gr137	28	137	49	35.77	0.01
29pr144	29	144	68	47.22	0.01
30ch150	30	150	46	30.67	0.01
30kroA150	30	150	47	31.33	0.01
30kroB150	30	150	56	37.33	0.01
31pr152	31	152	58	38.16	0.01
32u159	32	159	58	36.48	0.01
39rat195	39	195	55	28.21	0.01
40d198	40	198	67	33.84	0.01
40kroa200	40	200	67	33.5	0.01
40krob200	40	200	58	29	0.01
41gr202	41	202	65	32.18	0.01
45ts225	45	225	105	46.67	0.01
46gr229	46	229	79	34.5	0.01
49usa1097	49	1097	800	72.93	0.07
53gil262	53	262	74	28.24	0.01
56a280	56	280	66	23.57	0.02
87gr431	87	431	183	42.46	0.04
88pr439	88	439	146	33.26	0.04
89pcb442	89	442	109	24.66	0.04
99d493	99	493	166	33.67	0.04
107att532	107	532	180	33.83	0.05
107ali535	107	535	219	40.93	0.04
115rat575	115	575	133	23.13	0.04
115u574	115	574	144	25.09	0.05
131p654	131	654	308	47.09	0.05

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
132d657	132	657	173	26.33	0.05
134gr666	134	666	229	34.38	0.05
145u724	145	724	182	25.14	0.06
157rat783	157	783	174	22.22	0.07
200dsj1000	200	1000	277	27.7	0.1
201pr1002	201	1002	253	25.25	0.09
212u1060	212	1060	288	27.17	0.08
217vm1084	217	1084	395	36.44	0.11
235pcb1173	235	1173	242	20.63	0.12
259d1291	259	1291	196	15.18	0.12
261rl1304	261	1304	266	20.4	0.11
265rl1323	265	1323	278	21.01	0.11
276nrw1379	276	1379	290	21.03	0.12
280fl1400	280	1400	573	40.93	0.13
287u1432	287	1432	208	14.53	0.12
316fl1577	316	1577	333	21.12	0.11
331d1655	331	1655	262	15.83	0.13
350vm1748	350	1748	654	37.41	0.12
364u1817	364	1817	173	9.52	0.18
378rl1889	378	1889	415	21.97	0.18
421d2103	421	2103	205	9.75	0.16
431u2152	431	2152	213	9.9	0.19
464u2319	464	2319	331	14.27	0.24
479pr2392	479	2392	489	20.44	0.22
608pcb3038	608	3038	539	17.74	0.22
759fl3795	759	3795	1008	26.56	0.29
893fnl4461	893	4461	814	18.25	0.31
1183rl5915	1183	5915	1067	18.04	0.64

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
1187rl5934	1187	5934	1047	17.64	0.57
1480pla7397	1480	7397	1862	25.17	0.7
2000C10k.0.00	2000	10000	2917	29.17	1.3
2000E10k.0.00	2000	10000	2039	20.39	1.33
2370rl11849	2370	11849	1761	14.86	1.89
2702usa13509	2702	13509	4263	31.56	2.21
2811brd14051	2811	14051	3014	21.45	2.32
3023d15112	3023	15112	3670	24.29	2.94
3703d18512	3703	18512	3851	20.8	4.02
4996sw24978	4996	24978	6417	25.69	7.38
moyenne				**31.09**	**0.34**

Tableau 4.2: Performances de NN2C sur la librairie MOM

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
2lin105-2x1	2	105	103	98.1	0.00
4berlin52-2x2	4	52	46	88.46	0.00
4eil51-2x2	4	51	41	80.39	0.00
4eil76-2x2	4	76	69	90.79	0.01
4i200a	4	200	188	94	0.00
4i200h	4	200	188	94	0.00
4i200x1	4	200	188	94	0.00
4i200x2	4	200	188	94	0.00
4i200z	4	200	188	94	0.01
4i400a	4	400	388	97	0.01

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
4i400h	4	400	390	97.5	0.01
4i400x1	4	400	391	97.75	0.01
4i400x2	4	400	391	97.75	0.01
4i400z	4	400	391	97.75	0.00
4pr76-2x2	4	76	66	86.84	0.00
5berlin52	5	52	38	73.08	0.00
5eil51	5	51	34	66.67	0.00
5eil76	5	76	60	78.95	0.00
5i120-46	5	120	108	90	0.01
5i300-108	5	300	286	95.33	0.00
5i30-17	5	30	18	60	0.02
5i400-205	5	400	386	96.5	0.00
5i45-18	5	45	29	64.44	0.02
5i500-304	5	500	484	96.8	0.00
5i60-21	5	60	48	80	0.00
5i65-21	5	65	53	81.54	0.00
5i70-21	5	70	58	82.86	0.00
5i75-22	5	75	62	82.67	0.00
5i90-33	5	90	75	83.33	0.00
5pr76	5	76	61	80.26	0.00
5st70	5	70	57	81.43	0.00
5ulysses22	5	22	12	54.55	0.00
6berlin52-2x3	6	52	37	71.15	0.01
6i300	6	300	278	92.67	0.01
6i350	6	350	328	93.71	0.01
6i400	6	400	379	94.75	0.02
6i450	6	450	425	94.44	0.03
6i500	6	500	472	94.4	0.00

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
6pr76-2x3	6	76	58	76.32	0.00
6st70-2x3	6	70	54	77.14	0.00
7i30-17	7	30	10	33.33	0.00
7i45-18	7	45	24	53.33	0.00
7i60-21	7	60	41	68.33	0.00
7i65-21	7	65	43	66.15	0.00
7i70-21	7	70	49	70	0.00
8berlin52-2x4	8	52	29	55.77	0.06
8i1000a	8	1000	944	94.4	0.04
8i1000h	8	1000	961	96.1	0.05
8i1000x1	8	1000	961	96.1	0.05
8i1000x2	8	1000	961	96.1	0.05
8i1000z	8	1000	960	96	0.03
8i600a	8	600	545	90.83	0.03
8i600h	8	600	564	94	0.03
8i600x1	8	600	561	93.5	0.03
8i600x2	8	600	561	93.5	0.03
8i600z	8	600	561	93.5	0.01
9a280-3x3	9	280	244	87.14	0.00
9eil101-3x3	9	101	67	66.34	0.00
9eil51-3x3	9	51	21	41.18	0.00
9eil76-3x3	9	76	45	59.21	0.01
9gil262-3x3	9	262	223	85.11	0.01
9lin318-3x3	9	318	276	86.79	0.02
9pcb442-3x3	9	442	403	91.18	0.02
9pr439-3x3	9	439	403	91.8	0.00
9pr76-3x3	9	76	46	60.53	0.00
9st70-3x3	9	70	41	58.57	0.01

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
10a280	10	280	236	84.29	0.00
10berlin52-2x5	10	52	25	48.08	0.00
10berlin52	10	52	23	44.23	0.05
10C1k.0.00	10	1000	962	96.2	0.05
10C1k.1	10	1000	952	95.2	0.06
10C1k.2	10	1000	958	95.8	0.06
10C1k.3	10	1000	957	95.7	0.06
10C1k.4	10	1000	957	95.7	0.05
10C1k.5	10	1000	957	95.7	0.05
10C1k.6	10	1000	958	95.8	0.05
10C1k.7	10	1000	955	95.5	0.04
10C1k.8	10	1000	960	96	0.05
10C1k.9	10	1000	961	96.1	0.00
10eil51	10	51	18	35.29	0.00
10eil76	10	76	42	55.26	0.01
10gil262	10	262	216	82.44	0.05
10i1000-407	10	1000	955	95.5	0.00
10i120-46	10	120	81	67.5	0.07
10i1400a	10	1400	1315	93.93	0.07
10i1400h	10	1400	1344	96	0.07
10i1400	10	1400	1344	96	0.07
10i1400x1	10	1400	1344	96	0.08
10i1400x2	10	1400	1345	96.07	0.08
10i1400z	10	1400	1345	96.07	0.07
10i1500-503	10	1500	1461	97.4	0.09
10i2000a	10	2000	1911	95.55	0.09
10i2000h	10	2000	1946	97.3	0.08

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
10i2000x1	10	2000	1943	97.15	0.08
10i2000x2	10	2000	1940	97	0.08
10i2000z	10	2000	1940	97	0.01
10i300-109	10	300	251	83.67	0.00
10i30-17	10	30	7	23.33	0.02
10i400-206	10	400	355	88.75	0.00
10i45-18	10	45	11	24.44	0.03
10i500-305	10	500	459	91.8	0.00
10i60-21	10	60	30	50	0.00
10i65-21	10	65	31	47.69	0.00
10i70-21	10	70	35	50	0.00
10i75-22	10	75	45	60	0.00
10i90-33	10	90	51	56.67	0.00
10kroB100	10	100	66	66	0.01
10lin318	10	318	282	88.68	0.08
10nrw1379-2x5	10	1379	1329	96.37	0.02
10pcb442	10	442	396	89.59	0.02
10pr439	10	439	395	89.98	0.00
10pr76	10	76	43	56.58	0.00
10rat99	10	99	58	58.59	0.00
10st70	10	70	40	57.14	0.00
12eil51-3x4	12	51	14	27.45	0.00
12eil76-3x4	12	76	38	50	0.07
12nrw1379-2x6	12	1379	1303	94.49	0.08
12nrw1379-3x4	12	1379	1306	94.71	0.00

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
12pr76-3x4	12	76	40	52.63	0.00
12st70-3x4	12	70	36	51.43	0.00
15berlin52	15	52	14	26.92	0.00
15eil51	15	51	10	19.61	0.00
15eil76	15	76	27	35.53	0.01
15i300-110	15	300	223	74.33	0.02
15i400-207	15	400	331	82.75	0.02
15i500-306	15	500	429	85.8	0.00
15pr76-3x5	15	76	37	48.68	0.00
15pr76	15	76	25	32.89	0.00
15st70	15	70	27	38.57	0.00
16eil51-4x4	16	51	10	19.61	0.00
16eil76-4x4	16	76	24	31.58	0.00
16lin105-4x4	16	105	56	53.33	0.00
16st70-4x4	16	70	25	35.71	0.02
18pr439-3x6	18	439	364	82.92	0.00
18pr76-3x6	18	76	24	31.58	0.00
20eil51-4x5	20	51	6	11.76	0.00
20eil76-4x5	20	76	17	22.37	0.05
20i1000-408	20	1000	900	90	0.07
20i1500-504	20	1500	1395	93	0.09
20i2000-602	20	2000	1888	94.4	0.12
20i2400a	20	2400	2042	85.08	0.1
20i2400h	20	2400	2234	93.08	0.12
20i2400x1	20	2400	2234	93.08	0.11
20i2400x2	20	2400	2233	93.04	0.12
20i2400z	20	2400	2250	93.75	0.12
20i2400z	20	2400	2250	93.75	0.12

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
20i2500-706	20	2500	2388	95.52	0.14
20i3000-801	20	3000	2886	96.2	0.13
20i3000a	20	3000	2634	87.8	0.14
20i3000h	20	3000	2846	94.87	0.14
20i3000x1	20	3000	2843	94.77	0.14
20i3000x2	20	3000	2842	94.73	0.26
20i3000z	20	3000	2829	94.3	0.01
20i300-111	20	300	198	66	0.03
20i400-208	20	400	295	73.75	0.02
20i500-307	20	500	388	77.6	0.03
20i550	20	550	426	77.45	0.03
20i600	20	600	475	79.17	0.03
20i650	20	650	525	80.77	0.04
20i700	20	700	569	81.29	0.03
20pr439-4x5	20	439	356	81.09	0.00
20st70-4x5	20	70	19	27.14	0.00
21lin105	21	105	45	42.86	0.01
25a280-5x5	25	280	172	61.43	0.01
25a280	25	280	160	57.14	0.00
25eil101-5x5	25	101	26	25.74	0.00
25eil101	25	101	19	18.81	0.00
25eil51-5x5	25	51	4	7.84	0.00
25eil76-5x5	25	76	13	17.11	0.01
25gil262-5x5	25	262	145	55.34	0.01
25gil262	25	262	145	55.34	0.01
25i300-112	25	300	165	55	0.02
25i400-209	25	400	267	66.75	0.02
25i500-308	25	500	355	71	0.04

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
25i750	25	750	595	79.33	0.04
25i800	25	800	640	80	0.05
25i850	25	850	677	79.65	0.05
25i900	25	900	721	80.11	0.00
25kroA100	25	100	27	27	0.00
25lin105	25	105	37	35.24	0.01
25lin318-5x5	25	318	209	65.72	0.01
25lin318	25	318	200	62.89	0.03
25pcb442-5x5	25	442	324	73.3	0.03
25pcb442	25	442	310	70.14	0.03
25pr439	25	439	309	70.39	0.00
25rat99-5x5	25	99	21	21.21	0.00
25rat99	25	99	19	19.19	0.00
28kroA100-4x7	28	100	30	30	0.05
30i1000	30	1000	779	77.9	0.05
30i950	30	950	735	77.37	0.00
30kroB100-5x6	30	100	21	21	0.00
35kroB100-5x5	25	100	31	31	0.00
36eil101-6x6	36	101	9	8.91	0.02
36pcb442-6x6	36	442	279	63.12	0.05
36pr1002-6x6	36	1002	773	77.15	0.01
42a280-6x7	42	280	107	38.21	0.05
42pr1002-6x7	42	1002	741	73.95	0.00
42rat99-6x7	42	99	6	6.06	0.01
49gil262-7x7	49	262	75	28.63	0.01

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
49lin318-7x7	49	318	135	42.45	0.06
49pcb1173-7x7	49	1173	838	71.44	0.05
49pr1002-7x7	49	1002	696	69.46	0.04
49rat783-7x7	49	783	482	61.56	0.07
49vm1084-7x7	49	1084	811	74.82	0.01
50a280	50	280	87	31.07	0.00
50eil101	50	101	6	5.94	0.01
50gil262	50	262	79	30.15	0.07
50i1000-409	50	1000	700	70	0.11
50i1500-505	50	1500	1173	78.2	0.08
50i2000-603	50	2000	1675	83.75	0.13
50i2500-707	50	2500	2144	85.76	0.15
50i3000-802	50	3000	2657	88.57	0.00
50kroA100	50	100	8	8	0.00
50kroB100	50	100	5	5	0.00
50lin105	50	105	10	9.52	0.01
50lin318	50	318	115	36.16	0.09
50nrw1379	50	1379	998	72.37	0.08
50pcb1173	50	1173	825	70.33	0.03
50pcb442	50	442	195	44.12	0.07
50pr1002	50	1002	690	68.86	0.03
50pr439	50	439	207	47.15	0.05
50rat783	50	783	478	61.05	0.00
50rat99	50	99	1	1.01	0.07
50vm1084	50	1084	799	73.71	0.08
72vm1084-8x9	72	1084	718	66.24	0.00
75lin105	75	105	3	2.86	0.09

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
81vm1084-9x9	81	1084	694	64.02	0.07
100i1000-410	100	1000	431	43.1	0.09
100i1500-506	100	1500	857	57.13	0.08
100i2000-604	100	2000	1335	66.75	0.13
100i2500-708	100	2500	1797	71.88	0.25
100i3000-803	100	3000	2287	76.23	0.1
100nrw1379	100	1379	722	52.36	0.09
100pcb1173	100	1173	559	47.66	0.06
100pr1002	100	1002	470	46.91	0.08
100prb1173-10x10	100	1173	583	49.7	0.05
100rat783-10x10	100	783	293	37.42	0.04
100rat783	100	783	292	37.29	0.07
100vm1084	100	1084	612	56.46	0.07
144pcb1173-12x12	144	1173	440	37.51	0.04
144rat783-12x12	144	783	170	21.71	0.06
150i1000-411	150	1000	273	27.3	0.07
150i1500-507	150	1500	624	41.6	0.09
150i2000-605	150	2000	1033	51.65	0.15
150i2500-709	150	2500	1489	59.56	0.17
150i3000-804	150	3000	1958	65.27	0.08
150nrw1379	150	1379	527	38.22	0.07
150pcb1173	150	1173	428	36.49	0.08
150pr1002	150	1002	330	32.93	0.06
150rat783	150	783	195	24.9	0.08
150vm1084	150	1084	462	42.62	0.1

Instance	**Clusters**	**Nœuds**	**Nœuds supprimés**	ρ (%)	**t (s)**
200i2000-606	200	2000	784	39.2	0.2
200i2500-710	200	2500	1239	49.56	0.17
200i3000-805	200	3000	1656	55.2	0.00
Moyenne				**67.04**	**0.04**

À partir des tableaux 4.1 et 4.2 ci-dessus, on peut constater que les taux de réduction diffèrent largement d'une instance à une autre. Les instances de la GTSPLIB sont réduites avec taux allant de 9 à 73%, tandis que les instances de la librairie MOM varient entre 1 et 98%. Le nombre de nœuds par clusters est quasi identique pour les instances de la GTSPLIB (procédure de clustering standard [54]), alors que les instances de la librairie MOM sont construites avec différentes densités. La figure 4.6 est un diagramme de dispersion montrant pour chaque point la densité et le taux de réduction pour chaque instance MOM. On remarque que, comme expliqué plus tôt dans ce chapitre, le nombre de nœuds par cluster influence le taux de réduction.

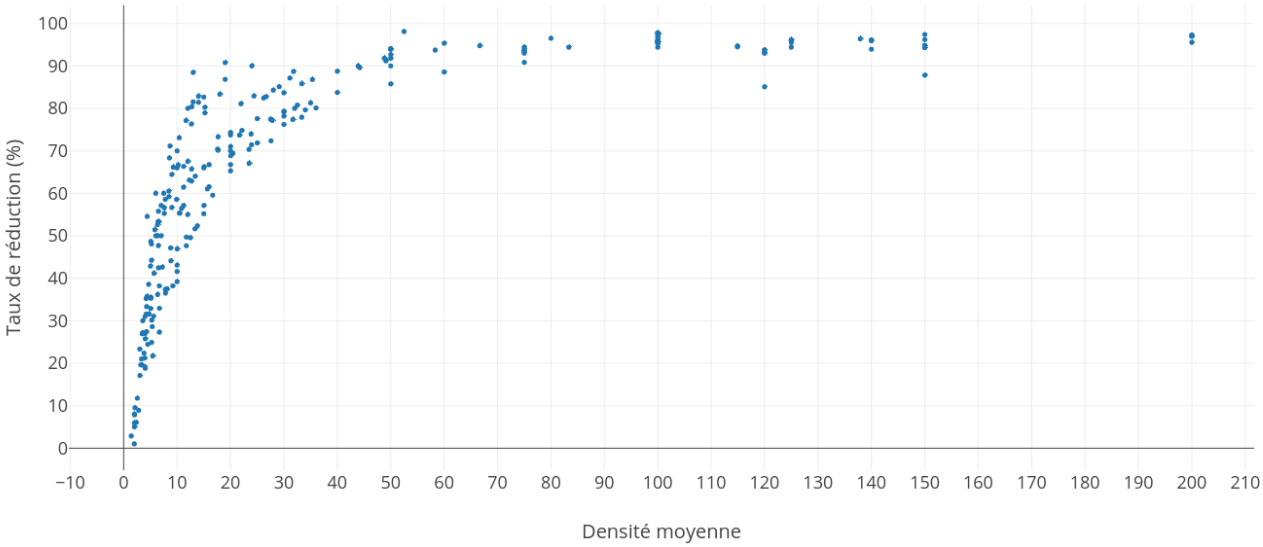


Figure 4.6: Influence de la densité sur le taux de réduction pour les instances MOM

4.4.2Étude comparative

Gutin et Karapetyan [135] ont présenté une méthode de réduction pour le GTSP qu'on a discuté plus tôt dans ce chapitre. L'approche proposée ne dépasse pas 10% (en moyenne) de taux de réduction, ce qui est bien inférieur à ce que propose NN2C. Le temps de réduction est aussi beaucoup plus élevé que celui de NN2C, surtout pour les instances larges. Son unique avantage est la préservation des nœuds de la solution optimale. À notre connaissance, il s'agit de la seule approche de réduction visant le GTSP. Le tableau 4.3 donne une comparaison détaillée entre NN2C et l'approche de Gutin et Karapetyan. Cette comparaison ne concerne que la procédure de prétraitement de la réduction, elle est basée sur les valeurs suivantes:

- Le nombre de nœuds supprimés
- Le taux de réduction (ρ)
- Le temps de réduction (t)

Tableau 4.3: Une comparaison entre les performances de NN2C et celles de la méthode de réduction proposée par Gutin & Karapetyan

Instance	**Clusters**	**Nœuds**	**NN2C**			**Gutin & Karapetyan**		
			Nœuds supprimés	ρ (%)	**t (s)**	**Nœuds supprimés**	ρ (%)	**t (s)**
3burma14	3	14	9	64.29	0.00	-	-	-
4ulysses16	4	16	10	62.5	0.00	9	56.25	0.00
5ulysses22	5	22	12	54.55	0.00	11	50	0.00
10att48	10	48	23	47.92	0.00	15	31.3	0.00
11eil51	11	51	15	29.41	0.00	9	17.65	0.00
11berlin52	11	52	25	48.08	0.00	15	28.85	0.00
14st70	14	70	28	40	0.00	12	17.14	0.00
16eil76	16	76	27	35.53	0.00	12	15.79	0.00
16pr76	16	76	27	35.53	0.00	2	2.63	0.00
20gr96	20	96	39	40.63	0.00	13	13.54	0.00

Instance	Clusters	Nœuds	NN2C			Gutin & Karapetyan		
			Nœuds supprimés	ρ (%)	t (s)	Nœuds supprimés	ρ (%)	t (s)
20rat99	20	99	30	30.3	0.00	11	11.11	0.00
20kroA100	20	100	35	35	0.00	16	16	0.00
20kroB100	20	100	30	30	0.00	8	8	0.00
20kroC100	20	100	42	42	0.00	19	19	0.00
20kroD100	20	100	40	40	0.00	19	19	0.00
20kroE100	20	100	41	41	0.00	21	21	0.00
20rd100	20	100	34	34	0.00	11	11	0.00
21eil101	21	101	37	36.63	0.00	14	13.86	0.00
21lin105	21	105	45	42.86	0.00	9	8.57	0.00
22pr107	22	107	32	29.91	0.00	9	8.41	0.00
25pr124	25	124	47	37.9	0.00	17	13.71	0.00
26bier127	26	127	53	41.73	0.00	2	1.57	0.00
26ch130	26	130	43	33.08	0.01	16	12.31	0.00
28pr136	28	136	40	29.41	0.01	14	10.29	0.00
28gr137	28	137	49	35.77	0.01	10	7.3	0.00
29pr144	29	144	68	47.22	0.01	19	13.19	0.00
30ch150	30	150	46	30.67	0.01	22	14.67	0.00
30kroA150	30	150	47	31.33	0.01	20	13.33	0.00
30kroB150	30	150	56	37.33	0.01	14	9.33	0.00
31pr152	31	152	58	38.16	0.01	34	22.37	0.00
32u159	32	159	58	36.48	0.01	33	20.75	0.00
39rat195	39	195	55	28.21	0.01	12	6.15	0.00
40d198	40	198	67	33.84	0.01	7	3.54	0.00
40kroa200	40	200	67	33.5	0.01	16	8	0.00
40krob200	40	200	58	29	0.01	7	3.5	0.00
41gr202	41	202	65	32.18	0.01	4	1.98	0.00
45ts225	45	225	105	46.67	0.02	40	17.78	0.00

Instance	**Clusters**	**Nœuds**	**NN2C**			**Gutin & Karapetyan**		
			Nœuds supprimés	ρ (%)	**t (s)**	**Nœuds supprimés**	ρ (%)	**t (s)**
46gr229	46	229	79	34.5	0.02	1	0.44	0.00
49usa1097	49	1097	800	72.93	0.08	-	-	-
53gil262	53	262	74	28.24	0.02	16	6.11	0.00
56a280	56	280	66	23.57	0.02	20	7.14	0.00
87gr431	87	431	183	42.46	0.06	0.00	0.00	0.00
88pr439	88	439	146	33.26	0.06	10	2.28	0.00
89pcb442	89	442	109	24.66	0.06	24	5.43	0.00
99d493	99	493	166	33.67	0.08	4	0.81	0.00
107att532	107	532	180	33.83	0.05	21	3.9	0.00
107ali535	107	535	219	40.93	0.07	29	5.42	0.1
115rat575	115	575	133	23.13	0.06	18	3.13	0.00
115u574	115	574	144	25.09	0.06	11	1.92	0.00
131p654	131	654	308	47.09	0.06	88	13.46	0.1.0.00
132d657	132	657	173	26.33	0.08	8	1.22	0.00
134gr666	134	666	229	34.38	0.07	0.00	0.00	0.00
145u724	145	724	182	25.14	0.09	34	4.7	0.1
157rat783	157	783	174	22.22	0.07	25	3.19	0.00
200dsj1000	200	1000	277	27.7	0.1	8	0.8	0.1
201pr1002	201	1002	253	25.25	0.08	20	2	0.1
212u1060	212	1060	288	27.17	0.08	36	3.4	0.1
217vm1084	217	1084	395	36.44	0.08	241	22.23	0.6
235pcb1173	235	1173	242	20.63	0.09	11	0.94	0.1
259d1291	259	1291	196	15.18	0.09	48	3.72	0.2
261rl1304	261	1304	266	20.4	0.11	19	1.46	0.2
265rl1323	265	1323	278	21.01	0.1	23	1.74	0.2
276nrw1379	276	1379	290	21.03	0.1	11	0.8	0.2

Instance	Clusters	Nœuds	NN2C			Gutin & Karapetyan		
			Nœuds supprimés	ρ (%)	t (s)	Nœuds supprimés	ρ (%)	t (s)
280fl1400	280	1400	573	40.93	0.1	23	1.64	0.9
287u1432	287	1432	208	14.53	0.1	33	2.3	0.2
316fl1577	316	1577	333	21.12	0.11	44	2.79	0.4
331d1655	331	1655	262	15.83	0.12	14	0.85	0.2
350vm1748	350	1748	654	37.41	0.19	285	16.3	2.5
364u1817	364	1817	173	9.52	0.12	5	0.28	0.1
378rl1889	378	1889	415	21.97	0.14	17	0.9	0.7
421d2103	421	2103	205	9.75	0.13	8	0.38	0.2
431u2152	431	2152	213	9.9	0.14	10	0.46	0.3
464u2319	464	2319	331	14.27	0.16	24	1.03	0.6
479pr2392	479	2392	489	20.44	0.18	33	1.38	0.9
608pcb3038	608	3038	539	17.74	0.25	29	0.95	1.9
759fl3795	759	3795	1008	26.56	0.29	21	0.55	4.9
893fnl4461	893	4461	814	18.25	0.43	22	0.49	3.4
1183rl5915	1183	5915	1067	18.04	0.68	28	0.47	7.9
1187rl5934	1187	5934	1047	17.64	0.72	38	0.64	9.4
1480pla7397	1480	7397	1862	25.17	0.7	196	2.6	31.5
2000C10k.0.00	2000	10000	2917	29.17	1.3	-	-	-
2000E10k.0.00	2000	10000	2039	20.39	1.33	-	-	-
2370rl11849	2370	11849	1761	14.86	2.98	37	0.31	40.7
2702usa13509	2702	13509	4263	31.56	2.47	21	0.16	98.7
2811brd14051	2811	14051	3014	21.45	2.86	-	-	-
3023d15112	3023	15112	3670	24.29	5.35	-	-	-
3703d18512	3703	18512	3851	20.8	7.04	-	-	-
4996sw24978	4996	24978	18561	25.69	7.6	-	-	-

Instance	Clusters	Nœuds	NN2C			Gutin & Karapetyan		
			Nœuds supprimés	ρ (%)	t (s)	Nœuds supprimés	ρ (%)	t (s)
moyenne				**31.09**	**0.43**		**8.50**	**2.59**

Les résultats présentés dans le tableau 4.3 montrent que NN2C surpasse considérablement l'approche proposée par Gutin et Karapetyan. NN2C offre toujours un meilleur taux de réduction dans un budget de temps largement convenable. Tandis que la complexité de NN2C est $O(m^2|n^2|)$, celle de la méthode comparée est plus grande, elle est de l'ordre de $O(n^3|n|)$, n étant toujours plus grand que m.

Rappelons que l'approche de Gutin et Karapetyan garantit de garder les nœuds constituant la solution optimale, tandis que NN2C peut en retirer. Ceci peut impacter légèrement la qualité des solutions fournis mais améliorera significativement le temps d'exécution.

4.4.3Évaluation des instances réduites par NN2C via des solveurs

Afin d'évaluer les performances de l'approche NN2C, deux solveurs du GTSP sont appliqués sur les instances réduites pour mesurer l'effet de la réduction sur ces instances. Une comparaison entre les instances complètes et réduites est effectuée en se référant au temps d'exécution ainsi que l'écart de la meilleure solution connue (*bks*). Les solveurs choisis sont des travaux récents fournissant de très bonne solutions sur les instances complètes, cependant ce ne sont pas des méthodes exactes (ne garantissent pas d'avoir la solution optimale). Les solveurs sollicités sont:

1. **GLKH**: un solveur proposé par Helsgaun[5] et implémenté en langage C. GLKH est souvent capable d'obtenir les *bks* pour presque toutes les instances, les petites et les plus grandes, que ce soit pour la GTSPLIB ou pour la librairie MOM.

5 Disponible sur http://www.akira.ruc.dk/~keld/research/GLKH/

2. **GLNS**: un travail fait par Smith & Imeson[6] et développé en langage Julia. GLNS est beaucoup plus rapide que GLKH mais moins performant sur les instances larges de la GTSPLIB.

Les tableaux 4.4 et 4.5 montre les résultats d'exécution de GLKH et GLNS sur les instances réduites et complètes. Est rapporté pour chaque instance le taux de réduction ρ et la *bks*, puis pour chaque solveur (i) l'écart moyen de *bks* en pourcentage (calculé par la formule 4.2) δ et (ii) le temps d'exécution T en secondes pour l'instance réduite et complète. Puis (iii) le taux de réduction du temps d'exécution τ pour les deux instances et qui est déterminé par l'équation 4.3.

$$\delta = \frac{Solution - bks}{bks} \times 100 \quad (4\ 2)$$

$$\tau = \frac{T(instance\ complète) - T(instance\ réduite)}{T(instance\ complète)} \times 100 \quad (4\ 3)$$

Tableau 4.4: Résolution des instances GTSPLIB avec les solveurs GLKH et GLNS

Instance	ρ (%)	***bks***	**GLKH**					**GLNS**				
			Instances réduites		**Instances complètes**		τ (%)	**Instances réduites**		**Instances complètes**		τ (%)
			δ (%)	**T (s)**	δ (%)	**T (s)**		δ (%)	**T (s)**	δ (%)	**T (s)**	
3burma14	64.29	1805	1.05	0.00	0.00	0.00	-	1.05	1.15	0.00	1.14	-0.88
4ulysses16	62.5	4539	0.18	0.00	0.00	0.00	-	0.18	1.15	0.00	1.16	0.86
5ulysses22	63.64	5307	0.04	0.00	0.00	0.1	100	0.04	1.22	0.00	1.25	2.4
10att48	47.92	5394	0.00	0.1	0.00	0.2	50	0.00	0.64	0.00	0.63	-1.59
11eil51	41.18	174	0.57	0.5	0.00	10	95	0.57	1.28	0.00	1.27	-0.79
11berlin52	50	4040	0.00	0.3	0.00	2.5	88	0.00	1.26	0.00	1.26	0.00
14st70	45.71	316	1.27	2.9	0.00	9.4	69.15	1.27	1.3	0.00	1.31	0.76
16eil76	34.21	209	0.00	3.3	0.00	9.4	64.89	0.00	1.31	0.00	1.32	0.76
16pr76	48.68	64925	0.1	2.9	0.00	10.7	72.9	0.1	1.31	0.00	1.32	0.76
20gr96	40.63	29440	0.85	6.5	0.00	20.2	67.82	0.85	1.36	0.00	1.37	0.73
20rat99	33.33	497	1.41	8.7	0.00	35.4	75.42	1.41	1.37	0.00	1.42	3.52

6 Disponible sur https://ece.uwaterloo.ca/~sl2smith/GLNS/

Instance	ρ (%)	***bks***	**GLKH**					**GLNS**				
			Instances réduites		**Instances complètes**		τ (%)	**Instances réduites**		**Instances complètes**		τ (%)
			δ (%)	**T (s)**	δ (%)	**T (s)**		δ (%)	**T (s)**	δ (%)	**T (s)**	
20kroA100	44	9711	0.00	5.7	0.00	17.8	67.98	0.00	1.36	0.00	1.37	0.73
20kroB100	42	10328	0.47	6.6	0.00	23.6	72.03	0.47	1.36	0.00	1.37	0.73
20kroC100	40	9554	1.28	17.5	0.00	14.6	-19.86	1.28	1.36	0.00	1.39	2.16
20kroD100	37	9450	1.28	13	0.00	20.5	36.59	1.28	1.36	0.00	1.39	2.16
20kroE100	42	9523	0.98	6.7	0.00	17	60.59	0.98	1.36	0.00	1.38	1.45
20rd100	45	3650	0.11	5	0.00	44.2	88.69	0.11	1.36	0.00	1.4	2.86
21eil101	38.61	249	1.2	5.2	0.00	16	67.5	1.2	1.37	0.00	1.39	1.44
21lin105	46.67	8213	0.41	4.4	0.00	25	82.4	0.41	1.36	0.00	1.39	2.16
22pr107	42.06	27898	0.01	26.3	0.00	22.3	-17.94	0.01	1.38	0.00	1.43	3.5
25pr124	45.97	36605	0.00	39.5	0.00	68	41.91	0.00	1.46	0.00	1.55	5.81
26bier127	46.46	72418	0.16	10.9	0.00	51	78.63	0.16	1.44	0.00	1.47	2.04
26ch130	36.15	2828	0.07	11.7	0.00	47.5	75.37	0.07	1.47	0.00	1.51	2.65
28pr136	37.5	42570	0.36	25.3	0.00	53	52.26	0.36	1.48	0.00	1.51	1.99
28gr137	35.04	36417	0.26	20.7	0.00	52	60.19	0.26	1.47	0.00	1.46	-0.68
29pr144	52.08	45886	0.00	10.4	0.00	65	84	0.00	1.49	0.00	1.57	5.1
30ch150	32.67	2750	0.00	28	0.00	64.6	56.66	0.00	1.53	0.00	1.59	3.77
30kroA150	42	11018	0.11	19	0.00	44.1	56.92	0.11	1.56	0.00	1.59	1.89
30kroB150	40	12196	0.79	15.4	0.00	44.6	65.47	0.79	1.51	0.00	1.56	3.21
31pr152	32.24	51576	0.16	78.1	0.00	148.9	47.55	0.16	1.52	0.00	1.59	4.4
32u159	45.28	22664	0.06	16.5	0.00	85.9	80.79	0.06	1.56	0.00	1.63	4.29
39rat195	33.33	854	0.47	50	0.00	132.4	62.24	0.47	1.78	0.00	1.9	6.32
40d198	42.42	10557	0.21	236.2	0.00	477.8	50.57	0.21	1.84	0.00	1.88	2.13
40kroa200	39	13406	0.16	32.1	0.00	85.5	62.46	0.16	1.78	0.00	1.86	4.3
40krob200	36	13111	0.32	39.8	0.00	108.2	63.22	0.32	1.82	0.00	1.9	4.21
41gr202	38.12	23301	0.05	42.9	0.00	170	74.76	0.05	1.72	0.00	1.82	5.49

Instance	ρ (%)	***bks***	**GLKH**					**GLNS**				
			Instances réduites		**Instances complètes**		τ (%)	**Instances réduites**		**Instances complètes**		τ (%)
			δ (%)	**T (s)**	δ (%)	**T (s)**		δ (%)	**T (s)**	δ (%)	**T (s)**	
45ts225	50.22	68340	0.00	48.3	0.00	200.6	75.92	0.00	2.07	0.00	2.39	13.39
46gr229	39.3	71972	0.04	43.1	0.00	193	77.67	0.04	2.01	0.00	2.03	0.99
53gil262	33.59	1013	0.2	110.9	0.00	249.5	55.55	0.2	2.31	0.00	2.52	8.33
56a280	39.64	1079	0.19	125.8	0.00	351.5	64.21	0.19	2.63	0.00	2.63	0.00
87gr431	42.69	101946	0.14	110.1	0.00	598.8	81.61	0.14	6.09	0.00	6.86	11.22
88pr439	40.09	60099	0.23	287.4	0.00	1219.2	76.43	0.23	6.56	0.00	7.23	9.27
89pcb442	41.86	21657	0.00	190.7	0.00	745.6	74.42	0.00	7.64	0.00	7.55	-1.19
99d493	39.15	20023	0.12	340.9	0.00	1281.9	73.41	0.15	9.72	0.00	11.56	15.92
107att532	33.67	13464	0.27	35.9	0.00	104.7	65.71	0.27	8.21	0.00	9.14	10.18
107ali535	43.36	128639	0.00	331.5	0.00	1541.5	78.49	0.00	11.55	0.00	13.26	12.9
115rat575	28.52	2388	0.17	595	0.00	1355.4	56.1	0.67	15.4	0.46	17.1	9.94
115u574	33.8	16689	0.05	483.5	0.00	1530.7	68.41	0.05	13.24	0.00	12.76	-3.76
131p654	50.15	27428	0.01	2850.8	0.00	7430.7	61.63	0.01	13.2	0.00	16.72	21.05
132d657	31.51	22498	0.2	1103.9	0.00	2788	60.41	0.3	20.24	0.00	24.8	18.39
134gr666	34.98	163028	0.01	659.8	0.00	2213.3	70.19	0.01	20.48	0.82	24.22	15.44
145u724	32.18	17272	0.01	837.2	0.00	2252.6	62.83	0.27	26.22	0.16	39.5	33.62
157rat783	26.69	3262	0.21	1223.7	0.04	2701	54.69	0.31	38.92	0.43	47.18	17.51
200dsj1000	22.22	9187884	0.23	466.4	0.05	838.3	44.36	0.28	61.21	0.36	72.2	15.22
201pr1002	31.24	114311	0.05	2038.9	0.00	5810.8	64.91	0.13	74.81	0.07	97.23	23.06
212u1060	31.6	106007	0.12	2158.5	0.02	5928.1	63.59	0.39	109.6	0.05	116.05	5.56
217vm1084	46.22	130704	0.07	1330.3	0.00	6911.9	80.75	0.3	102.05	0.22	112.23	9.07
Moyenne			0.29	283.59	0.00	846.39	64.46	0.32	10.60	0.05	12.24	5.73

Tableau 4.5: Résolution des instances de la librairie MOM avec les solveurs GLKH et GLNS

Instance	ρ (%)	***bks***	**GLKH**					**GLNS**				
			Instances réduites		**Instances complètes**		τ (%)	**Instances réduites**		**Instances complètes**		τ (%)
			δ (%)	**T (s)**	δ (%)	**T (s)**		δ (%)	**T (s)**	δ (%)	**T (s)**	
2lin105-2x1	98.1	126	-	0.00	0.00	0.1	100	0.00	0.43	0.00	0.55	21.82
4berlin52-2x2	88.46	534	0.00	0.00	0.00	0.1	100	0.00	0.52	0.00	0.56	7.14
4eil51-2x2	80.39	37	0.00	0.00	0.00	0.00	-	0.00	0.52	0.00	0.61	14.75
4eil76-2x2	90.79	27	0.00	0.00	0.00	0.1	100	0.00	0.53	0.00	0.62	14.52
4i200h	94	444	0.00	0.00	0.00	0.3	100	0.00	0.49	0.00	1.11	55.86
4i200x1	94	336	0.00	0.00	0.00	0.3	100	0.00	0.52	0.00	1.1	52.73
4i200x2	94	336	0.00	0.00	0.00	0.3	100	0.00	0.52	0.00	1.1	52.73
4i200z	94	336	0.00	0.00	0.00	0.3	100	0.00	0.52	0.00	1.12	53.57
4i400h	97.5	161	0.00	0.00	0.00	0.9	100	0.00	0.5	0.00	4.48	88.84
4i400x1	97.75	160	0.00	0.00	0.00	1.1	100	0.00	0.52	0.00	4.38	88.13
4i400x2	97.75	160	0.00	0.00	0.00	1.2	100	0.00	0.5	0.00	4.35	88.51
4i400z	97.75	160	0.00	0.00	0.00	1.2	100	0.00	0.5	0.00	4.34	88.48
4pr76-2x2	86.84	4552	0.00	0.00	0.00	0.1	100	0.00	0.52	0.00	0.62	16.13
5eil51	66.67	75	0.00	0.00	0.00	0.1	100	0.00	0.54	0.00	0.61	11.48
5eil76	78.95	74	0.00	0.00	0.00	0.3	100	0.00	0.54	0.00	0.64	15.63
5i120-46	90	1042	0.00	0.00	0.00	0.3	100	0.00	0.54	0.00	0.79	31.65
5i45-18	64.44	939	0.00	0.00	0.00	0.1	100	0.00	0.54	0.00	0.62	12.9
5st70	81.43	103	0.00	0.00	0.00	0.1	100	0.00	0.55	0.00	0.65	15.38
6i300	92.67	270	0.00	0.00	0.00	5.4	100	0.00	0.59	0.00	1.69	65.09
6i350	93.71	247	0.00	0.00	0.00	4.6	100	0.00	0.61	0.00	2.54	75.98
6i400	94.75	262	0.00	0.00	0.00	7.6	100	0.00	0.61	0.00	2.75	77.82
6i450	94.44	257	0.00	0.00	0.00	7.2	100	0.00	0.61	0.00	4.11	85.16
6i500	94.4	245	0.00	0.1	0.00	10.8	99.07	0.00	0.6	0.00	4.57	86.87

Instance	ρ (%)	***bks***	**GLKH**					**GLNS**				
			Instances réduites		**Instances complètes**		τ (%)	**Instances réduites**		**Instances complètes**		τ (%)
			δ (%)	**T (s)**	δ (%)	**T (s)**		δ (%)	**T (s)**	δ (%)	**T (s)**	
6pr76-2x3	76.32	15729	0.00	0.00	0.00	0.1	100	0.00	0.61	0.00	0.66	7.58
7i30-17	33.33	1743	0.00	0.00	0.00	0.00	-	0.00	0.61	0.00	0.61	0.00
7i45-18	53.33	1289	0.00	0.00	0.00	0.3	100	0.00	0.62	0.00	0.63	1.59
7i65-21	66.15	1674	0.00	0.00	0.00	0.3	100	0.00	0.6	0.00	0.62	3.23
8i1000z	96	1224	0.00	0.3	0.00	58.5	99.49	0.00	0.63	0.00	49.62	98.73
9eil76-3x3	59.21	120	0.00	0.3	0.00	0.5	40	0.00	0.62	0.00	0.66	6.06
9st70-3x3	58.57	188	0.00	0.2	0.00	0.4	50	0.00	0.62	0.00	0.65	4.62
10berlin52	44.23	3223	0.00	0.1	0.00	0.3	66.67	0.00	0.63	0.00	0.64	1.56
10C1k.1	95.2	2459554	1.9	0.3	0.00	71.4	99.58	1.9	0.69	0.00	113.76	99.39
10eil51	35.29	145	0.00	0.3	0.00	0.3	0.00	0.00	0.63	0.00	0.65	3.08
10eil76	55.26	127	0.00	0.1	0.00	0.7	85.71	0.00	0.64	0.00	0.65	1.54
10i1500-503	97.4	1535	0.26	0.8	0.00	131.7	99.39	0.26	0.63	0.00	407.4	99.85
10i2000h	97.3	1282	0.00	1.3	0.00	267.3	99.51	0.00	0.64	0.00	332.83	99.81
10i2000x1	97.15	1296	0.00	1.8	0.00	328.5	99.45	0.00	0.66	0.00	331.83	99.8
10i2000x2	97	1301	0.00	1.1	0.00	367.3	99.7	0.00	0.66	0.00	326.49	99.8
10i2000z	97	1301	0.00	1.2	0.00	362.7	99.67	0.00	0.64	0.00	356.67	99.82
10i30-17	23.33	1528	0.00	0.00	0.00	0.1	100	0.00	0.64	0.00	0.64	0.00
10i300-109	83.67	1357	1.47	0.5	0.00	24.3	97.94	1.47	0.66	0.00	2.35	71.91
10i65-21	47.69	1756	0.00	0.2	0.00	0.4	50	0.00	0.65	0.00	0.67	2.99
10i70-21	50	1784	0.00	0.2	0.00	0.5	60	0.00	0.63	0.00	0.67	5.97
12eil51-3x4	27.45	163	0.00	0.2	0.00	0.2	0.00	0.00	0.65	0.00	0.66	1.52
12eil76-3x4	50	152	0.00	0.2	0.00	0.5	60	0.00	0.74	0.00	0.74	0.00
12st70-3x4	51.43	209	0.00	0.1	0.00	0.4	75	0.00	0.66	0.00	0.74	10.81
15eil51	19.61	201	0.00	0.2	0.00	0.3	33.33	0.00	0.74	0.00	0.74	0.00
16eil51-4x4	19.61	176	0.00	0.1	0.00	0.3	66.67	0.00	0.76	0.00	0.77	1.3

Instance	ρ (%)	***bks***	**GLKH**					**GLNS**				
			Instances réduites		**Instances complètes**		τ (%)	**Instances réduites**		**Instances complètes**		τ (%)
			δ (%)	**T (s)**	δ (%)	**T (s)**		δ (%)	**T (s)**	δ (%)	**T (s)**	
16lin105-4x4	53.33	5959	0.00	0.6	0.00	3.2	81.25	0.00	0.76	0.00	0.8	5
16st70-4x4	35.71	256	0.00	0.3	0.00	0.4	25	0.00	0.76	0.00	0.78	2.56
20eil51-4x5	11.76	219	0.00	0.2	0.00	0.2	0.00	0.00	0.85	0.00	0.86	1.16
20i1500-504	93	2741	0.33	2.1	0.00	431.5	99.51	0.33	0.88	0.00	219.48	99.6
20i2000-602	94.4	3101	1	2.6	0.00	539.3	99.52	1	0.92	0.00	697.27	99.87
20i2400h	93.08	1686	0.12	42.1	0.00	1222.3	96.56	0.12	1.03	0.00	348.32	99.7
20i2400x1	93.08	1705	0.12	43.6	0.00	1189.4	96.33	0.12	0.99	0.00	336.91	99.71
20i2400x2	93.04	1705	0.12	42	0.00	1185.7	96.46	0.12	1.01	0.00	344.99	99.71
20i2400z	93.75	1722	0.06	30.4	0.00	1245.3	97.56	0.06	0.97	0.00	348.67	99.72
20i2500-706	95.52	3462	0.64	5.1	0.00	961.5	99.47	0.64	0.9	0.00	1602.07	99.94
20i3000-801	96.2	3626	0.22	3.6	0.00	1118.9	99.68	0.22	1.07	0.00	3022.21	99.96
20i3000h	94.87	1710	0.00	39.8	0.00	1583.5	97.49	0.00	1.12	0.00	997.27	99.89
20i3000x1	94.77	1671	0.00	50.1	0.00	1690	97.04	0.00	0.94	0.00	836.74	99.89
20i3000x2	94.73	1671	0.00	47.9	0.00	1701.4	97.18	0.00	0.98	0.00	801.65	99.88
20i3000z	94.3	1661	0.18	51.2	0.00	1538.2	96.67	0.18	1.03	0.00	868.1	99.88
20i700	81.29	500	0.2	3.7	0.00	136.7	97.29	0.2	0.92	0.00	11.03	91.66
20st70-4x5	27.14	296	0.00	0.3	0.00	0.4	25	0.00	0.85	0.00	0.87	2.3
25eil76-5x5	17.11	255	0.00	0.3	0.00	0.4	25	0.00	0.94	0.00	0.95	1.05
25i850	79.65	520	0.38	6.8	0.00	201.1	96.62	0.38	1.07	0.00	11.11	90.37
25i900	80.11	517	0.39	8.8	0.00	228.3	96.15	0.39	1.06	0.00	12.33	91.4
25lin105	35.24	8223	0.00	0.9	0.00	2.8	67.86	0.00	0.98	0.00	0.99	1.01
25pcb442-5x5	73.3	10063	0.00	3.3	0.00	75.9	95.65	0.00	1	0.00	2.62	61.83
25rat99	19.19	509	0.00	0.8	0.00	1	20	0.00	0.94	0.00	0.95	1.05

Instance	ρ (%)	***bks***	**GLKH**					**GLNS**				
			Instances réduites		**Instances complètes**		τ (%)	**Instances réduites**		**Instances complètes**		τ (%)
			δ (%)	**T (s)**	δ (%)	**T (s)**		δ (%)	**T (s)**	δ (%)	**T (s)**	
30i1000	77.9	552	0.18	15.4	0.00	277.6	94.45	0.18	1.23	0.00	11.46	89.27
30i950	77.37	557	0.00	13.3	0.00	266.5	95.01	0.00	1.22	0.00	10.36	88.22
36eil101-6x6	8.91	302	0.00	1.2	0.00	1.3	7.69	0.00	1.16	0.00	1.17	0.85
42rat99-6x7	6.06	679	0.00	1.4	0.00	1.3	-7.69	0.00	1.3	0.00	1.32	1.52
49vm1084-7x7	74.82	52350	0.7	34	0.04	665.7	94.89	0.7	1.86	0.00	4.96	62.5
50eil101	5.94	396	0.00	1.4	0.00	1.3	-7.69	0.00	1.66	0.00	1.72	3.49
50i2000-603	83.75	4325	0.6	50.8	0.18	1903.3	97.33	0.6	2.27	0.00	234.54	99.03
50i2500-707	85.76	3961	0.45	83.7	0.61	2724.4	96.93	0.45	2.31	0.00	735.41	99.69
50nrw1379	72.37	7449	0.82	61.5	0.00	1068.9	94.25	0.79	3.07	0.00	56.71	94.59
moyenne			0.13	8.24	0.01	295.34	81.41	0.13	0.83	0.00	168.76	50.35

Alors que les deux solveurs ont pu trouver la *bks* pour presque toutes les instances étudiées dans les tableaux 4.4 et 4.5 lors de l'utilisation des librairies originales, les résultats semblent assez attrayants en utilisant les instances réduites par NN2C. Des 57 instances étudiées de la GTSPLIB, 51 sont résolues avec un écart ne dépassant pas 1%, 47 avec un écart inférieur à 0.5% (46 pour le solveur GLNS), tandis que 10 atteignent la bks. Le solveur GLNS a même obtenu des solutions meilleures en utilisant les instances réduites sur 3 instances (*134gr666*, *157rat783* et *200dsj1000*). Le temps d'exécution des solveurs a lui aussi baissé considérablement: les instances GTSPLIB ont été résolues avec un gain de 5% de temps pour le solveur GLNS et 65% pour GLKH.

La résolution des instances réduites de la librairie MOM a été plus remarquable: Les deux solveurs ont obtenu des résultats similaires et ont réussi à atteindre la bks pour 60 instances sur les 80 étudiées, tandis que 17 instances ont été résolues avec un écart inférieur à 1%. Ces bonnes solutions ont été obtenues avec un temps d'exécution inférieur à 50% par rapport aux instances complètes lors de l'application du solveur GLNS et 81% inférieur lors de l'exécution de GLKH.

L'exécution des solveurs s'est faite sans définir le critère d'arrêt qui consiste à atteindre la bks. En effet, la solution optimale n'est pas donnée et n'est pas facile à déterminer dans des applications réelles. Obtenir des solutions proche de l'optimalité dans un délai raisonnable peut s'avérer plus rentable que de chercher la solution optimale souvent inconnue.

4.4.4Résolution des instances en budget de temps limité

De ce fait, nous exécutons les deux solveurs en définissant un budget de temps fixe afin d'inspecter les performances des instances réduites dans de tels environnements. 20 instances larges sont choisies de la GTSPLIB et exécutées respectivement avec 10, 20 et 100 secondes comme durée d'exécution maximale. Chaque ligne du tableau 4.6 ci-dessous expose, pour une instance donnée, complète(comp) et réduite(réd), l'écart de la *bks* en utilisant les solveurs GLKH et GLNS avec le budget de temps donné.

Tableau 4.6: Résolution des instances GTSPLIB en budget de temps limité

Instance	**GLKH**						**GLNS**					
	10 sec		**20 sec**		**100 sec**		**10 sec**		**20 sec**		**100 sec**	
	comp	**réd**	**comp**	**réd**	**comp**	**réd**	**comp**	**réd**	**comp**	**réd**	**comp**	**réd**
287u1432	3.86	**3.67**	3.75	**2.97**	**1.71**	1.98	**4.37**	4.93	**4.02**	4.48	1.84	**0.88**
316fl1577	1.09	**0.99**	0.86	**0.75**	0.52	**0.16**	2.18	**1.87**	2.04	**1.87**	0.15	**0.08**
331d1655	3.36	**3.18**	2.74	**2.71**	**0.85**	1.02	3.51	**3.4**	**3.23**	3.39	0.92	**0.21**
350vm1748	1.95	**1.82**	1.93	**1.22**	0.61	**0.41**	4.25	**3.9**	**3.93**	**3.93**	**0.62**	1.01
364u1817	3.04	**2.85**	**2.27**	2.3	1.14	**1.01**	**5.38**	5.51	**4.63**	4.72	1.87	**1.14**
378rl1889	2.83	**2.58**	2.44	**2.03**	1.06	**0.79**	**4.68**	4.81	**4.05**	4.7	3.77	**2.08**

Instance	GLKH						GLNS					
	10 sec		20 sec		100 sec		10 sec		20 sec		100 sec	
	comp	réd	comp	réd	comp	réd	comp	réd	comp	réd	comp	réd
421d2103	3.38	**2.95**	**2.98**	2.99	2.06	**1.77**	**4.6**	4.93	**4.43**	4.46	4.48	**3.73**
431u2152	4.09	**4.02**	3.48	**3.37**	1.9	**1.6**	5.7	**5.61**	5.53	**5.44**	5.47	**4.79**
464u2319	6.22	**5.78**	**5.69**	5.76	4.41	**3.89**	**6.17**	6.54	**6.15**	6.34	5.74	**5.09**
479pr2392	4.04	**3.5**	3.52	**3.25**	2.36	**1.94**	5.85	**5.81**	5.33	**5.28**	**4.98**	5.25
608pcb3038	5.39	**4.92**	5.13	**4.29**	3.41	**3.3**	**7.06**	7.09	**6.64**	6.79	**5.76**	5.87
759fl3795	**2.16**	2.18	**1.87**	2.31	1.82	**1.37**	5.31	**4.61**	**3.84**	3.92	**2.61**	4.18
893fnl4461	6.53	**6.34**	**5.98**	6.11	4.78	**4.56**	**7.91**	8.41	**7.68**	8.11	7.48	**6.43**
1183rl5915	4.8	**4.3**	4.27	**4.19**	4	**3.24**	**8.98**	9.18	**8.46**	8.7	**7.72**	8.47
1187rl5934	4.4	**4.3**	4.24	**4.12**	3.48	**3.26**	**8.41**	8.58	**8.31**	8.86	7.96	**7.68**
2370rl11849	5.64	**5.18**	5.36	**5.26**	4.64	**4.38**	12.7	**11.93**	11.4	**11.15**	10.38	**10.11**
2702usa13509	3.67	**3.33**	3.66	**3.2**	3.34	**2.65**	11.85	**9.99**	9.87	**9.86**	**9**	9.58
2811brd14051	5.66	**5.6**	5.87	**5.47**	5.39	**4.94**	13.51	**11.6**	**9.95**	10.1	9.73	**9.38**
3023d15112	5.42	**4.9**	5.24	**4.73**	4.73	**4.4**	18.3	**11.4**	10.42	**10.03**	**9.31**	9.84
3703d18512	6.4	**6.26**	6.42	**6.01**	5.83	**5.51**	19.01	20.13	19.22	**12.01**	10.32	**10.3**
moyenne	4.20	**3.93**	3.89	**3.65**	2.90	**2.61**	7.99	**7.51**	6.96	**6.71**	5.51	**5.30**

Dans le tableau 4.6 ci-dessus, nous observons clairement que les instances réduites obtenues par NN2C offrent de meilleures solutions lorsque le temps d'exécution est limité. Dans la majorité des cas, les solutions produites par les instances réduites sont plus proches des solutions optimales que celles fournies par les instances complètes, ce qui rend les instances proposées utiles dans de tels environnements.

4.4.5k-NN2C: Une généralisation de l'approche NN2C

Au lieu de ne sélectionner que les premiers nœuds les plus proches pour chaque couple de clusters, NN2C peut être généralisée pour sélectionner les k (2,3,..)

plus proches de chaque paire. Cette généralisation, intitulé k-NN2C, est supposée apporter différents taux de réduction en fonction de la valeur de k. En effet, plus la valeur de k est grande, moins est le taux de réduction, étant donné que plus de nœuds seront sélectionnés. De l'autre côté, les nouveaux nœuds sélectionnés pourraient aider à obtenir des solutions meilleures.

Les tableaux 4.7 et 4.8 ci-dessous présentent les taux de réduction obtenus pour k=1,2,3 et 4 respectivement pour les librairies GTSPLIB et MOM.

Tableau 4.7: Taux de réduction des instances GTSPLIB par k-NN2C

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
3burma14	14	3	9	64.29	7	50	6	42.86	4	28.57
4ulysses16	16	4	10	62.5	6	37.5	5	31.25	4	25
5ulysses22	22	5	12	54.55	8	36.36	7	31.82	4	18.18
10att48	48	10	23	47.92	17	35.42	12	25	10	20.83
11eil51	51	11	15	29.41	8	15.69	5	9.8	3	5.88
11berlin52	52	11	25	48.08	13	25	11	21.15	9	17.31
14st70	70	14	28	40	13	18.57	3	4.29	2	2.86
16eil76	76	16	27	35.53	11	14.47	6	7.89	2	2.63
16pr76	76	16	27	35.53	16	21.05	7	9.21	1	1.32
20gr96	96	20	39	40.63	23	23.96	9	9.38	5	5.21
20rat99	99	20	30	30.3	13	13.13	7	7.07	4	4.04
20kroA100	100	20	35	35	19	19	10	10	4	4
20kroB100	100	20	30	30	14	14	8	8	6	6
20kroC100	100	20	42	42	19	19	8	8	4	4
20kroD100	100	20	40	40	19	19	12	12	7	7
20kroE100	100	20	41	41	22	22	13	13	7	7
20rd100	100	20	34	34	17	17	9	9	5	5
21eil101	101	21	37	36.63	21	20.79	8	7.92	4	3.96

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
21lin105	105	21	45	42.86	27	25.71	12	11.43	9	8.57
22pr107	107	22	32	29.91	13	12.15	10	9.35	5	4.67
25pr124	124	25	47	37.9	26	20.97	18	14.52	11	8.87
26bier127	127	26	53	41.73	33	25.98	21	16.54	11	8.66
26ch130	130	26	43	33.08	21	16.15	7	5.38	4	3.08
28pr136	136	28	40	29.41	23	16.91	11	8.09	2	1.47
28gr137	137	28	49	35.77	21	15.33	5	3.65	3	2.19
29pr144	144	29	68	47.22	32	22.22	15	10.42	7	4.86
30ch150	150	30	46	30.67	24	16	11	7.33	6	4
30kroA150	150	30	47	31.33	26	17.33	14	9.33	5	3.33
30kroB150	150	30	56	37.33	20	13.33	12	8	7	4.67
31pr152	152	31	58	38.16	28	18.42	20	13.16	11	7.24
32u159	159	32	58	36.48	33	20.75	21	13.21	13	8.18
39rat195	195	39	55	28.21	21	10.77	7	3.59	2	1.03
40d198	198	40	67	33.84	29	14.65	15	7.58	8	4.04
40kroa200	200	40	67	33.5	32	16	15	7.5	6	3
40krob200	200	40	58	29	25	12.5	13	6.5	3	1.5
41gr202	202	41	65	32.18	31	15.35	17	8.42	8	3.96
45ts225	225	45	105	46.67	35	15.56	20	8.89	0	0
46gr229	229	46	79	34.5	47	20.52	28	12.23	22	9.61
49usa1097	1097	49	800	72.93	670	61.08	582	53.05	518	47.22
53gil262	262	53	74	28.24	30	11.45	7	2.67	0	0
56a280	280	56	66	23.57	25	8.93	13	4.64	6	2.14
87gr431	431	87	183	42.46	122	28.31	91	21.11	68	15.78
88pr439	439	88	146	33.26	70	15.95	33	7.52	18	4.1
89pcb442	442	89	109	24.66	35	7.92	17	3.85	12	2.71
99d493	493	99	166	33.67	96	19.47	69	14	52	10.55

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
107ali535	535	107	219	40.93	145	27.1	109	20.37	82	15.33
107att532	532	107	180	33.83	94	17.67	52	9.77	33	6.2
115rat575	575	115	133	23.13	28	4.87	5	0.87	2	0.35
115u574	574	115	144	25.09	46	8.01	19	3.31	3	0.52
131p654	654	131	308	47.09	200	30.58	151	23.09	129	19.72
132d657	657	132	173	26.33	55	8.37	23	3.5	8	1.22
134gr666	666	134	229	34.38	125	18.77	102	15.32	77	11.56
145u724	724	145	182	25.14	72	9.94	32	4.42	11	1.52
157rat783	783	157	174	22.22	43	5.49	7	0.89	1	0.13
200dsj1000	1000	200	277	27.7	118	11.8	45	4.5	13	1.3
201pr1002	1002	201	253	25.25	102	10.18	39	3.89	13	1.3
212u1060	1060	212	288	27.17	121	11.42	64	6.04	44	4.15
217vm1084	1084	217	395	36.44	250	23.06	180	16.61	129	11.9
235pcb1173	1173	235	242	20.63	85	7.25	29	2.47	6	0.51
259d1291	1291	259	196	15.18	87	6.74	54	4.18	34	2.63
261rl1304	1304	261	266	20.4	70	5.37	17	1.3	4	0.31
265rl1323	1323	265	278	21.01	81	6.12	25	1.89	8	0.6
276nrw1379	1379	276	290	21.03	92	6.67	27	1.96	4	0.29
280fl1400	1400	280	573	40.93	398	28.43	271	19.36	210	15
287u1432	1432	287	208	14.53	67	4.68	19	1.33	4	0.28
316fl1577	1577	316	333	21.12	168	10.65	99	6.28	63	3.99
331d1655	1655	331	262	15.83	82	4.95	23	1.39	9	0.54
350vm1748	1748	350	654	37.41	398	22.77	262	14.99	168	9.61
364u1817	1817	364	173	9.52	56	3.08	27	1.49	8	0.44
378rl1889	1889	378	415	21.97	158	8.36	58	3.07	24	1.27
421d2103	2103	421	205	9.75	38	1.81	8	0.38	3	0.14
431u2152	2152	431	213	9.9	76	3.53	32	1.49	9	0.42

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
464u2319	2319	464	331	14.27	123	5.3	27	1.16	2	0.09
479pr2392	2392	479	489	20.44	168	7.02	44	1.84	10	0.42
608pcb3038	3038	608	539	17.74	168	5.53	74	2.44	22	0.72
759fl3795	3795	759	1008	26.56	403	10.62	240	6.32	159	4.19
893fnl4461	4461	893	814	18.25	181	4.06	32	0.72	7	0.16
1183rl5915	5915	1183	1067	18.04	221	3.74	34	0.57	3	0.05
1187rl5934	5934	1187	1047	17.64	252	4.25	49	0.83	7	0.12
1480pla7397	7397	1480	1862	25.17	1047	14.15	616	8.33	317	4.29
2000C10k.0	10000	2000	2917	29.17	1239	12.39	571	5.71	291	2.91
2000E10k.0	10000	2000	2039	20.39	474	4.74	99	0.99	22	0.22
2370rl11849	11849	2370	1761	14.86	426	3.6	86	0.73	13	0.11
2702usa13509	13509	2702	4263	31.56	2330	17.25	1518	11.24	1079	7.99
2811brd14051	14051	2811	3014	21.45	1137	8.09	506	3.6	259	1.84
3023d15112	15112	3023	3670	24.29	1600	10.59	725	4.8	327	2.16
3703d18512	18512	3703	3851	20.8	1390	7.51	628	3.39	319	1.72
4996sw24978	24978	4996	6417	25.69	2780	11.13	1279	5.12	583	2.33
moyenne			517.48	31.09	216.5	15.45	108.60	8.99	61.60	5.51

Tableau 4.8: Taux de réduction des instances de la librairie MOM par k-NN2C

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
2lin105-2x1	105	2	103	98.1	101	96.19	99	94.29	98	93.33
4berlin52-2x2	52	4	46	88.46	41	78.85	37	71.15	34	65.38
4eil51-2x2	51	4	41	80.39	37	72.55	33	64.71	31	60.78
4eil76-2x2	76	4	69	90.79	60	78.95	55	72.37	52	68.42

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
4i200a	200	4	188	94	177	88.5	166	83	158	79
4i200h	200	4	188	94	181	90.5	174	87	169	84.5
4i200x1	200	4	188	94	183	91.5	176	88	173	86.5
4i200x2	200	4	188	94	183	91.5	176	88	173	86.5
4i200z	200	4	188	94	183	91.5	177	88.5	173	86.5
4i400a	400	4	388	97	377	94.25	365	91.25	354	88.5
4i400h	400	4	390	97.5	384	96	374	93.5	369	92.25
4i400x1	400	4	391	97.75	385	96.25	380	95	374	93.5
4i400x2	400	4	391	97.75	385	96.25	380	95	374	93.5
4i400z	400	4	391	97.75	385	96.25	380	95	374	93.5
4pr76-2x2	76	4	66	86.84	60	78.95	54	71.05	50	65.79
5berlin52	52	5	38	73.08	30	57.69	24	46.15	21	40.38
5eil51	51	5	34	66.67	28	54.9	24	47.06	22	43.14
5eil76	76	5	60	78.95	54	71.05	48	63.16	46	60.53
5i120-46	120	5	108	90	99	82.5	90	75	81	67.5
5i300-108	300	5	286	95.33	277	92.33	270	90	264	88
5i30-17	30	5	18	60	13	43.33	9	30	9	30
5i400-205	400	5	386	96.5	376	94	369	92.25	363	90.75
5i45-18	45	5	29	64.44	26	57.78	20	44.44	15	33.33
5i500-304	500	5	484	96.8	474	94.8	465	93	456	91.2
5i60-21	60	5	48	80	43	71.67	36	60	35	58.33
5i65-21	65	5	53	81.54	47	72.31	41	63.08	39	60
5i70-21	70	5	58	82.86	51	72.86	46	65.71	43	61.43
5i75-22	75	5	62	82.67	55	73.33	48	64	42	56
5i90-33	90	5	75	83.33	68	75.56	62	68.89	57	63.33
5pr76	76	5	61	80.26	55	72.37	47	61.84	43	56.58
5st70	70	5	57	81.43	51	72.86	44	62.86	39	55.71

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
5ulysses22	22	5	12	54.55	8	36.36	7	31.82	4	18.18
6berlin52-2x3	52	6	37	71.15	31	59.62	23	44.23	21	40.38
6i300	300	6	278	92.67	267	89	259	86.33	252	84
6i350	350	6	328	93.71	317	90.57	307	87.71	299	85.43
6i400	400	6	379	94.75	366	91.5	357	89.25	351	87.75
6i450	450	6	425	94.44	416	92.44	406	90.22	397	88.22
6i500	500	6	472	94.4	461	92.2	453	90.6	449	89.8
6pr76-2x3	76	6	58	76.32	49	64.47	44	57.89	40	52.63
6st70-2x3	70	6	54	77.14	45	64.29	37	52.86	36	51.43
7i30-17	30	7	10	33.33	5	16.67	3	10	2	6.67
7i45-18	45	7	24	53.33	21	46.67	17	37.78	12	26.67
7i60-21	60	7	41	68.33	32	53.33	28	46.67	22	36.67
7i65-21	65	7	43	66.15	36	55.38	30	46.15	25	38.46
7i70-21	70	7	49	70	41	58.57	33	47.14	28	40
8berlin52-2x4	52	8	29	55.77	18	34.62	11	21.15	8	15.38
8i1000a	1000	8	944	94.4	895	89.5	847	84.7	795	79.5
8i1000h	1000	8	961	96.1	938	93.8	911	91.1	896	89.6
8i1000x1	1000	8	961	96.1	938	93.8	911	91.1	896	89.6
8i1000x2	1000	8	961	96.1	938	93.8	911	91.1	896	89.6
8i1000z	1000	8	960	96	937	93.7	922	92.2	906	90.6
8i600a	600	8	545	90.83	494	82.33	450	75	412	68.67
8i600h	600	8	564	94	543	90.5	523	87.17	507	84.5
8i600x1	600	8	561	93.5	539	89.83	527	87.83	508	84.67
8i600x2	600	8	561	93.5	539	89.83	527	87.83	508	84.67
8i600z	600	8	561	93.5	539	89.83	527	87.83	508	84.67
9a280-3x3	280	9	244	87.14	222	79.29	205	73.21	194	69.29
9eil101-3x3	101	9	67	66.34	54	53.47	45	44.55	38	37.62

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
9eil51-3x3	51	9	21	41.18	14	27.45	10	19.61	8	15.69
9eil76-3x3	76	9	45	59.21	35	46.05	27	35.53	25	32.89
9gil262-3x3	262	9	223	85.11	202	77.1	186	70.99	172	65.65
9lin318-3x3	318	9	276	86.79	261	82.08	239	75.16	222	69.81
9pcb442-3x3	442	9	403	91.18	374	84.62	349	78.96	337	76.24
9pr439-3x3	439	9	403	91.8	389	88.61	372	84.74	357	81.32
9pr76-3x3	76	9	46	60.53	37	48.68	33	43.42	27	35.53
9st70-3x3	70	9	41	58.57	33	47.14	27	38.57	22	31.43
10a280	280	10	236	84.29	208	74.29	192	68.57	182	65
10berlin52-2x5	52	10	25	48.08	18	34.62	13	25	9	17.31
10berlin52	52	10	23	44.23	12	23.08	8	15.38	5	9.62
10C1k.0	1000	10	962	96.2	936	93.6	917	91.7	898	89.8
10C1k.1	1000	10	952	95.2	924	92.4	897	89.7	875	87.5
10C1k.2	1000	10	958	95.8	931	93.1	907	90.7	890	89
10C1k.3	1000	10	957	95.7	938	93.8	919	91.9	900	90
10C1k.4	1000	10	957	95.7	934	93.4	913	91.3	896	89.6
10C1k.5	1000	10	957	95.7	932	93.2	910	91	895	89.5
10C1k.6	1000	10	958	95.8	933	93.3	909	90.9	894	89.4
10C1k.7	1000	10	955	95.5	924	92.4	904	90.4	881	88.1
10C1k.8	1000	10	960	96	936	93.6	919	91.9	905	90.5
10C1k.9	1000	10	961	96.1	937	93.7	916	91.6	898	89.8
10eil51	51	10	18	35.29	10	19.61	7	13.73	6	11.76
10eil76	76	10	42	55.26	29	38.16	25	32.89	22	28.95
10gil262	262	10	216	82.44	192	73.28	175	66.79	162	61.83
10i1000-407	1000	10	955	95.5	934	93.4	910	91	889	88.9
10i120-46	120	10	81	67.5	65	54.17	56	46.67	47	39.17
10i1400a	1400	10	1315	93.93	1235	88.21	1149	82.07	1081	77.21

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
10i1400h	1400	10	1344	96	1306	93.29	1276	91.14	1254	89.57
10i1400	1400	10	1344	96	1306	93.29	1276	91.14	1254	89.57
10i1400x1	1400	10	1344	96	1306	93.29	1276	91.14	1254	89.57
10i1400x2	1400	10	1345	96.07	1312	93.71	1282	91.57	1256	89.71
10i1400z	1400	10	1345	96.07	1312	93.71	1282	91.57	1257	89.79
10i1500-503	1500	10	1461	97.4	1431	95.4	1406	93.73	1385	92.33
10i2000a	2000	10	1911	95.55	1825	91.25	1742	87.1	1663	83.15
10i2000h	2000	10	1946	97.3	1906	95.3	1878	93.9	1855	92.75
10i2000x1	2000	10	1943	97.15	1903	95.15	1880	94	1850	92.5
10i2000x2	2000	10	1940	97	1905	95.25	1874	93.7	1851	92.55
10i2000z	2000	10	1940	97	1905	95.25	1875	93.75	1851	92.55
10i300-109	300	10	251	83.67	222	74	198	66	177	59
10i30-17	30	10	7	23.33	1	3.33	1	3.33	0	0
10i400-206	400	10	355	88.75	334	83.5	315	78.75	297	74.25
10i45-18	45	10	11	24.44	3	6.67	3	6.67	2	4.44
10i500-305	500	10	459	91.8	435	87	413	82.6	398	79.6
10i60-21	60	10	30	50	20	33.33	14	23.33	11	18.33
10i65-21	65	10	31	47.69	22	33.85	15	23.08	8	12.31
10i70-21	70	10	35	50	27	38.57	21	30	12	17.14
10i75-22	75	10	45	60	34	45.33	24	32	19	25.33
10i90-33	90	10	51	56.67	36	40	28	31.11	17	18.89
10kroB100	100	10	66	66	53	53	45	45	38	38
10lin318	318	10	282	88.68	258	81.13	247	77.67	236	74.21
10nrw1379-2x5	1379	10	1329	96.37	1294	93.84	1270	92.1	1250	90.65
10pcb442	442	10	396	89.59	364	82.35	339	76.7	329	74.43
10pr439	439	10	395	89.98	366	83.37	346	78.82	328	74.72
10pr76	76	10	43	56.58	35	46.05	28	36.84	21	27.63

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
10rat99	99	10	58	58.59	43	43.43	38	38.38	32	32.32
10st70	70	10	40	57.14	29	41.43	19	27.14	16	22.86
12eil51-3x4	51	12	14	27.45	9	17.65	4	7.84	3	5.88
12eil76-3x4	76	12	38	50	26	34.21	18	23.68	11	14.47
12nrw1379-2x6	1379	12	1303	94.49	1269	92.02	1235	89.56	1199	86.95
12nrw1379-3x4	1379	12	1306	94.71	1271	92.17	1242	90.07	1213	87.96
12pr76-3x4	76	12	40	52.63	26	34.21	19	25	16	21.05
12st70-3x4	70	12	36	51.43	25	35.71	17	24.29	14	20
15berlin52	52	15	14	26.92	5	9.62	3	5.77	1	1.92
15eil51	51	15	10	19.61	5	9.8	3	5.88	2	3.92
15eil76	76	15	27	35.53	16	21.05	12	15.79	4	5.26
15i300-110	300	15	223	74.33	182	60.67	157	52.33	142	47.33
15i400-207	400	15	331	82.75	298	74.5	262	65.5	233	58.25
15i500-306	500	15	429	85.8	390	78	365	73	334	66.8
15pr76-3x5	76	15	37	48.68	27	35.53	21	27.63	18	23.68
15pr76	76	15	25	32.89	17	22.37	10	13.16	5	6.58
15st70	70	15	27	38.57	16	22.86	10	14.29	8	11.43
16eil51-4x4	51	16	10	19.61	3	5.88	2	3.92	1	1.96
16eil76-4x4	76	16	24	31.58	8	10.53	5	6.58	2	2.63
16lin105-4x4	105	16	56	53.33	34	32.38	26	24.76	20	19.05
16st70-4x4	70	16	25	35.71	14	20	10	14.29	9	12.86
18pr439-3x6	439	18	364	82.92	326	74.26	305	69.48	281	64.01
18pr76-3x6	76	18	24	31.58	13	17.11	8	10.53	7	9.21
20eil51-4x5	51	20	6	11.76	2	3.92	2	3.92	2	3.92
20eil76-4x5	76	20	17	22.37	8	10.53	2	2.63	1	1.32
20i1000-408	1000	20	900	90	834	83.4	785	78.5	746	74.6
20i1500-504	1500	20	1395	93	1334	88.93	1269	84.6	1226	81.73

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
20i2000-602	2000	20	1888	94.4	1826	91.3	1778	88.9	1726	86.3
20i2400a	2400	20	2042	85.08	1739	72.46	1487	61.96	1274	53.08
20i2400h	2400	20	2234	93.08	2150	89.58	2082	86.75	2025	84.38
20i2400x1	2400	20	2234	93.08	2149	89.54	2091	87.13	2034	84.75
20i2400x2	2400	20	2233	93.04	2147	89.46	2090	87.08	2036	84.83
20i2400z	2400	20	2250	93.75	2158	89.92	2095	87.29	2039	84.96
20i2400z	2400	20	2250	93.75	2158	89.92	2095	87.29	2039	84.96
20i2500-706	2500	20	2388	95.52	2337	93.48	2273	90.92	2231	89.24
20i3000-801	3000	20	2886	96.2	2830	94.33	2783	92.77	2728	90.93
20i3000a	3000	20	2634	87.8	2327	77.57	2055	68.5	1812	60.4
20i3000h	3000	20	2846	94.87	2758	91.93	2689	89.63	2638	87.93
20i3000x1	3000	20	2843	94.77	2755	91.83	2691	89.7	2639	87.97
20i3000x2	3000	20	2842	94.73	2754	91.8	2692	89.73	2640	88
20i3000z	3000	20	2829	94.3	2746	91.53	2680	89.33	2635	87.83
20i300-111	300	20	198	66	156	52	118	39.33	97	32.33
20i400-208	400	20	295	73.75	247	61.75	215	53.75	181	45.25
20i500-307	500	20	388	77.6	330	66	278	55.6	242	48.4
20i550	550	20	426	77.45	380	69.09	355	64.55	331	60.18
20i600	600	20	475	79.17	428	71.33	393	65.5	371	61.83
20i650	650	20	525	80.77	476	73.23	439	67.54	416	64
20i700	700	20	569	81.29	517	73.86	473	67.57	455	65
20pr439-4x5	439	20	356	81.09	318	72.44	283	64.46	257	58.54
20st70-4x5	70	20	19	27.14	10	14.29	5	7.14	2	2.86
21lin105	105	21	45	42.86	27	25.71	12	11.43	9	8.57
25a280-5x5	280	25	172	61.43	123	43.93	99	35.36	86	30.71
25a280	280	25	160	57.14	108	38.57	84	30	66	23.57
25eil101-5x5	101	25	26	25.74	12	11.88	6	5.94	3	2.97

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
25eil101	101	25	19	18.81	9	8.91	3	2.97	2	1.98
25eil51-5x5	51	25	4	7.84	3	5.88	1	1.96	1	1.96
25eil76-5x5	76	25	13	17.11	5	6.58	1	1.32	1	1.32
25gil262-5x5	262	25	145	55.34	110	41.98	78	29.77	69	26.34
25gil262	262	25	145	55.34	99	37.79	68	25.95	52	19.85
25i300-112	300	25	165	55	118	39.33	82	27.33	61	20.33
25i400-209	400	25	267	66.75	210	52.5	170	42.5	138	34.5
25i500-308	500	25	355	71	280	56	222	44.4	186	37.2
25i750	750	25	595	79.33	519	69.2	480	64	449	59.87
25i800	800	25	640	80	571	71.38	528	66	498	62.25
25i850	850	25	677	79.65	616	72.47	576	67.76	549	64.59
25i900	900	25	721	80.11	657	73	619	68.78	590	65.56
25kroA100	100	25	27	27	12	12	6	6	5	5
25lin105	105	25	37	35.24	13	12.38	3	2.86	2	1.9
25lin318-5x5	318	25	209	65.72	152	47.8	125	39.31	106	33.33
25lin318	318	25	200	62.89	154	48.43	122	38.36	107	33.65
25pcb442-5x5	442	25	324	73.3	253	57.24	208	47.06	184	41.63
25pcb442	442	25	310	70.14	243	54.98	205	46.38	185	41.86
25pr439	439	25	309	70.39	250	56.95	199	45.33	170	38.72
25rat99-5x5	99	25	21	21.21	7	7.07	3	3.03	2	2.02
25rat99	99	25	19	19.19	7	7.07	4	4.04	2	2.02
28kroA100-4x7	100	28	30	30	7	7	2	2	0	0
30i1000	1000	30	779	77.9	704	70.4	657	65.7	625	62.5
30i950	950	30	735	77.37	665	70	618	65.05	582	61.26
30kroB100-5x6	100	30	21	21	10	10	4	4	4	4
35kroB100-5x5	100	25	31	31	12	12	9	9	4	4
36eil101-6x6	101	36	9	8.91	1	0.99	0	0	0	0

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
36pcb442-6x6	442	36	279	63.12	192	43.44	153	34.62	133	30.09
36pr1002-6x6	1002	36	773	77.15	675	67.37	602	60.08	540	53.89
42a280-6x7	280	42	107	38.21	50	17.86	35	12.5	23	8.21
42pr1002-6x7	1002	42	741	73.95	620	61.88	545	54.39	489	48.8
42rat99-6x7	99	42	6	6.06	2	2.02	0	0	0	0
49gil262-7x7	262	49	75	28.63	33	12.6	14	5.34	4	1.53
49lin318-7x7	318	49	135	42.45	67	21.07	45	14.15	35	11.01
49pcb1173-7x7	1173	49	838	71.44	682	58.14	589	50.21	507	43.22
49pr1002-7x7	1002	49	696	69.46	588	58.68	518	51.7	456	45.51
49rat783-7x7	783	49	482	61.56	356	45.47	287	36.65	239	30.52
49vm1084-7x7	1084	49	811	74.82	706	65.13	628	57.93	570	52.58
50a280	280	50	87	31.07	31	11.07	17	6.07	11	3.93
50eil101	101	50	6	5.94	1	0.99	1	0.99	1	0.99
50gil262	262	50	79	30.15	32	12.21	14	5.34	7	2.67
50i1000-409	1000	50	700	70	547	54.7	435	43.5	357	35.7
50i1500-505	1500	50	1173	78.2	1030	68.67	902	60.13	810	54
50i2000-603	2000	50	1675	83.75	1485	74.25	1337	66.85	1212	60.6
50i2500-707	2500	50	2144	85.76	1953	78.12	1793	71.72	1667	66.68
50i3000-802	3000	50	2657	88.57	2481	82.7	2330	77.67	2197	73.23
50kroA100	100	50	8	8	1	1	0	0	0	0
50kroB100	100	50	5	5	0	0	0	0	0	0
50lin105	105	50	10	9.52	0	0	0	0	0	0
50lin318	318	50	115	36.16	49	15.41	29	9.12	18	5.66
50nrw1379	1379	50	998	72.37	860	62.36	738	53.52	670	48.59
50pcb1173	1173	50	825	70.33	683	58.23	586	49.96	508	43.31
50pcb442	442	50	195	44.12	119	26.92	77	17.42	57	12.9
50pr1002	1002	50	690	68.86	568	56.69	472	47.11	409	40.82

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
50pr439	439	50	207	47.15	119	27.11	72	16.4	44	10.02
50rat783	783	50	478	61.05	352	44.96	278	35.5	223	28.48
50rat99	99	50	1	1.01	0	0	0	0	0	0
50vm1084	1084	50	799	73.71	665	61.35	569	52.49	508	46.86
72vm1084-8x9	1084	72	718	66.24	599	55.26	525	48.43	476	43.91
75lin105	105	75	3	2.86	0	0	0	0	0	0
81vm1084-9x9	1084	81	694	64.02	562	51.85	489	45.11	444	40.96
100i1000-410	1000	100	431	43.1	242	24.2	132	13.2	80	8
100i1500-506	1500	100	857	57.13	598	39.87	436	29.07	311	20.73
100i2000-604	2000	100	1335	66.75	1016	50.8	768	38.4	603	30.15
100i2500-708	2500	100	1797	71.88	1444	57.76	1178	47.12	972	38.88
100i3000-803	3000	100	2287	76.23	1920	64	1637	54.57	1406	46.87
100nrw1379	1379	100	722	52.36	532	38.58	396	28.72	301	21.83
100pcb1173	1173	100	559	47.66	345	29.41	235	20.03	162	13.81
100pr1002	1002	100	470	46.91	299	29.84	190	18.96	137	13.67
100prb1173-10x10	1173	100	583	49.7	359	30.61	244	20.8	161	13.73
100rat783-10x10	783	100	293	37.42	131	16.73	47	6	25	3.19
100rat783	783	100	292	37.29	128	16.35	66	8.43	32	4.09
100vm1084	1084	100	612	56.46	422	38.93	298	27.49	220	20.3
144pcb1173-12x12	1173	144	440	37.51	207	17.65	98	8.35	44	3.75
144rat783-12x12	783	144	170	21.71	43	5.49	16	2.04	4	0.51
150i1000-411	1000	150	273	27.3	101	10.1	39	3.9	19	1.9
150i1500-507	1500	150	624	41.6	318	21.2	177	11.8	98	6.53
150i2000-605	2000	150	1033	51.65	655	32.75	413	20.65	268	13.4
150i2500-709	2500	150	1489	59.56	1056	42.24	761	30.44	544	21.76
150i3000-804	3000	150	1958	65.27	1492	49.73	1122	37.4	873	29.1
150nrw1379	1379	150	527	38.22	303	21.97	167	12.11	100	7.25

Instance	**N**	**M**	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
			N supp.	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)	**N supp.**	ρ (%)
150pcb1173	1173	150	428	36.49	194	16.54	102	8.7	57	4.86
150pr1002	1002	150	330	32.93	140	13.97	74	7.39	33	3.29
150rat783	783	150	195	24.9	49	6.26	13	1.66	6	0.77
150vm1084	1084	150	462	42.62	244	22.51	131	12.08	75	6.92
200i2000-606	2000	200	784	39.2	393	19.65	196	9.8	103	5.15
200i2500-710	2500	200	1239	49.56	753	30.12	456	18.24	282	11.28
200i3000-805	3000	200	1656	55.2	1091	36.37	741	24.7	496	16.53
moyenne			564.93	67.04	501.25	56.76	459.17	50.50	429.44	46.38

Comme prévu, les taux de réduction diminuent en passant de 1-NN2C à 2-NN2C et ainsi de suite. Il est même réduit de moitié de 1-NN2C à 2-NN2C sur les instances de la GTSPLIB, ce qui devrait fournir de meilleures solutions.

Pour confirmer cette hypothèse, nous avons choisi 20 instances ayant les écarts les plus importants de leurs *bks* respectifs lors de l'application des solveurs sur les instances réduites de 1-NN2C (tableaux 4.4 et 4.5) et les avons exécutées après réduction avec les différentes variantes de k-NN2C décrites ci-dessus.

Tableau 4.9: Instances de la GTSPLIB réduites par k-NN2C et résolues par GLNS

Instance	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
	ρ (%)	δ (%)	ρ (%)	δ (%)	ρ (%)	δ (%)	ρ (%)	δ (%)
3burma14	64.29	1.05	50	1.05	42.86	1.05	28.57	0
11eil51	41.18	0.57	15.69	0	9.8	0	5.88	0
14st70	45.71	1.27	18.57	0.63	4.29	0	2.86	0
20gr96	40.63	0.85	23.96	0.01	9.38	0	5.21	0
20kroB100	42	0.47	14	0.45	8	0	6	0
20kroC100	40	1.28	19	0.09	8	0.09	4	0.09
20kroD100	37	1.28	19	0	12	0	7	0

Instance	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
	ρ **(%)**	δ **(%)**	ρ (%)	δ (%)	ρ (%)	δ (%)	ρ (%)	δ (%)
20kroE100	42	0.98	22	0.39	13	0	7	0
20rat99	33.33	1.41	13.13	0.4	7.07	0.4	4.04	0
21eil101	38.61	1.2	20.79	0.4	7.92	0	3.96	0
21lin105	46.67	0.41	25.71	0	11.43	0	8.57	0
28pr136	37.5	0.36	16.91	0.17	8.09	0.09	1.47	0
30kroB150	40	0.79	13.33	0.02	8	0.02	4.67	0
39rat195	33.33	0.47	10.77	0.12	3.59	0.12	1.03	0.12
40krob200	36	0.32	12.5	0	6.5	0	1.5	0
115rat575	28.52	0.67	4.87	0.04	0.87	0.46	0.35	0.46
132d657	31.51	0.3	8.37	0.28	3.5	0.01	1.22	0.01
157rat783	26.69	0.31	5.49	0.28	0.89	0.12	0.13	0.09
212u1060	31.6	0.39	11.42	0.21	6.04	0.04	4.15	0.15
217vm1084	46.22	0.3	23.06	0.22	16.61	0	11.9	0.22
moyenne	**39.14**	**0.73**	**17.43**	**0.24**	**9.39**	**0.12**	**5.48**	**0.06**

Tableau 4.10: Instances de la librairie MOM réduites par k-NN2C et résolues par GLNS

Instance	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
	ρ (%)	δ (%)	ρ (%)	δ (%)	ρ (%)	δ (%)	ρ (%)	δ (%)
10C1k.1	95.2	1.9	92.4	1.88	89.7	0.19	87.5	0.19
10i1500-503	97.4	0.26	95.4	0	93.73	0	92.33	0
10i300-109	83.67	1.47	74	0.52	66	0.44	59	0.22
20i1500-504	93	0.33	88.93	0.33	84.6	0.26	81.73	0.22
20i2000-602	94.4	1	91.3	0.71	88.9	0.13	86.3	0.13
20i2400h	93.08	0.12	89.58	0	86.75	0	84.38	0
20i2400x1	93.08	0.12	89.54	0	87.13	0	84.75	0
20i2400x2	93.04	0.12	89.46	0	87.08	0	84.83	0
20i2400z	93.75	0.06	89.92	0.06	87.29	0	84.96	0

Instance	**1-NN2C**		**2-NN2C**		**3-NN2C**		**4-NN2C**	
	ρ (%)	δ (%)	ρ (%)	δ (%)	ρ (%)	δ (%)	ρ (%)	δ (%)
20i2500-706	95.52	0.64	93.48	0.64	90.92	0.38	89.24	0.38
20i3000-801	96.2	0.22	94.33	0.17	92.77	0.17	90.93	0.17
20i3000z	94.3	0.18	91.53	0	89.33	0	87.83	0
20i700	81.29	0.2	73.86	0.2	67.57	0.2	65	0
25i850	79.65	0.38	72.47	0.19	67.76	0.19	64.59	0.19
25i900	80.11	0.39	73	0.19	68.78	0.19	65.56	0.19
30i1000	77.9	0.18	70.4	0	65.7	0	62.5	0
49vm1084-7x7	74.82	0.7	65.13	0.11	57.93	0.1	52.58	0.03
50i2000-603	83.75	0.6	74.25	0.3	66.85	0.25	60.6	0.23
50i2500-707	85.76	0.45	78.12	0.28	71.72	0.18	66.68	0.15
50nrw1379	72.37	0.79	62.36	0.23	53.52	0.04	48.59	0.04
moyenne	**87.91**	**0.51**	**82.47**	**0.29**	**78.20**	**0.14**	**74.99**	**0.11**

Les tableaux 4.9 et 4.10 montrent que les instances sont mieux résolues en conservant plus de nœuds. Les solutions proposées par **k**-NN2C sont presque toujours meilleures que celles proposées par **k**−**1**-NN2C. L'écart de la *bks* est réduit d'environ la moitié en passant de **k**-NN2C à **k**+**1**-NN2C.

4.5 Conclusion

La réduction de la dimension est un processus important qui aide les algorithmes à obtenir des solutions avec un temps d'exécution relativement court. La qualité des solutions peut être perturbée par cette réduction mais devrait rester pratiquement acceptable.

Nous avons proposé dans ce chapitre l'algorithme de réduction NN2C pour le GTSP. L'approche consiste à sélectionner les nœuds les plus proches des autres clusters et à supprimer les autres. Les résultats expérimentaux ont montré que la taille des instances est considérablement réduite et donne toujours de bons résultats dans un temps d'exécution raisonnable.

Une généralisation de NN2C a été présentée plus tard, elle vise à sélectionner les k nœuds les plus proches pour chaque paire de clusters, au lieu de sélectionner qu'un seul. Le taux de réduction diminue avec un k supérieur mais donne des solutions meilleures.

Les différents nœuds candidats obtenus pour chaque k peuvent être utilisés, dans un travail ultérieur, pour résoudre le GTSP ou tout autre problème réductible en mettant en place un espace de recherche variable. D'autres approches dans le même principe de NN2C pourraient être explorées plus tard, la sélection de nœud peut être effectuée par exemple par rapport à tous les nœuds de chaque cluster ou vis à vis de leurs barycentres [26].

Conclusion générale

L'objectif de cette thèse était de proposer des méthodes efficientes pour résoudre des POC dont les champs d'applications incluent le domaine des transports. Les bons résultats obtenus par BLS sur plusieurs POC étaient la principale motivation derrière l'adaptation de cette métaheuristique pour le TSP puis le GTSP.

Les instances du TSP étant plus difficiles par leurs topologies bien diversifiées, BLS n'aura pas le même succès atteint dans les travaux antérieurs. Malgré les améliorations proposées sur la métaheuristique, les solutions fournis resteront limitées par rapport aux solveurs les plus populaires. En terme de complexité temporelle, la stratégie de sélection de BLS et le nombre de sauts croissant font en sorte qu'elle requière un temps d'exécution assez conséquent. Soulevant ainsi deux contraintes majeures à la métaheuristique.

La proposition d'une version mémétique (pour le GTSP) basée sur BLS avait pour but de tirer profit des AG pour mieux explorer l'espace de recherche et augmenter les chances de tomber sur des optima locaux meilleurs. BLS étant très lente, l'appliquer à une population de solutions sur plusieurs générations la rendrait interminable. Une amélioration sur la complexité de la stratégie de sélection est proposée afin de rendre l'application de l'algorithme possible sur le GTSP. Les résultats finaux permettent de se comparer à l'état de l'art, mais restent tout de même atteignables dans un temps à la limite du raisonnable.

Le problème des instances larges face auxquelles a été confrontée BLS est bien connu dans l'optimisation combinatoire, mais dans une ampleur plus grande dans ce cas. La réductibilité du GTSP a été exploitée afin de rendre l'exécution de quelconque méthode plus rapide. Comme la plupart des approches de prétraitement destinées à réduire la dimension d'une instance, la réduction proposée ne garantit pas de conserver de la solution optimale dans l'espace de recherche mais offre des solutions qui lui sont proches dans un laps de temps beaucoup plus court que ce que peut nécessiter l'instance complète correspondante.

Que ce soit pour les instances complètes ou réduites, proposer une (méta)heuristique performante basée sur la recherche locale se doit de connaître les différentes topologies des espaces de recherches des instances difficiles. L'étude du paysage de recherche contribue amplement dans la proposition de nouvelles approches bien adaptées au problème. Elle permet de connaître les fonctions de voisinage et les opérateurs les plus appropriés pour anticipant les difficultés pour améliorer la/les solution(s) courante(s). Un intérêt pour les études de paysage sera d'avantage accordé dans les travaux postdoctoraux, une contribution dans ce sens a même déjà été apportée au *Travelling Thief Problem* [37].

Trouver la meilleure configuration pour certaines métaheuristiques disposantes de plusieurs paramètres de différents types (à l'image de BLS) pénalise souvent l'avancement du travail en question, sans pour autant avoir la garantie de tomber sur la plus convenable. La configuration automatique des algorithmes a émergé durant les deux dernières décennies en tant que problème d'optimisation. Plusieurs techniques ont depuis été proposées entraînant une motivation supplémentaire pour prendre en considération de telles méthodes dans les travaux futurs.

Travaux effectués

(1) M. El Krari, B. Ahiod, and B. El Benani, "An empirical study of the multi-fragment tour construction algorithm for the travelling salesman problem," in *Advances in Intelligent Systems and Computing*, vol. 552, 2017, pp. 278–287.

(2) M. El Krari, B. Ahiod, and B. El Benani, "Breakout Local Search for the Travelling Salesman Problem," *Computing and Informatics*, vol. 37, no. 3, pp. 656–672, 2018.

(3) M. El Krari, B. Ahiod, and B. El Benani, "A novel reduction algorithm for the generalized traveling salesman problem," *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17*, pp. 105–106, 2017.

(4) M. El Krari, B. Ahiod, and B. El Benani, "Using cluster barycenters for the generalized traveling salesman problem," in *Advances in Intelligent Systems and Computing*, vol. 557, 2017, pp. 135–143.

(5) M. El Yafrani, M. S. R. Martins, M. El Krari, M. Wagner, M. R. B. S. Delgado, B. Ahiod, and R. Lüders, "A fitness landscape analysis of the travelling thief problem," in *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '18*, 2018, pp. 277–284.

Bibliographie

[1] K. R. Overgaard, “Urban transportation planning’traffic estimation,” *Traffic Quarterly*, vol. 21, no. 2, 1967.

[2] P. A. Steenbrink, *Optimization of transport networks*. Wiley, 1974.

[3] K. J. Hensher David A.; Button, Ed., *Handbook of Transport Modelling*, 2nd ed. 2007.

[4] S. P. Hoogendoorn and P. H. L. Bovy, “State-of-the-art of vehicular traffic flow modelling,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 215, no. 4, pp. 283–303, Jun. 2001.

[5] G. B. Dantzig and J. H. Ramser, “The Truck Dispatching Problem,” *Management Science*, vol. 6, no. 1, pp. 80–91, Oct. 1959.

[6] B. L. Golden and R. T. Wong, “Capacitated arc routing problems,” *Networks*, vol. 11, no. 3, pp. 305–315, 1981.

[7] T. C. Koopmans, M. J. Beckmann, T. Koopmans, and M. J. Beckmann, “Assignment Problems and the Location of Economic Activities,” 1955.

[8] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, *The traveling salesman problem: a guided tour of combinatorial optimization*, vol. 3. Wiley New York, 1985.

[9] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*. Berlin, Heidelberg: Springer-Verlag, 1994.

[10] R. G. Bland and D. F. Shallcross, “Large travelling salesman problems arising from experiments in X-ray crystallography: A preliminary report on computation,” *Operations Research Letters*, vol. 8, no. 3, pp. 125–128, Jun. 1989.

[11] S. S. Srivastava, S. Kumar, R. C. Garg, and P. Sen, "Generalized travelling salesman problem through n sets of nodes," *CORS J.*, vol. 7, pp. 97–101, 1969.

[12] A. L. Henry-Labordere, "The record balancing problem: A dynamic programming solution of a generalized travelling salesman problem," *RIRO*, vol. B-2, pp. 43–49, 1969.

[13] R. F. da Silva and S. Urrutia, "A General VNS heuristic for the traveling salesman problem with time windows," *Discrete Optimization*, vol. 7, no. 4, pp. 203–211, Nov. 2010.

[14] H. N. Psaraftis, "Dynamic vehicle routing problems. BL Golden, AA Assad, eds," *Vehicle Routing: Methods and Studies*, vol. 16, pp. 223–248, 1988.

[15] G. Gutin and A. P. Punnen, Eds., *The Traveling Salesman Problem and Its Variations*, vol. 12. Boston, MA: Springer US, 2007.

[16] M. Dror and M. Haouari, "Generalized Steiner Problems and Other Variants," *Journal of Combinatorial Optimization*, vol. 4, no. 4, pp. 415–436, 2000.

[17] N. C.E, "The generalized traveling salesman problem," University of Michigan, 1988.

[18] G. Laporte, A. Asef-Vaziri, and C. Sriskandarajah, "Some applications of the generalized travelling salesman problem," *Journal of the Operational Research Society*, vol. 47, no. 12, pp. 1461–1467, 1996.

[19] J. P. Saksena, *Mathematical Model of Scheduling Clients Through Welfare Angencies: II*. Depts. of Electrical Engineering and Medicine, University of Southern California, 1967.

[20] V. Dimitrijević and Z. Šarić, "An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs," *Information Sciences*, vol. 102, no. 1–4, pp. 105–110, 1997.

[21] G. Laporte and F. Semet, "Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem," *INFOR:*

Information Systems and Operational Research, vol. 37, no. 2, pp. 114–120, 1999.

[22] Y.-N. Lien, E. Ma, and B. W.-S. Wah, "Transformation of the generalized traveling-salesman problem into the standard traveling-salesman problem," *Information Sciences*, vol. 74, no. 1–2, pp. 177–189, 1993.

[23] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993.

[24] B. Hu and G. R. Raidl, "Effective neighborhood structures for the generalized traveling salesman problem," in *Evolutionary Computation in Combinatorial Optimization*, Springer, 2008, pp. 36–47.

[25] M. Pourhassan and F. Neumann, "Theoretical Analysis of Local Search and Simple Evolutionary Algorithms for the Generalized Travelling Salesperson Problem," *Evolutionary Computation*, pp. 1–34, Jun. 2018.

[26] M. El Krari, B. Ahiod, and B. El Benani, "Using cluster barycenters for the generalized traveling salesman problem," in *Advances in Intelligent Systems and Computing*, vol. 557, 2017, pp. 135–143.

[27] J. A. Chisman, "The clustered traveling salesman problem," *Computers & Operations Research*, vol. 2, no. 2, pp. 115–119, Sep. 1975.

[28] G. Laporte and U. Palekar, "Some applications of the clustered travelling salesman problem," *Journal of the Operational Research Society*, vol. 53, no. 9, pp. 972–976, Sep. 2002.

[29] S. Anily, J. Bramel, and A. Hertz, "A 53-approximation algorithm for the clustered traveling salesman tour and path problems," *Operations Research Letters*, vol. 24, no. 1–2, pp. 29–35, Feb. 1999.

[30] F. C. J. Lokin, “Procedures for travelling salesman problems with additional constraints,” *European Journal of Operational Research*, vol. 3, no. 2, pp. 135–141, Mar. 1979.

[31] M. W. P. Savelsbergh, “Local search in routing problems with time windows,” *Annals of Operations Research*, vol. 4, no. 1, pp. 285–305, Dec. 1985.

[32] N. Ascheuer, M. Fischetti, and M. Grötschel, “A polyhedral study of the asymmetric traveling salesman problem with time windows,” *Networks*, vol. 36, no. 2, pp. 69–79, Sep. 2000.

[33] G. Ghiani, G. Laporte, and F. Semet, “The Black and White Traveling Salesman Problem,” *Operations Research*, vol. 54, no. 2, pp. 366–378, Apr. 2006.

[34] M. R. Bonyadi, Z. Michalewicz, and L. Barone, “The travelling thief problem: The first step in the transition from theoretical problems to realistic problems,” in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 1037–1044.

[35] S. Polyakovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann, “A comprehensive benchmark set and heuristics for the traveling thief problem,” in *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO ’14*, 2014, pp. 477–484.

[36] P. J. Kolesar, “A Branch and Bound Algorithm for the Knapsack Problem,” *Management Science*, vol. 13, no. 9, pp. 723–735, May 1967.

[37] M. El Yafrani, M. S. R. Martins, M. El Krari, M. Wagner, M. R. B. S. Delgado, B. Ahiod, and R. Lüders, “A fitness landscape analysis of the travelling thief problem,” in *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO ’18*, 2018, pp. 277–284.

[38] “Time and Space Complexity Tutorials & Notes | Basic Programming | HackerEarth.” [Online]. Available:

https://www.hackerearth.com/practice/basic-programming/complexity-analysis/time-and-space-complexity/tutorial/. [Accessed: 16-Nov-2018].

[39] V. Tsurkov, *Large-scale Optimization — Problems and Methods*, vol. 51. Boston, MA: Springer US, 2001.

[40] O. Montiel, F. J. Diaz-Delgadillo, and R. Sepúlveda, "Combinatorial complexity problem reduction by the use of artificial vaccines," *Expert Systems with Applications*, vol. 40, no. 5, pp. 1871–1879, Apr. 2013.

[41] F. J. D. Delgadillo, O. Montiel, and R. Sepúlveda, "Reducing the size of traveling salesman problems using vaccination by fuzzy selector," *Expert Systems with Applications*, vol. 49, pp. 20–30, May 2016.

[42] S. A. Mulder and D. C. Wunsch, "Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks," *Neural Networks*, vol. 16, no. 5–6, pp. 827–832, Jun. 2003.

[43] A. A. Khan, M. U. Khan, and M. Iqbal, "Multilevel Graph Partitioning Scheme to Solve Traveling Salesman Problem," in *2012 Ninth International Conference on Information Technology - New Generations*, 2012, pp. 458–463.

[44] B. Delaunay and others, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, no. 793–800, pp. 1–2, 1934.

[45] J. L. Bentley and J. Louis, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[46] É. D. Taillard and K. Helsgaun, "POPMUSIC for the travelling salesman problem," *European Journal of Operational Research*, vol. 272, no. 2, pp. 420–429, Jan. 2019.

[47] G. Laporte, “The traveling salesman problem: An overview of exact and approximate algorithms,” *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, Jun. 1992.

[48] E. L. Lawler and D. E. Wood, “Branch-and-Bound Methods: A Survey,” *Operations Research*, vol. 14, no. 4, pp. 699–719, Aug. 1966.

[49] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, “Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning,” *Discrete Optimization*, vol. 19, pp. 79–102, Feb. 2016.

[50] J. E. Mitchell, “Branch-and-cut algorithms for combinatorial optimization problems,” *Handbook of applied optimization*, pp. 65–77, 2002.

[51] E. Balas and P. Toth, “Branch and Bound Methods for the Traveling Salesman Problem.” 1983.

[52] M. Padberg and G. Rinaldi, “Optimization of a 532-city symmetric traveling salesman problem by branch and cut,” *Operations Research Letters*, vol. 6, no. 1, pp. 1–7, Mar. 1987.

[53] M. Padberg and G. Rinaldi, “A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems,” *SIAM Review*, vol. 33, no. 1, pp. 60–100, Mar. 1991.

[54] M. Fischetti, J. J. Salazar Gonzalez, and P. Toth, “A branch-and-cut algorithm for the symmetric generalized traveling salesman problem,” *Operations Research*, vol. 45, no. 3, pp. 378–394, 1997.

[55] V. V. Burkhovetskiy and B. Y. Steinberg, “Parallelizing an exact algorithm for the traveling salesman problem,” *Procedia Computer Science*, vol. 119, pp. 97–102, Jan. 2017.

[56] M. Held and R. M. Karp, “A Dynamic Programming Approach to Sequencing Problems,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, no. 1, pp. 196–210, Mar. 1962.

[57] C. L. Valenzuela and A. J. Jones, “Estimating the Held-Karp lower bound for the geometric TSP,” *European Journal of Operational Research*, vol. 102, no. 1, pp. 157–175, Oct. 1997.

[58] G. Reinelt, “TSPLIB - A Traveling Salesman Problem Library,” *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, Nov. 1991.

[59] D. S. Johnson and L. A. McGeoch, “Experimental Analysis of Heuristics for the STSP,” in *The Traveling Salesman Problem and Its Variations*, G. Gutin and A. P. Punnen, Eds. Springer US, 2007, pp. 369–443.

[60] M. El Krari, B. Ahiod, and B. El Benani, “An empirical study of the multi-fragment tour construction algorithm for the travelling salesman problem,” in *Advances in Intelligent Systems and Computing*, vol. 552, 2017, pp. 278–287.

[61] J. L. Bentley, “Fast algorithms for geometric traveling salesman problems,” *ORSA Journal on computing*, vol. 4, no. 4, pp. 387–411, 1992.

[62] J. L. Bentley, “Experiments on traveling salesman heuristics,” in *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, 1990, pp. 91–99.

[63] G. A. Croes, “A Method for Solving Traveling-Salesman Problems,” *Operations Research*, vol. 6, no. 6, pp. 791–812, Dec. 1958.

[64] S. Lin, “Computer solutions of the traveling salesman problem,” *The Bell System Technical Journal*, vol. 44, no. 10, pp. 2245–2269, Dec. 1965.

[65] S. Reiter and G. Sherman, “Discrete Optimizing,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 13, no. 3, pp. 864–889, Sep. 1965.

[66] T. A. J. Nicholson, “A method for optimizing permutation problems and its industrial applications,” *Combinatorial Mathematics and its Applications, Academic Press, London*, pp. 201–217, 1971.

[67] E. H. L. (Emile H. L. . Aarts and J. K. Lenstra, *Local search in combinatorial optimization*. Wiley, 1997.

[68] H. R. Lourenço, O. C. Martin, and T. Stutzle, "Iterated local search," *arXiv preprint math/0102188*, 2001.

[69] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, Nov. 1997.

[70] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing.," *Science (New York, N.Y.)*, vol. 220, no. 4598, pp. 671–80, May 1983.

[71] nojhan, "Classification of metaheuristics," 2007. [Online]. Available: http://metah.nojhan.net/post/2007/10/12/Classification-of-metaheuristics. [Accessed: 23-Nov-2018].

[72] F. Glover, "Tabu Search - Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, Aug. 1989.

[73] F. Glover, "Tabu Search - Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, Feb. 1990.

[74] E. K. Burke and Y. Bykov, "A late acceptance strategy in hill-climbing for exam timetabling problems," in *PATAT 2008 Conference, Montreal, Canada*, 2008.

[75] E. K. Burke and Y. Bykov, "The late acceptance hill-climbing heuristic," *University of Stirling, Tech. Rep*, 2012.

[76] E. K. Burke and Y. Bykov, "The late acceptance Hill-Climbing heuristic," *European Journal of Operational Research*, vol. 258, no. 1, pp. 70–78, Apr. 2017.

[77] F. Chicano, G. Ochoa, D. Whitley, and R. Tinós, "Enhancing partition crossover with articulation points analysis," in *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '18*, 2018, pp. 269–276.

[78] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948.

[79] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43.

[80] M. Dorigo, “Optimization, learning and natural algorithms,” *PhD Thesis, Politecnico di Milano*, 1992.

[81] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[82] V. Maniezzo and A. Colorni, “The ant system applied to the quadratic assignment problem,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 5, pp. 769–778, 1999.

[83] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[84] T. Stützle and H. H. Hoos, “MAX–MIN Ant System,” *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, Jun. 2000.

[85] T. Stutzle and H. Hoos, “MAX-MIN Ant System and local search for the traveling salesman problem,” in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC ’97)*, pp. 309–314.

[86] T. Stützle and H. Hoos, “The Max-Min ANT System and Local Search for Combinatorial Optimization Problems,” in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Boston, MA: Springer US, 1999, pp. 313–329.

[87] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *Applied Soft Computing*, vol. 11, no. 6, pp. 4135–4151, Sep. 2011.

[88] P. Moscato, "Memetic Algorithms: A Short Introduction," D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V Price, Eds. Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999, pp. 219–234.

[89] N. Krasnogor and J. Smith, "A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, Oct. 2005.

[90] P. Shaw, "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems," Springer, Berlin, Heidelberg, 1998, pp. 417–431.

[91] C. Solnon and N. Jussien, Eds., *Ant Colony Optimization and Constraint Programming*. Hoboken, NJ USA: John Wiley & Sons, Inc., 2013.

[92] H. H. Hoos and T. Stützle, *Stochastic local search : foundations and applications*. Morgan Kaufmann Publishers, 2005.

[93] O. C. Martin and S. W. Otto, "Combining simulated annealing with local search heuristics," *Annals of Operations Research*, vol. 63, no. 1, pp. 57–75, Feb. 1996.

[94] O. Martin, S. W. Otto, and E. W. Felten, *Large-step Markov chains for the traveling salesman problem*, vol. 5, no. 3. Oregon Graduate Institute of Science and Technology, Department of Computer Science and Engineering, 1991.

[95] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, Nov. 1997.

[96] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," *Local search in combinatorial optimization*, vol. 1, pp. 215–310, 1997.

[97] P. Hansen and N. Mladenović, "An Introduction to Variable Neighborhood Search," in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Boston, MA: Springer US, 1999, pp. 433–458.

[98] U. Benlic, "Breakout local search pour les problèmes d'optimisation difficiles," 2012.

[99] U. Benlic and J.-K. Hao, "A study of breakout local search for the minimum sum coloring problem," in *Simulated Evolution and Learning*, Springer, 2012, pp. 128–137.

[100] U. Benlic and J.-K. Hao, "Breakout local search for the quadratic assignment problem," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4800–4815, Jan. 2013.

[101] U. Benlic and J.-K. Hao, "Breakout Local Search for maximum clique problems," *Computers {&} Operations Research*, vol. 40, no. 1, pp. 192–206, Jan. 2013.

[102] U. Benlic and J.-K. Hao, "Breakout Local Search for the Max-Cutproblem," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 3, pp. 1162–1173, Mar. 2013.

[103] U. Benlic and J.-K. Hao, "Breakout Local Search for the Vertex Separator Problem." 2013.

[104] Z.-H. Fu and J.-K. Hao, "Breakout local search for the Steiner tree problem with revenue, budget and hop constraints," *European Journal of Operational Research*, vol. 232, no. 1, pp. 209–220, 2014.

[105] S. Ghandi and E. Masehian, "A breakout local search (BLS) method for solving the assembly sequence planning problem," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 245–266, 2015.

[106] U. Benlic, E. K. Burke, and J. R. Woodward, "Breakout local search for the multi-objective gate allocation problem," *Computers & Operations Research*, vol. 78, pp. 80–93, Feb. 2017.

[107] S. Tsubakitani and J. R. Evans, "Optimizing tabu list size for the traveling salesman problem," *Computers & Operations Research*, vol. 25, no. 2, pp. 91–97, Feb. 1998.

[108] M. El Krari, B. Ahiod, and B. El Benani, "Breakout Local Search for the Travelling Salesman Problem," *Computing and Informatics*, vol. 37, no. 3, pp. 656–672, 2018.

[109] P. F. Stadler and W. Schnabl, "The landscape of the traveling salesman problem," *Physics Letters A*, vol. 161, no. 4, pp. 337–344, 1992.

[110] W. Hordijk, "A measure of landscapes," *Evolutionary Computation*, vol. 4, no. 4, pp. 335–360, 1996.

[111] B. Arbel Krakhofer, "Local Optima in Landscapes of Combinatorial Optimization Problems," University of Vienna, Austria, 1995.

[112] P. F. Stadler and W. Schnabl, "The landscape of the traveling salesman problem," *Physics Letters A*, vol. 161, no. 4, pp. 337–344, Jan. 1992.

[113] C. Fonlupt, D. Robilliard, and P. Preux, "Fitness landscape and the behavior of heuristics," in *Evolution Artificielle*, 1997, vol. 97.

[114] O. Martin, S. W. Otto, and E. W. Felten, *Large-step Markov chains for the traveling salesman problem.* Oregon Graduate Institute of Science and Technology, Department of Computer Science and Engineering, 1991.

[115] A. Blažinskas and A. Lenkevičius, "Modified Local Search Heuristics for the Symmetric Traveling Salesman Problem," *Information Technology And Control*, vol. 42, no. 3, Sep. 2013.

[116] P. Moscato and others, “On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms,” *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.

[117] B. Freisleben and P. Merz, “Fitness landscape analysis and memetic algorithms for the quadratic assignment problem,” *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, 2000.

[118] P. Merz and B. Freisleben, “Genetic Algorithms for Binary Quadratic Programming,” in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, 1999, pp. 417–424.

[119] P. Merz and B. Freisleben, “Fitness Landscapes, Memetic Algorithms, and Greedy Operators for Graph Bipartitioning,” *Evol. Comput.*, vol. 8, no. 1, pp. 61–91, Mar. 2000.

[120] D. Bonachea, E. Ingerman, J. Levy, and S. McPeak, “An Improved Adaptive Multi-start Approach to Finding Near-optimal Solutions to the Euclidean TSP,” in *Proceedings of the 2Nd Annual Conference on Genetic and Evolutionary Computation*, 2000, pp. 143–150.

[121] W. W. Hsu and C.-C. Hsu, “The spontaneous evolution genetic algorithm for solving the traveling salesman problem,” in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, 2001, pp. 359–366.

[122] S. Jung and B.-R. Moon, “The Natural Crossover for the 2D Euclidean TSP,” in *Proceedings of the 2Nd Annual Conference on Genetic and Evolutionary Computation*, 2000, pp. 1003–1010.

[123] K. Katayama and H. Narihisa, “Iterated Local Search Approach Using Genetic Transformation to the Traveling Salesman Problem,” in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, 1999, pp. 321–328.

[124] P. MERZ, "Memetic algorithms for combinatorial optimization problems: Fitness Iandscapes and effective search strategies," *Ph. D. thesis, Department of Electrical Engineering and Computer Science, University of Siegen*, 2000.

[125] P. Merz and B. Freisleben, "Genetic local search for the TSP: New results," in *Evolutionary Computation, 1997., IEEE International Conference on*, 1997, pp. 159–164.

[126] G. Gutin, D. Karapetyan, and N. Krasnogor, "Memetic algorithm for the generalized asymmetric traveling salesman problem," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, Springer, 2008, pp. 199–210.

[127] B. Bontoux, C. Artigues, and D. Feillet, "A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 37, no. 11, pp. 1844–1852, 2010.

[128] G. Gutin and D. Karapetyan, "A memetic algorithm for the generalized traveling salesman problem," *Natural Computing*, vol. 9, no. 1, pp. 47–60, 2010.

[129] D. Karapetyan and G. Gutin, "Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 219, no. 2, pp. 234–251, 2012.

[130] J. Bang-Jensen and G. Gutin, *Digraphs : theory, algorithms, and applications*. Springer, 2009.

[131] G. Syswerda, "Uniform Crossover in Genetic Algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, 1989, pp. 2–9.

[132] B. L. Miller, D. E. Goldberg, and others, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[133] L. V Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 174, no. 1, pp. 38–53, 2006.

[134] M. Mestria, L. S. Ochi, and S. de Lima Martins, "GRASP with path relinking for the symmetric Euclidean clustered traveling salesman problem," *Computers & Operations Research*, vol. 40, no. 12, pp. 3218–3229, 2013.

[135] G. Gutin and D. Karapetyan, "Generalized traveling salesman problem reduction algorithms," *arXiv preprint arXiv:0804.0735*, 2008.

[136] M. El Krari, B. Ahiod, and B. El Benani, "A novel reduction algorithm for the generalized traveling salesman problem," *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17*, pp. 105–106, 2017.

[137] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the Lin--Kernighan--Helsgaun Algorithm," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 269–287, 2015.

[138] S. L. Smith and F. Imeson, "GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem," *Computers & Operations Research*, 2017.