Heuristiques pour un Problème de m-Tournées Sélectives

Mahdi Khemakhem

LAMIH
LABORATOIRE D'AUTOMATIQUE, DE MÉCANIQUE ET D'INFORMATIQUE, INDUSTRIELLES ET HUMAINES
UMR CNRS 8530

UVHC

Numéro d'ordre : 08/04

THÈSE

Présentée à

l'Université de Valenciennes et du Hainaut-Cambrésis

en vue de l'obtention du grade de

DOCTEUR

Spécialité Automatique et Informatique des Systèmes Industriels et Humains
Mention Informatique

Par

Mahdi Khemakhem

Maître ès Sciences

Heuristiques pour un Problème de m-Tournées Sélectives

Soutenu publiquement le 1er Février 2008 devant le jury composé de :

M. Abid	Professeur à l'Université de Sfax	Président
H. Chabchoub	Professeur à l'Université de Sfax	Directeur
P. Dejax	Professeur à l'université de Nantes	Rapporteur
T. Loukil	Maître de conférence à l'université de Sfax	Invitée
K. Mellouli	Professeur à l'Université de Carthage	Rapporteur
A. Moukrim	Professeur à l'Université de Technologie de Compiègne	Examinateur
F. Semet	Professeur à l'université de Valenciennes	Directeur

République Tunisienne
Ministère de l'Enseignement Supérieur et
de la Recherche Scientifique

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Cycle de Formation Doctorale
dans la Discipline informatique

Ingénierie des systèmes Informatiques

Thèse de *DOCTORAT*
N° d'ordre: 42/2007

THESE

présentée à

l'École Nationale d'Ingénieurs de Sfax

en vue de l'obtention du

DOCTORAT

Dans la discipline informatique
Ingénierie des systèmes informatiques

par

Mahdi KHEMAKHEM

(Maître ès Sciences)

Heuristiques pour un Problème de m-Tournées Sélectives

soutenu le 1er février 2008, devant le jury composé de :

M.	**Mohamed ABID**	Professeur à l'Université de Sfax	*Président*
M.	**Habib CHABCHOUB**	Professeur à l'Université de Sfax	*Directeur*
M.	**Pierre DEJAX**	Professeur à l'Université de Nantes	*Rapporteur*
Mme	**Taicir LOUKIL**	MdC à l'Université de Sfax	*Invitée*
M.	**Khaled MELLOULI**	Professeur à l'Université de Carthage	*Rapporteur*
M.	**Aziz MOUKRIM**	Professeur à l'Université de Compiègne	*Examinateur*
M.	**Frédéric SEMET**	Professeur à l'Université de Valenciennes	*Directeur*


بسم الله الرحمن الرحيم

A la mémoire de mes grands pères...
A mes parents...
A ma femme et ma fille...
A ma soeur et sa famille...
A mon frère et sa femme...
A mes ami(e)s...

Remerciements

Ce n'est pas par tradition que cette page figure au préambule de ce mémoire, mais c'est plutôt un devoir moral et une reconnaissance sincère qui me pousse à la faire. Je serais en effet ingrat si je n'exprime pas ma reconnaissance et ma gratitude à tous ceux, de prés ou de loin, qui ont facilité ma tâche et m'ont permis de mener à bien ce travail.

Je tiens tout particulièrement à exprimer toute ma gratitude et mes vifs remerciements à mes professeurs Frédéric Semet et Habib Chabchoub qui m'ont accordé leur confiance en me proposant ce sujet de thèse et pour leur aide avec leurs expériences et savoir faire. Je les prie de croire à ma respectueuse estime et ma sincère reconnaissance pour leurs conseils qui m'ont été très précieux et indispensables.

Je ne saurais oublier de remercier Pr Said Hanafi, Dr Mohamed Tmar pour leur collaboration durant ce travail de recherche avec leurs connaissances dans ce domaine de recherche. Qu'ils me permettent de leur exprimer l'assurance de ma gratitude et de mon profond respect.

Pour m'avoir fait l'honneur d'accepter de participer au jury de cette thèse, je remercie mes deux rapporteurs, Pr Khaled Mellouli de l'université de Tunis-Carthage et Pr Pierre Dejax de l'université de Nantes. Je remercie également Pr Aziz Moukrim de l'université de Technologie de Compiègne, Pr Mohamed Abid de l'université de Sfax et Madame Taicir Loukil Maître de conférence de l'université de Sfax, d'avoir accepté d'examiner mon travail.

Je remercie tous les membres des unités de recherche LOGIQ à l'Institut Supérieur de Gestion Industrielle de Sfax, ROI à l'Institut des Sciences et Techniques de Valenciennes et CES à l'École Nationale des Ingénieurs de Sfax pour leur accueil et les moments agréables passés durant ma thèse.

Un grand Merci à mes Parents Abdelmajid et Fatma qui m'ont donné la chance de poursuivre mes études et qui m'ont appris à surpasser les moments difficiles, à ma précieuse femme Nadia et ma mignonne fille Emna pour leur patience éternel et qui mon amour n'a pas de limite, à ma fine soeur Hanen, à mon frère Mhamed, à mon beau frère Khaled, à ma belle soeur Souhir, à ma nièce Hend et mes neveux Youssef et Hedi.

J'aimerais remercier tous ceux que j'ai côtoyé au cours de ces années à l'université de Sfax, à l'université de Valenciennes et à l'université de Gabès, qui m'ont aidé de près ou de loin dans la réalisation de ce travail.

Table des matières

Table des figures

Liste des algorithmes

Liste des tableaux

Introduction

De plus en plus souvent, les informaticiens, les ingénieurs de productions et les gestionnaires sont confrontés à des problèmes combinatoires. Ces problèmes ont un nombre de solutions fini mais très grand, de sorte qu'une résolution par énumération des solutions est inconcevable en pratique. Avec l'augmentation de la concurrence, la pression se fait plus forte non seulement pour trouver une solution qui satisfait les contraintes et soit raisonnablement performante, mais encore pour trouver la meilleure solution possible. C'est notamment le cas en télécommunications où le développement des transports et des communications pose des problèmes dont la taille croît très rapidement et où les enjeux financiers sont considérables. D'une manière plus générale, la phase de l'informatisation de base des administrations publiques et des entreprises est largement entamée et des préoccupations d'amélioration et d'optimisation des processus se font jour.

Face à l'apparition des problèmes d'optimisation de plus en plus complexes, les méthodes de résolution classiques se sont trouvées impuissantes à les résoudre. Dans ce contexte, l'élaboration des heuristiques ou méthodes dites approchées, c'est à dire déterminant une solution satisfaisante non obligatoirement optimale, constitue un passionnant axe de recherche. Ces méthodes sont apparues face à l'impuissance des méthodes dites exactes, c'est à dire déterminant la meilleure solution admissible existante, à résoudre des problèmes NP-difficiles de taille importante. Ainsi, dans les années 1960 et 1970, ont été conçues des heuristiques consacrées à différents problèmes particuliers d'optimisation combinatoire (voyageur de commerce, tournées de distribution, ordonnancement de production...).

Depuis le début des années 1980, différentes approches d'exploration de l'espace des solutions ont vu le jour et ont connu une popularité remarquable. Ces techniques sont connues sous le nom de métaheuristiques ou heuristiques générales de recherche locale. Ces approches empruntent un principe d'optimisation bien spécifique qui peut être adapté sur plusieurs types de problèmes.

Notre étude consiste à traiter un problème de transport appelé Problème de m-Tournées Sélectives PmTS ou Problème de tournées de Véhicules Sélectives PTVS connu encore sous le nom "Team Orienteering Problem" TOP. Le PmTS dérive d'un problème particulier appelé Problème de Tournées de Véhicules PTV. Plus

précisément, il présente une variante d'un ensemble de problèmes, de nature bi-objectifs, d'une classe appelée les Problèmes de Tournées avec Gains PTG. Le PTG consiste à construire une ou plusieurs tournées en maximisant le gain total récolté chez les clients et en minimisant la longueur des tournées. La nature bi-objectifs induit plusieurs variantes qui peuvent figurer dans cette classe en jouant sur la considération des deux objectifs déjà mentionnés. Parmi ces variantes : Le Problème de m-Tournées Sélectives PmTS consiste à maximiser le gain récolté par tous les véhicules et à limiter, au niveau des contraintes, la longueur de chaque tournée par une distance maximale. Le PmTS consiste à construire plusieurs tournées pour m véhicules afin de desservir un portefeuille de clientèle. Ici, la capacité des véhicules n'est pas prise en considération vue que l'on considère que l'on fournit un service aux clients. En contrepartie de ce service, un gain est récolté chez chaque client desservi. Chaque véhicule doit partir d'un dépôt et revenir en un autre après avoir visité un ensemble de clients et sans dépasser la longueur maximale autorisée. Chaque client ne peut être desservi que par un seul véhicule. L'une des caractéristique du PmTS est qu'une solution peut avoir des clients non desservis qui le seront éventuellement ultérieurement. Cette caractéristique est due d'une part à la limitation du nombre de véhicules et d'autre part à la limitation de chaque tournée par une longueur maximale. Le PmTS peut être défini avec un dépôt unique, c'est à dire que chaque véhicule part du dépôt et y revient après la visite d'un ensemble de clients. La variante du PmTS à un seul dépôt est un cas particulier de la variante à deux dépôts. En effet, il suffit de mettre une distance nulle entre les deux dépôts. Un autre cas particulier du problème peut être présenté dans le cas où nous considérons qu'un seul véhicule et un seul dépôt, cette variante revient à résoudre un problème connu sous le nom du Problème du Voyageur de Commerce Sélectif PVCS lorsque nous considérons un seul dépôt et "Orienteering Problem" OP lorsque nous considérons un dépôt de départ et un dépôt d'arrivé.

Le PmTS est un problème NP-difficile. Face à ce constat, notre objectif de recherche consiste à proposer des méthodes de résolution approchées. Ces heuristiques sont prévues pour être intégrées dans un logiciel d'aide à la décision de la planification des tournées des techniciens de maintenance. Ce mémoire présente les résultats de notre travail de recherche et comprend cinq chapitres.

Dans le premier chapitre, nous présentons les caractéristiques du PmTS ainsi que la terminologie utilisée lors de sa description et sa résolution. Ensuite, nous présentons l'intérêt porté à ce problème ainsi que quelques domaines de ses applications. Nous décrivons aussi un modèle pour le PmTS sous forme d'un programme linéaire en nombres entiers ainsi q'un autre sous forme d'un problème sur un graphe. A la fin de ce chapitre, nous présentons une revue de la littérature présentant les travaux déjà effectués pour résoudre le PmTS ainsi que ceux pour résoudre des cas particuliers et voisins de ce problème.

Dans le deuxième chapitre, nous présentons une méthode à deux phases pour résoudre PmTS basée sur la résolution d'un problème de classification et d'un problème d'élaboration des tournées. Nous définissons le problème de classification généralisé, et nous présentons la technique utilisée pour le résoudre. Cette première phase consiste à partitionner le graphe en m sous-ensembles de sommets distincts qui ne s'intersectent qu'aux deux sommets de départ et d'arrivée. Pour ce faire, nous avons utilisé une technique basée sur la recherche à voisinages variables. Elle consiste à construire une partition de classe initiale ensuite d'effectuer des améliorations en échangeant ou déplaçant k sommets entre les classes de sommets. Suite à la description de cette technique nous présentons les résultats numériques obtenus lors de son utilisation au sein de la méthode à deux phases. A la fin du chapitre nous présentons une comparaison analytique de cette méthode en la comparant à d'autres méthodes de la littérature.

Dans le troisième chapitre, nous présentons une autre méthode de classification pour la méthode à deux phases. Nous avons proposé cette méthode pour remédier aux limites de la méthode, présentée dans le deuxième chapitre, à résoudre la classe de problèmes où les dépôts sont séparés. Cette nouvelle méthode consiste à adapter une technique de classification appelée *agrégation autour des centres mobiles*. Nous avons utilisé cette méthode, au sein de la méthode à deux phases, avec trois métriques différentes à savoir les distance *Euclidienne*, de *Profilage* et de *Mahalanobis*. Après avoir définir cette méthode avec ces trois métriques, nous présentons les résultats numériques obtenus lors de leurs utilisations au sein de la méthode à deux phases. A la fin de ce chapitre nous présentons une comparaison des résultats numériques obtenus par la méthode d'agrégation autour des centres mobiles avec les différentes métriques utilisées.

Dans le quatrième chapitre, nous nous intéressons à la deuxième phase de routage de la méthode à deux phases. Nous proposons des approches basées sur la recherche tabou. Ces approches commencent toujours par une solution initiale générée une des méthodes à deux phases décrite dans le deuxième et le troisième chapitre. La première approche est basée sur la recherche tabou avec une structure de voisinage étendu. La deuxième consiste, à réduire cette structure de voisinage, à changer et raffiner certains paramètres et composants utilisés dans la première approche. Ces changements nous ont permis des solutions de meilleure qualité et d'obtenir des temps de calcul plus raisonnables. Dans la troisième approche, nous avons utilisé la notion de pénalisation de la fonction objectif afin de pourvoir osciller entre les zones de recherche réalisables et non réalisables. Nous avons proposé une normalisation de la fonction objectif pénalisée afin de réduire d'avantage le temps d'exécution de notre approche. Nous présentons, après la description de chaque approche, les résultats numériques obtenus par nos méthodes afin de les comparer aux résultats obtenus par les méthodes de la littératures pour la résolution du PmTS.

Dans le cinquième chapitre, nous présentons une approche de résolution du PmTS basée sur le principe de la mémoire adaptative. Cette approche a été inspirée d'une heuristique de résolution du Problème de Tournées de Véhicules avec contraintes de Capacités PTVC, appelée *BoneRoute*. Notre heuristique utilise la méthode taboue à voisinage réduit, présentée dans le quatrième chapitre, au sein d'un concept de mémoire adaptative. Nous avons testé cette heuristique sur les instances benchmark du PmTS et elle a fournit des résultats compétitifs par rapport aux méthodes que nous avons proposé et aux méthodes déjà existantes. Nous commençons par la présentation de la méthode *BoneRoute* pour la résolution du PTVC ainsi que les schémas générale et détaillé de notre approche de résolution du PmTS. Ensuite, nous précisons les paramétrages utilisés dans cette approche. Enfin, nous exposons les résultats numériques et leurs interprétations.

La dernière partie de ce mémoire représente une conclusion générale sur ce travail et précise nos perspectives de recherche.

Chapitre 1

Le Problème de m-Tournées Sélectives

1.1 Introduction

De nos jours, vu l'augmentation importante du prix de l'énergie, l'intérêt pour le domaine de transport est devenu de plus en plus marqué. En effet, afin de minimiser les coûts de transport, les entreprises visent à perfectionner leurs systèmes logistiques pour pouvoir rester compétitives dans leurs domaines. Les systèmes logistiques comportent plusieurs éléments, dont l'un des plus importants est la planification des tournées des véhicules dans les réseaux de transport.

Durant le dernier siècle, plusieurs modèles et méthodes ont été proposés pour répondre aux divers problèmes se posant dans les systèmes logistiques. Toutes ces techniques sont directement liés à l'évolution de l'informatique. En effet, aujourd'hui et avec un calculateur de moyenne capacité, nous pouvons résoudre en quelques secondes des problèmes venant de la logistique qu'il aurait été impossible à résoudre auparavant. La programmation mathématique est la plus ancienne technique de résolution des problèmes issus de la logistique. Cependant, cette méthode est actuellement beaucoup moins utilisée vu la complexité des problèmes traités qui engendre l'impossibilité de leur résolutions au bout d'un temps raisonnable. Avec la révolution des ordinateurs personnels, une nouvelle méthodologie est apparue. Elle consiste à résoudre les problèmes en utilisant des algorithmes de recherche fournissant des bonnes solutions, pas nécessairement optimales, au bout d'un temps de calcul raisonnable.

Le long de ce travail nous nous intéressons aux problèmes de distribution des biens entre entrepôts et clients. Ces problèmes sont connus sous le nom de Problèmes de Tournées de Véhicules (PTV). Les applications pratiques de ces problèmes incluent : la distribution de journaux (Dillmann *et al.*, 1996), la distribution de pro-

duits aux hypermarchés et magasins (Semet et Taillard, 1993), le transport pour des entreprises de laiteries et de construction (Tarantilis et Kiranoudis, 2007), l'alimentation pour le bétail (Mullaseril *et al.*, 1997), l'approvisionnement de distributeurs de boissons (Golden et Wasil, 1987; Kleywegt, 1996), le transport de produits dangereux (Boulanger *et al.*, 2006, 2007).

Le PTV a été introduit par Dantzig et Ramser qui l'ont nommé le Problème de Répartition de Camions (Dantzig et Ramser, 1959). Il consiste à planifier les tournées de la flotte de véhicules pour desservir les clients. La construction des tournées devient de plus en plus complexe face à l'ajout de contraintes et d'objectifs. Parmi les plus importants figurent la maximisation de la satisfaction des clients et la minimisation des coûts des tournées.

La version classique du PTV est le Problème de Tournées de Véhicules avec contrainte de Capacité PTVC. Cette classe de PTV, consiste à livrer des biens à chacun des clients, connu à l'avance, par un seul passage d'un seul véhicule. Tous les véhicules sont identiques et possèdent une capacité maximale de chargement. L'objectif consiste à minimiser le coût total de la desserte des clients. Le PTVC est un problème classé NP-difficile (Lenstra et Rinnooy-Kan, 1981). La majorité des travaux, dans la littérature, sont donc basées sur des méthodes heuristiques fournissant des solutions de bonne qualité en un temps de calcul raisonnable (Laporte, 1992b; Toth et Vigo, 2001).

En considérant une seule tournée, la résolution du PTV revient à résoudre le Problème de Voyageur de Commerce PVC. Ce problème fait partie des problèmes d'optimisation les plus traités. La résolution du PVC consiste à construire la plus courte tournée d'un représentant de commerce passant par tous les clients et revenant à son point de départ. Ce problème peut être rencontré dans de nombreuses problématiques complexes. En effet, certains problèmes peuvent être décomposés en sous-problèmes parmi lesquels nous retrouvons le PVC.

Le problème, traité dans cette thèse, est un problème de transport dérivé du PTV appelé le Problème de m-Tournées Sélectives PmTS ou le Problème de Tournées de Véhicules Sélectives PTVS. Le PmTS est une variante d'un ensemble de problèmes d'une classe appelée les Problèmes de Tournées avec Gains PTG étudiés par (Feillet, 2001). Le PTG consiste à construire une ou plusieurs tournées en maximisant le gain total récolté chez les clients et en minimisant la longueur des tournées. Ces problèmes sont par nature bi-objectifs. De ce fait, plusieurs variantes peuvent figurer dans cette classe en jouant sur la considération des deux objectifs déjà mentionnés. Parmi ces variantes nous pouvons citer : Le Problème de Tournées Sélectives PTS qui consiste à maximiser le gain récolté par tous les véhicules et à limiter, au niveau des contraintes, la longueur de chaque tournée par une distance maximale. Le Problème de Tournées avec Quota PTQ qui consiste à minimiser les longueurs parcourues par toutes les véhicules et à remplacer la maximisation du gain total récolté par des contraintes

imposant un gain minimal récolté par chaque véhicule. Le Problème de Tournées Profitables PTP qui consiste à prendre en considération simultanément les deux objectifs.

Le PmTS consiste à construire les tournées de m véhicules afin de desservir un portefeuille de clientèle. Ici, la capacité des véhicules n'est pas prise en considération vu que l'on considère que l'on fournit un service aux clients. En contrepartie de ce service, un gain est récolté chez chaque client desservi. Chaque véhicule doit partir d'un dépôt et revenir en un autre après avoir visité un ensemble de clients sans que la tournée ne dépasse la longueur maximale autorisée. Chaque client ne peut être desservi que par un seul véhicule. L'une des caractéristiques du PmTS est qu'une solution peut avoir des clients non desservis qui le seront éventuellement ultérieurement (voir figure 1.1).

Cette caractéristique est due d'une part à la limitation du nombre de véhicules et d'autre part à la limitation de chaque tournée par une longueur maximale. Le PmTS peut être défini avec un dépôt unique, c'est à dire que chaque véhicule part du dépôt et y revient après la visite d'un ensemble de clients. La variante du PmTS à un seul dépôt est un cas particulier de la variante à deux dépôts. En effet, il suffit de mettre une distance nulle entre les deux dépôts (voir figure 1.2).

Un autre cas particulier du problème correspond au cas où nous considérons qu'un seul véhicule et un seul dépôt, cette variante revient à résoudre un problème connu sous le nom du Problème du Voyageur de Commerce Sélectif PVCS.

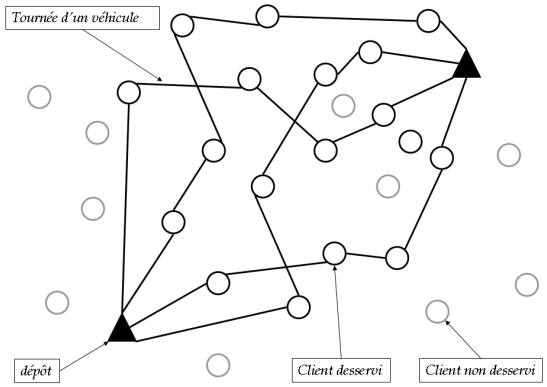

FIGURE 1.1 – Exemple d'une solution du PmTS à deux dépôts avec 4 véhicules

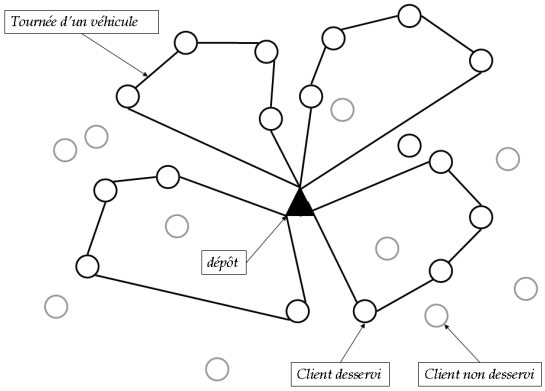

FIGURE 1.2 – Exemple d'une solution du PmTS à un seul dépôt avec 4 véhicules

Dans la section suivante, nous présentons la terminologie utilisée dans la description et la résolution du PmTS, sa complexité, ainsi que certaines propriétés. Dans la troisième section, nous décrivons des exemples d'applications du PmTS. La quatrième section est consacrée à la modélisation du PmTS. Dans la dernière section, nous présentons un survol de la littérature décrivant les différentes méthodologies utilisées pour résoudre le PmTS.

1.2 Terminologie, complexité et propriétés

Afin de modéliser et de décrire les modèles et méthodes de résolution du PmTS, nous présentons dans cette section la terminologie que nous avons utilisé dans ce manuscrit.

1.2.1 Terminologie

L'un des concepts fondamentaux en géographie des transports est rattaché aux réseaux et leurs structures spatiales. Un réseau est en quelque sorte le squelette d'un système visant à établir une forme de communication tangible telle que les routes, les voies ferrées ou encore les corridors aériens et maritimes. Une route est un lien simple entre deux sommets s'inscrivant à l'intérieur d'un réseau plus vaste. Les réseaux de transport illustrent l'organisation territoriale des activités économiques ainsi que l'effort déployé à dessein de s'affranchir des distances (Rodrigue *et al.*, 2007).

Les données d'un PmTS est un réseau d'arêtes pouvant être traversées par l'ensemble de véhicules fixé, que nous pouvons représenter symboliquement par un graphe. Il s'agit en fait, d'une abstraction de la réalité de sorte à permettre sa modélisation. Nous définissons dans cette section les éléments de base utiles à l'encodage d'un graphe et ainsi à la définition d'une solution du PmTS.

Un graphe G non orienté est un ensemble de sommets (ou de noeuds) V et d'arêtes A. Un sommet est un point d'extrémité d'une arête. Il s'agit d'une abstraction d'un lieu tel une ville, une division administrative, une intersection routière ou une infrastructure de transfert (stations, terminus, ports et aéroports). Une arête est un lien entre deux sommets. L'arrête $\{v_i, v_j\}$ est caractérisée par deux sommets v_i et v_j. Une arrête est une représentation abstraite d'infrastructures de support des déplacements entre deux noeuds. Contrairement au terme *orienté*, la désignation du terme *non orienté* consiste dans le fait que les liens entre les sommets n'ont pas une direction bien déterminée (c'est dire que l'on distingue pas le parcours de v_i vers v_j et celui de v_j à v_i).

Un réseau de transport permet la circulation des flux d'individus, de frêt ou d'information. La théorie des graphes se doit donc de considérer la possibilité de représenter les mouvements. Une arête incarne toute possibilité de mouvement entre deux noeuds sans considérer la direction. Une chaîne élémentaire est une séquence d'arêtes adjacentes où chaque sommet est de degré deux à l'exception des sommets des extrémités. Le coût d'une arête est un nombre associé à une arête qui peut être une distance, un temps de parcours, etc. Le coût d'une chaîne est la somme des coûts des arêtes constituant cette chaîne. Une chaîne simple dont le sommet initial et terminal coïncident constitue un cycle.

Dans ce qui suit, nous présentons des termes propre au PmTS. Soit un graphe $G = (V, A)$ non orienté, $V = \{v_1, v_2, \ldots, v_n\}$ désigne l'ensemble de n sommets, Pour $i \in \{2, ..., n-1\}$, v_i représente un client i, v_1 et v_n représentent les dépôts. $A \subseteq V \times V$ désigne l'ensemble des arêtes, représentant les différentes routes reliant les clients. Une matrice des distances $C = \{c_{ij}\}$, de taille $(n \times n)$ permet de définir pour tout $(v_i, v_j) \in A$ la distance entre le client i et le client j, $c_{ij} = \infty$ si $(v_i, v_j) \notin A$. Un vecteur $P = \{p_i\}$, de taille n permet de définir pour tout $(v_i) \in V$ le gain (ou profit) récolté lors de la visite d'un client i.

Dans le cas du PmTS à deux dépôts, une *tournée* est représentée par une chaîne où ces extrémités sont constituées par les deux sommets (ou dépôts) v_1 et v_n. Pour le PmTS à un seul dépôt, une *tournée* est représentée par un cycle partant du dépôt et revenant au même dépôt après la visite d'un ensemble de sommets.
Une solution du PmTS notée Π est un ensemble de m *tournées* assurées par les m véhicules définis, $\Pi = \{\pi_1, \pi_2, \ldots, \pi_k, \ldots, \pi_m\}$ où π_k est la chaîne (ou *tournée*) composée par les sommets $v_i \in V$ desservi par le véhicule k.

$$\pi_k = (V(\pi_k), A(\pi_k)), \quad V(\pi_k) \subseteq V, \quad A(\pi_k) \subseteq A$$

Mise à part les dépôts, un sommet (ou un client) n'est visité qu'une seule fois et par une seule *tournée* (ou un véhicule). La longueur $\ell(\pi_k)$ d'une *tournée* π_k est définie par le cumul les distances des arêtes qui la composent. Chaque *tournée* ne doit pas dépasser une longueur maximale L définie. Un gain (ou profit) $p(\pi_k)$ d'une

tournée est la somme des gains p_i des sommets v_i qui la composent. Le gain total d'une solution est le gain récolté par les m *tournées*. Il est à noter que les dépôt v_1 et v_n ont des gains nuls ($p_1 = p_n = 0$).

1.2.2 Propriétés

Nous supposons qu'une solution Π du PmTS vérifie les hypothèse suivantes :

- Π est définie sur un graphe non orienté.
- Π peut être fournie avec un seul ou deux dépôts selon l'instance du PmTS traitée.
- Chaque véhicule part du dépôt v_1 et arrive à v_n.
- Le nombre des véhicule m est fixé par le décideur.
- Un véhicule n'a pas de limite de capacité.
- Chaque véhicule k est limité par une distance limite de parcours L, $\ell(\pi_k) \leq L$.
- Chaque client v_i peut être desservi par n'importe quel véhicule k de la flotte.
- La matrice des distance est symétrique (graphe orienté) et vérifie l'inégalité triangulaire : $\forall v_i, v_j, v_k \in V$, $c_{ij} = c_{ji}$ et $c_{ik} + c_{kj} \leq c_{ij}$.

La définition du PmTS et les hypothèses précédentes permettent de déduire les propriétés suivantes :

Propriété 1 : Une solution optimale peut être composée uniquement par un sous-ensemble de sommets (ou clients) sélectionnés par la méthode de résolution. Cette caractéristique de la solution du PmTS a motivé l'apparition du terme "*sélectif*" dans le nom du problème.

Propriété 2 : Pour un graphe $G(V, A)$ de taille n et une distance maximale L, la résolution du PmTS peut être effectuée sur un graphe *réduit* $G'(V', A')$ de taille $\overline{n}$ tel que $\overline{n} \leq n$, $V' \subseteq V$ et $\forall v_i \in V - V', c_{1i} + c_{in} > L$.

Propriété 3 : Dans une solution optimale Π^*, nous pouvons trouver un sous ensemble de k tournées vides $\Pi^v \subset \Pi^*$ tel que $\Pi^v = \{\pi_1, \ldots, \pi_k\}$ et $0 \leq |\Pi^v| \leq m$. $\forall \pi_k \in \Pi^v, \pi_k = \{v_1, v_n\}$ et $\ell(\pi_k) = c_{1n}$.

Preuve **:** En considérant la deuxième propriété et l'hypothèse de l'inégalité triangulaire, et supposant que $\forall v_i \in V - \{v_1, v_n\}$, $c_{1i} + c_{in} > L$ tous les sommets v_i ne peuvent pas figurer dans la solution et nous aurons toutes les tournées vides ($|\Pi^v| = m$). Pour $v_i \in V$ tel que $c_{1i} + c_{in} > L$, la solution peut avoir un certain nombre de tournées vides ($0 \leq |\Pi^v| < m$).

Propriété 4 : Résoudre un PmTS à deux dépôts sur un graphe non orienté de taille n revient à résoudre un PmTS à un seul dépôt sur un graphe orienté dont le nombre des sommets est $(n-1)$.

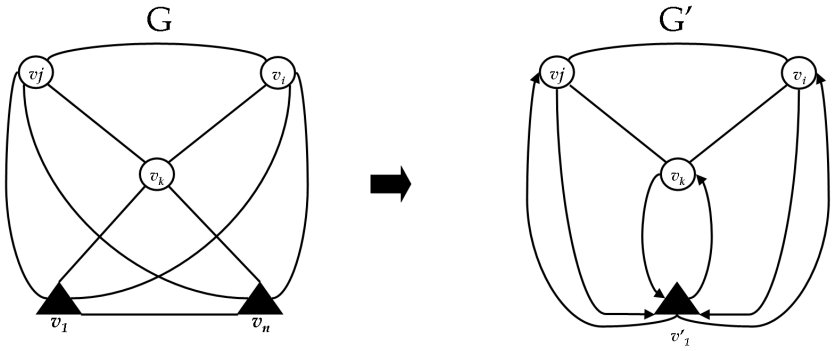

FIGURE 1.3 – Illustration de la propriété 4

Preuve **:** Supposons que nous avons un graphe $G(V, A)$ non orienté ($\forall v_i, v_j \in V, c_{ij} = c_{ji}$) de taille n, v_1 et v_n représentant les dépôts (voir le graphe G de la figure 1.3). Pour se ramener à la résolution du PmTS à un seul dépôt, il suffit d'annuler la distance entre les deux dépôts v_1 et v_n ($c_{1n} = 0$). Dans ce cas, v_1 et v_n sont considérés comme un seul sommet soit v'_1 et nous obtenons un graphe $G'(V', A')$ avec $V' = \{v'_1\} \cup V \backslash \{v_1, v_n\}$ et $A' = \{\{(v'_1, v_i), (v_i, v'_1)\} \cup A \backslash \{\{v_1, v_i\}, \{v_n, v_i\}\}, v_i \in V'\}$ (voir le graphe G' de la figure 1.3). Dans le graphe G', les distances c'_{1i} et c'_{i1} de v'_1 par rapport aux autres sommets v_i (clients) et inversement sont asymétriques. Nous avons $c'_{1i} = c_{1i}$, $c'_{i1} = c_{n1}$ donc $c'_{1i} \neq c'_{i1} \forall v_i \in V' \backslash \{v'_1\}$. De ce fait, le graphe G' est un graphe asymétrique de taille $(n-1)$.

Propriété 5 : Résoudre un PmTS à deux dépôts revient à résoudre un PmTS à un seul dépôt sur un graphe non orienté ayant $|A| + 1$ arêtes.

Preuve : Pour la résolution du PmTS à deux dépôt v_1 et v_2 sur un graphe G nous avons $p_i = p_n = 0$ et $c_{1n} \neq 0$. Si nous considérons que les deux dépôt v_1 et v_n sont confondus c'est à dire $c_{1n} = 0$ et que $p_n = \infty$ nous obtenons un nouveau graphe G' (voir figure 1.4). Avec les condition du graphe G', le processus de résolution considère obligatoirement le sommet v_n car il possède un gain infini et il est est situé à une distance nulle du dépôt v_1. La distance nulle entre v_1 et v_n garantit l'adjacence entre ces deux sommets dans chaque tournée de la solution. Par conséquent nous aurons une solution du PmTS à deux dépôts.

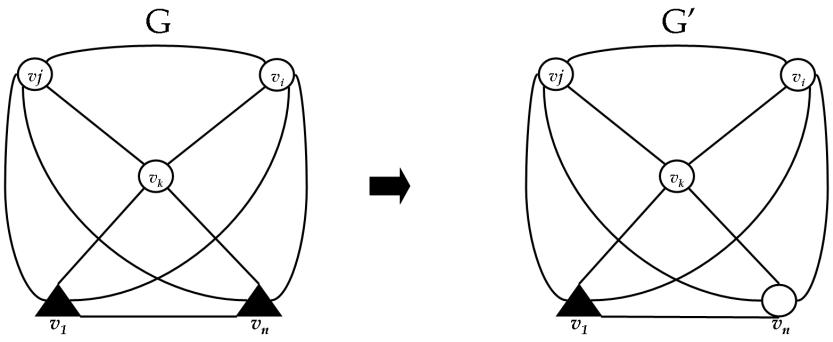

FIGURE 1.4 – Illustration de la propriété 5

1.2.3 Complexité

Nous avons mentionné, dans la section précédente, que le PmTS est une généralisation du PVCS. D'un autre coté, le PVCS est une variante du PVC classique. En se basant sur la complexité de la détermination d'un cycle hamiltonien introduite par (Garey et Johnson, 1979), Laporte (1992a) a affirmé que le PVC est un problème NP-difficile. Puisque le PVC peut se ramener au PVCS, le PVCS a été démontrée NP-difficile par (Golden *et al.*, 1987). En conséquence, le PmTS est un problème NP-difficile (Chao *et al.*, 1996b).

En pratique, la difficulté de la résolution du PmTS dépend des trois paramètres n, m et L associés à une instance. La variation de l'un de ces paramètres joue un rôle très important sur la qualité de la solution et le temps de résolution du PmTS. Ceci sera montré expérimentalement dans les chapitres suivants.

1.3 Exemples d'applications

En pratique, nous pouvons trouver plusieurs applications du PmTS. Elles différent selon les domaines et selon les besoins du décideur. Les domaines les plus courants sont ceux du transport tels que la distribution des biens logiques (services) ou physiques (marchandises) et du routage dans les réseaux informatiques.

La planification des tournées des techniciens de maintenance et de réparation peut être une application directe du PmTS. Une entreprise, spécialiste de la vente des produits nécessitant des services après-vente, possède un ensemble de techniciens motorisés qui assurent les tâches de maintenance et de réparation après-vente. Chaque technicien doit visiter quotidiennement un ensemble de clients pour satisfaire leurs demandes. Une demande d'un client est caractérisée par un poids qui peut se représenter sous forme d'un profit au faveur de la société ou d'une priorité

d'urgence. Le système logistique de cette société doit affecter quotidiennement à chaque technicien, disposant d'un véhicule, un ensemble de clients à satisfaire. Pour un jour quelconque, la sélection d'un client dépend, d'une part de sa priorité ou du bénéfice engendré par sa visite, d'autre part de son emplacement géographique par rapport au(x) dépôt(s). Dans le cas où les tournées des techniciens sont effectuées sur le plan régional, l'application revient sur un PmTS à un seul dépôt. Dans le cas où elle s'effectuent sur le plan national, la nécessité d'un deuxième dépôt peut se présentée.

La distribution urgente des produits pharmaceutiques peut être une autre application directe du PmTS. En effet, un grossiste en produits pharmaceutiques peut avoir des demandes de médicaments, en urgence, de la part d'un ensemble de pharmacies clientes. Par le fait que la demande est urgente, un client peut être satisfait par la livraison d'une petite quantité du produit demandé. Dans ce cas, la capacité des véhicules ne pose pas une contrainte. Le système logistique du fournisseur doit lui fournir la planification des tournées de ses véhicules selon la disposition géographique du client et de sa priorité. La priorité d'un client est définie, d'une part selon la nature du produit demandé, et d'autre part de son importance déduite généralement de son historique.

Dans les réseaux informatiques, le routage d'une information en temps réel entre un ensemble de clients peut être une application directe du PmTS. En effet, supposons que nous avons un serveur d'informations en temps réel lié à un ensemble de clients. Ces clients communique à travers une connexion distante (voir figure 1.5). A l'arrivée d'une information au serveur, celle-ci doit être transmise immédiatement vers toutes les machines clientes. Pour respecter l'instantanéité de la diffusion de l'information reçue, le serveur doit l'envoyer à travers un certain nombre d'enveloppes de données (véhicules par analogie). Chaque enveloppe doit desservir un certain nombre de machines et est limité par une longueur d'acheminement maximale vu le risque de perte d'information face au parcours d'un long chemin. La priorité de chaque machine est déterminée selon l'impacte entre le type de l'information reçue et le profil du client. Après l'acheminement de l'information vers un certain nombre de machine, chaque enveloppe de données doit retourner au serveur marqué par les accusés de réception des clients servis. L'exemple d'illustration de la figure 1.5 présente le routage d'une information vers les machines par deux enveloppes d'informations. Le sommet de faible impact n'est pas servi car sa disposition géographique n'est pas à proximité des routes établies entre les autres machines. En effet, si la disposition de cette machine est proche de l'une de ces routes, elle sera prise par cette route à condition que cela ne viole pas la contrainte de longueur maximale.

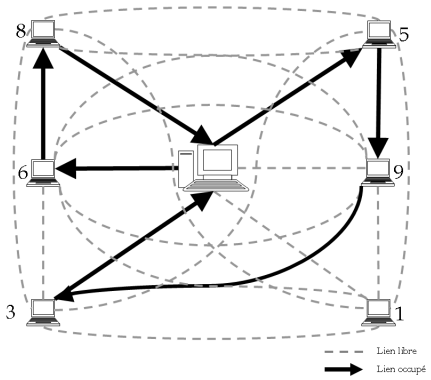

FIGURE 1.5 – Illustration de l'application sur les réseaux informatiques

1.4 Modélisation

Nous présentons dans cette section deux formes de modélisation du PmTS à deux dépôts. La première forme est une modélisation sous forme d'un graphe. La deuxième consiste à une modélisation en programme linéaire en nombre entiers binaires.

1.4.1 Modélisation du PmTS à deux dépôts sous forme d'un problème sur un graphe

Afin de pouvoir modéliser le PmTS à deux dépôts sur un graphe, nous considérons les notations supplémentaires suivantes.

- $\pi_k^h = (V(\pi_k^h), A(\pi_k^h))$ est une *chaîne* extraite d'une tournée π_k avec $V(\pi_k^h) \subseteq V(\pi_k)$ et $A(\pi_k^h) \subseteq A(\pi_k)$.
- $\pi_k^h \leftrightarrow \pi_k^q$ signifie que la chaîne extraite π_k^h est non adjacente à la chaîne extraite π_k^q.
- $V(\pi_k) \cap V(\pi_h)$ est l'ensemble des sommets en intersection entre les deux tournées π_k et π_h.

$$Max \sum_{k=1}^{m} p(\pi_k) \tag{1.1}$$

sous les contraintes

$$V(\pi_k) \cap V(\pi_h) = \{v_1, v_n\} \qquad \forall k, h \in \{1, \ldots, m\}, k \neq h \tag{1.2}$$

$$V(\pi_k^h) \cap V(\pi_k^q) = \oslash \qquad \forall k \in \{1, \ldots, m\}, \forall \pi_k^h, \pi_k^q \subset \pi_k, \pi_k^h \leftrightarrow \pi_k^q \tag{1.3}$$

$$\ell(\pi_k) \leq L \qquad \forall k \in \{1, \ldots, m\} \tag{1.4}$$

Les significations de chaque équation du modèle sont les suivantes. L'équation (1.1) présente la fonction objectif qui consiste à maximiser la somme des gain récoltés par les m tournées. La contrainte (1.2) assure qu'un sommet n'est visité que par une seule tournée à l'exception des sommets v_1, v_n. L'équation (1.3) assure qu'un sommet n'est visité qu'une seule fois par une tournée, en d'autre terme elle permet d'éliminer les sous-tours sur une tournée, on s'assure que la chaîne est élémentaire. La contrainte (1.4) assure que chaque tournée ne peut pas excéder la longueur limite L.

1.4.2 Modélisation sous forme d'un programme linéaire en nombres entiers

Afin de formuler le PmTS à deux dépôts sous forme d'un programme linéaire en nombres entiers nous considérons les notations mentionnées précédemment et les variables de décisions suivantes :

- $x_{ijk} = 1$ si l'arrête $\{v_i, v_j\} \in A$ est utilisée par la tournée k, 0 sinon.
- $y_{ik} = 1$ si le sommet $v_i \in V \backslash \{v_1, v_n\}$ est visité par la tournée k, 0 sinon.

En supposant que la matrice des distances est symétrique, le PmTS peut être modélisé comme suit :

$$Max \sum_{i=2}^{n-1} \sum_{k=1}^{m} p_i y_{ik} \tag{1.5}$$

sous les contraintes

$$\sum_{j=2}^{n} \sum_{k=1}^{m} x_{1jk} = m \tag{1.6}$$

$$\sum_{j=1}^{n-1} \sum_{k=1}^{m} x_{jnk} = m \tag{1.7}$$

$$\sum_{i<j} x_{ijk} + \sum_{i>j} x_{jik} = 2y_{ik} \qquad (j = 2, \ldots, n-1), (k = 1, \ldots, m) \tag{1.8}$$

$$\sum_{i=1}^{n} \sum_{i>j} c_{ij} x_{ijk} \leq L \qquad (k = 1, \ldots, m) \tag{1.9}$$

$$\sum_{k=1}^{m} y_{ik} \leq 1 \qquad (i = 2, \ldots, n-1) \tag{1.10}$$

$$\sum_{i,j \in U, i<j} x_{ijk} \leq |U| - 1 \qquad (U \subset V - \{v_1, v_n\}, |U| \geq 3)$$

$$(k = 1, \ldots, m) \tag{1.11}$$

$$x_{ijk} \in \{0,1\} \qquad (1 \leq i < j \leq n), (k = 1, \ldots, m) \tag{1.12}$$

$$y_{ik} \in \{0,1\} \qquad (i = 2, \ldots, n-1), (k = 1, \ldots, m) \tag{1.13}$$

Les significations des équations du modèle sont les suivantes. La fonction objectif (1.5) représente la maximisation du gain total. Les contraintes (1.6) et (1.7) assurent (respectivement) qu'il y a m tournées partant du sommet v_1 et qu'il y a m tournées arrivant au sommet v_n. La contrainte (1.8) garantie que le degré de chaque sommet visité est deux. La contrainte (1.9) limite la longueur de chaque tournée par la longueur maximale L. La contrainte (1.10) assure que chaque sommet est visité au plus une seule fois à l'exception des sommets de départ et d'arrivée v_1 et v_n. La contrainte (1.11) est la contrainte d'élimination des sous-tours. Finalement, les deux dernières contraintes (1.12) et (1.13) sont les contraintes d'intégrité.

1.5 Revue de la littérature

Nous avons mentionné dans les sections précédentes que le PmTS est un problème dérivé du PTV et qu'il possède de nombreux cas particuliers en modifiant certaines hypothèses. Nous présentons dans cette section une étude bibliographique de plusieurs travaux de la littérature qui ont des liaisons directes et indirectes avec la PmTS.

Le PmTS est un problème dérivé du PTV et appartient à une famille de problèmes appelés les Problèmes de Tournées avec Gains PTG. Cette famille de problèmes a été étudiée en détail par Feillet (2001). En général, ces problèmes sont des généralisation du Problème du Voyageur de Commerce PVC et du Problème de Tournées de Véhicules PTV. Mise à part de la minimisation des tournées selon les longueurs des chemins empruntés, les PTG sont caractérisés par la maximisation des gains récoltés chez les clients visités.

Le PmTS appartient à la sous-classe des PTG. Il existe plus de travaux liés à cette classe que à celle des PTQ et PTF. La plupart de ces travaux ont considéré le cas d'une seule tournée. Notre étude bibliographique, est organisée selon la nature des problèmes et la technique de résolution en suivant l'ordre chronologique des travaux. Dans la première partie nous présentons brièvement les travaux sur les problèmes voisins du PmTS. Dans la deuxième partie nous présentons, plus profondément, les travaux existants traitant directement le PmTS.

1.5.1 Littérature sur les problèmes voisins du PmTS

Le problème voisin du PmTS le plus étudié, dans la littérature, est le Problème du Voyageur de Commerce Sélectif PVCS qui est un problème de la classe PTG. Comme déjà mentionné, la résolution du PmTS à une seule tournée revient à résoudre le

PVCS ($m = 1$). Le PVCS a eu différentes appellations et diverses méthodes de sa résolution ont été proposées.

Feillet *et al.* (2005) ont proposé une revue de la littérature du PVCS en comparant les méthodes existantes pour sa résolution. Les plus anciens travaux sur le PVCS ont été réalisés par Hayes et Norman (1984) et Tsiligirides (1984). Tsiligirides (1984) a proposée une heuristique stochastique, par contre Hayes et Norman (1984) ont proposé une approche basée sur la programmation dynamique. L'appellation du problème utilisée, selon la terminologie anglaise, est "Orienteering Problem" OP. Le OP a été étudié d'avantage et appliqué à un cas pratique par Golden *et al.* (1987), ils ont proposé une nouvelle approche heuristique pour ce problème. Une autre heuristique a été proposée par Golden *et al.* (1988) pour améliorer les résultats trouvés dans (Golden *et al.*, 1987). Keller (1989a) a proposé une autre méthode heuristique et a effectué une étude comparative avec toutes les autres méthodes de résolution du OP (PVCS), proposée dans la littératures de l'époque, du point de vue des méthodologies de résolution et de celui de la qualités de solutions fournies. Ramesh et Brown (1991) ont proposée une heuristique à quatre phase pour résoudre la PVCS. Wang *et al.* (1995) ont proposé une méthode de résolution du PVCS basée sur l'apprentissage avec les réseaux de neurones. Chao *et al.* (1996a) ont proposé une heuristique rapide et efficace qui a dépasser toutes les autres approches au point de vue qualité de solution et temps d'exécution.

Toutes les méthodes approchées déjà citées ont été testées sur des problème de petites taille ($n \leq 33$). Gendreau *et al.* (1998b) ont proposé une approche tabou pour la résolution du PVCS, cette approche a été testée sur des instances de la littérature ainsi que sur d'autres instances jusqu'à $n = 300$. Cette heuristique commence par une solution initiale produite par une procédure appelée *Insert and Shake*. Cette dernière combine deux heuristiques de résolution du PVC à savoir celle de Rosenkrantz *et al.* (1977) et celle appelée $GENIUS$ de Gendreau *et al.* (1992). Elle consiste à construire une tournée selon le principe utilisée par l'heuristique appelée $H2$ de Laporte et Martello (1990) ensuite à la ré-optimiser avec l'algorithme $GENIUS$. Une fois que la solution initiale est construite, une procédure basée sur la recherche tabou est lancée. La structure de voisinage utilisée dans cette procédure tabou consiste à éliminer ou insérer de la tournée un ensemble de sommets défini par des cluster (groupes). Afin d'identifier ces cluster un indice de proximité, basée sur le principe de la distance moyenne, a été défini. Le passage d'une solution à une autre solution voisine est guidé par une liste de mouvements candidats évalués par ratio tenant compte des longueurs des tournées et de leurs profits. La liste des mouvements tabous utilisée est de teneur variable. Le principe d'aspiration, utilisé dans cette heuristique, consiste à insérer dans la tournée actuelle des sommets déclarés tabous et qui apparaissent prometteurs selon une évaluation bien déterminée. Ensuite et tant que la longueur de la tournée dépasse la longueur limite, la procédure US de Gendreau *et al.* (1992)

est appliquée pour éliminer des sommets dont le but de réduire la longueur de la tournée. La procédure tabou actuelle, applique un principe d'oscillation utilisée dans l'heuristique *TabouRoute* pour la résolution du PTV (Gendreau *et al.*, 1994). Ce principe consiste à accepter, au cours de la recherche, des solutions non réalisables, c'est à dire, des tournées de longueur qui dépassent la longueur autorisée dans le cas du PVCS. L'oscillation est assurée par une pénalisation par un paramètre évaluant dynamiquement au cours de la recherche. Ce paramètre est utilisé lors du choix du cluster à utiliser dans un passage d'une solution à une autre solution voisine. Bar-Yehuda *et al.* (2005) ont présenté une interprétation géométrique d'un problème dérivé du PVCS. Ce problème, appelé PVCSFT, consiste à ajouter les contraintes de fenêtre de temps sur chaque sommets (client). Il est appliqué généralement sur le problème de tournée de réparateur. Bar-Yehuda *et al.* (2005) ont proposé encore des approches heuristiques pour le cas symétrique et asymétrique du PVCSFT. Gimadi *et al.* (2004) ont introduit un problème plus généralisé que le PVC, il consiste à maximiser le profit récolté par une tournée après des actions d'achat et de vente. La résolution de ce problème a été effectuée, après sa réduction à un PVCS, avec une heuristique à temps d'exécution polynomial. Gimadi *et al.* (2004) ont montré que les solutions fournies, par leur algorithme sur des cas spéciaux, sont optimales ou asymptotiquement optimales. Aráoz *et al.* (2006) ont résolu un cas particulier du Problème du Facteur Rural PFR avec une heuristique. Le PFR est un problème voisin du PVCS mais en considérant des profit sur les arêtes et non pas sur les sommets.

Pour résoudre le PVCS de façons exacte, Laporte et Martello (1990) ont proposé une définition formelle du PVCS ainsi qu'un ensemble de propriétés et de méthodes de résolution exacte et approchée de ce problème. Dès l'apparition de ces travaux, l'appellation de PVCS est plus utilisée dans la littérature. Tous les résultats obtenus, par la résolution du PVCS avec les différentes méthodes déjà mentionnées, ont été améliorés par un algorithme exacte proposé par Ramesh *et al.* (1992). Chao (1993) a proposé des solutions comparables à celles de Ramesh *et al.* (1992). Leifer et Rosenwein (1994) ont utilisé une relaxation forte, sur certaines contraintes, pour fournir des résultats comparables aux précédentes. (Gendreau *et al.*, 1998a) ont proposé une méthode exacte pour la résolution du PVCS. Elle est basée sur un algorithme de branchements et de coupes (Branch-and-Cut) et elle a fourni des solutions optimales pour toutes les instances jusqu'à $n = 300$. Fischetti *et al.* (1998) ont simplifié l'heuristique proposée par Ramesh et Brown (1991) afin de l'adapter dans une méthode exacte basée sur un algorithme de branchements et de coupes. Cette méthodes a fourni des solutions optimales sur des instances réelles et aléatoires jusqu'à $n = 500$. Kataoka *et al.* (1998) ont proposé un algorithme qui combine deux méthodes pour améliorer la borne inférieure du PVCS. L'une est basée sur la méthode de simplexe duale et l'autre sur la relaxation lagrangienne. Deitch et

Ladany (2000) ont résolu un problème de tournée de bus à une seule période avec la combinaison d'une heuristique du PVCS et un algorithme d'amélioration.

Pour la classe des PTQ, Awerbuch *et al.* (1998) ont introduit un problème dans lequel un vendeur doit vendre un quota de brosse (borne inférieure sur le gain) en minimisant la longueur de son parcours. Ce problème est appelé le Problème du Voyageur de Commerce avec Quota PVCQ. Les auteurs proposent un algorithme avec garantie de performance pour la résolution du PVCQ. Feillet (2001) a montré que la résolution du PVCQ peut être effectuée par des procédures de résolution conçues pour la résolution du Problème du voyageur de Commerce avec Collecte de Gains PVCCG. Le PVCCG est introduit par Balas et Martin (1985) comme modèle pour la planification quotidienne des opérations d'une aciérie. Fischetti et Toth (1988) ont mentionné que le PVCCG permet de modéliser la problématique d'une usine ayant besoin d'une certaine quantité d'un produit fourni par divers fournisseurs en prix et quantité variables. D'autres travaux ont des liaisons avec le PVCCG tels que (Balas, 1989, 1995, 1999, 2002), (Pekny et Miller, 1990), (Kubo et Kasugai, 1992), (Gothe-Lundgren *et al.*, 1995), (Kabadi et Punnen, 1996), (Dell'Amico *et al.*, 1998) et (Gueguen, 1999).

Concernant la classe des Problèmes de Tournées Profitables PTP, les premiers travaux ont été proposés par Keller (1985, 1989b). Les auteurs ont introduit le problème comme étant multi-objectifs par contre ils proposent des approches de résolution séquentielles pour des versions mono-objectif du problème. Les premiers travaux sur l'aspect bi-objectifs de ce type de problèmes ont été proposés récemment par Jozefowiez *et al.* (2006, 2007) appelé le Problème de Voyageur de Commerce Profitable PVCP. Leur approche est basée sur un processus de chaîne d'éjection (Glover, 1996) hybridé avec un algorithme évolutionnaire multi-objectif utilisant une contrainte surrogate (Glover, 1968, 1977; Glover et Laguna, 1993). Un autre problème , connu sous le nom du "Prize Collecting Traveling Salesman Problem" et il peut être classé sous les PTP, a été traité récemment par Bérubé *et al.* (2007).

Dans la suite, nous présentons une revue de la littérature liée directement au PmTS.

1.5.2 Littérature du PmTS

1.5.2.1 Méthodes de résolution du PmTS proposées

Le PmTS est mentionné dans la littérature, pour la première fois, dans un article de Butt et Cavalier (1994) sous la dénomination "The Multiple Tour Maximum Collection Problem". Chao *et al.* (1996b) ont renommé le PmTS le "Team Orienteering Problem" TOP. Les derniers auteurs ont proposé deux extensions, séquentielle et parallèle, de l'heuristique stochastique proposée par Tsiligirides (1984) pour la résolution du OP (PVCS). Ils ont proposé de plus un algorithme à cinq étapes pour

la résolution du PmTS. Celui-ci débute par une étape d'initialisation pour obtenir une solution initiale au PmTS. Ensuite, des phases d'améliorations sont effectuées. Ces améliorations sont basées sur des échanges entre deux sommets (*Two-point exchange*) ou des mouvements d'un seul sommet (*One-point movement*). Sur chaque nouvelle solution produite par les phases précédentes, une étape de nettoyage (*Clean up*) est appliquée sur chaque tournée de la solution. Le but de cette étape est de réduire d'avantage les longueurs des tournées qui peut conduire à ajouter ensuite des sommets dans la solution. Elle est assurée par la procédure *2-opt* proposée par Lin (1965). Deux différentes phases de ré-initialisation sont appliquées pour générer d'autres solutions du PmTS. Elles consistent à éliminer, des tournées, un ensemble de sommets de moindre gain ou de moindre ratio du gain sur coût.

Tang et Miller-Hooks (2005) ont proposé une heuristique tabou pour la résolution du PmTS. La procédure de recherche tabou proposée peut être expliquée en trois étapes. La première étape consiste en une initialisation par une procédure de mémoire adaptative (*Adaptive Memory Precedure*). La deuxième consiste à générer un certains nombre de solutions voisines de la solution actuelle et ensuite à les améliorer par des procédures spécifiques.

La procédure de recherche tabou est encapsulée dans une procédure de mémoire adaptative. Au début, un ensemble de solutions partielles est généré par des techniques heuristiques et est stocké dans la mémoire adaptative. Une solution initiale est construite en combinant les solutions partielles sélectionnées possédant les meilleures gains. Les solutions maintenues dans la mémoire adaptative seront mise à jour en utilisant les solutions partielles de la solution initiale améliorée. Comme mis en évidence par Golden *et al.* (1997), la procédure de mémoire adaptative est similaire à la population de solutions dans les algorithmes génétiques. La différence vient du fait que la solution initiale peut être générée par plus de deux parents. Les tournées initiales sont générées par une procédure d'insertion modifiée, appelée (*cheapest insertion*), proposée pour le TSP par Rosenkrantz *et al.* (1974).

La structure de voisinage utilisée par Tang et Miller-Hooks (2005), consiste à choisir des solutions différentes de la solution actuelle. La différence entre deux solutions est définie par le fait qu'elles n'ont pas le même ensemble de sommets. Si deux solutions ont le même ensemble de sommets mais d'ordres différents, elles ne sont pas considérées comme des solutions voisines. Cette technique a été introduite par Gendreau *et al.* (1994) pour la résolution du PTV et a été testée dans plusieurs expérimentations dans différents contextes.

Afin d'améliorer les tournées d'une solution courante, plusieurs procédures d'amélioration sont utilisées dans cette heuristique, telles que l'équilibrage entre les tournées longues et courtes, l'échange des sommets entre deux tournées et l'insertion de sommets aléatoires (Hart et Shogan, 1987). L'évaluation d'une solution est effectuée par une fonction objectif pénalisée. Cette fonction, introduite par Gendreau

et al. (1994), permet à la procédure de recherche de considérer à la fois des solutions réalisables et irréalisables.

Archetti *et al.* (2007b) ont proposé quatre heuristiques pour la résolution du PmTS. Les deux premières heuristiques sont basées sur la recherche avec tabou, sans et avec pénalité. Les deux dernières sont basées sur la Recherche à Voisinages Variables RVV. Mise à part quelques différences, les composants des deux heuristiques basées sur la recherche tabou, sont semblables à celles proposées par Tang et Miller-Hooks (2005). La solution initiale est générée par l'algorithme proposé par Chao *et al.* (1996b). La structure de voisinage est définie par une procédure appelée (*Jumps*). Elle consiste à passer d'une solution courante à une solution voisine en utilisant des procédures d'échanges de sommets entre les tournées et au niveau d'une seule tournée, et ensuite à appliquer une procédure tabou pour améliorer la solution voisine générée. Afin d'osciller entre les parties de l'espace de recherche réalisables et irréalisables, plusieurs fonctions objectifs ont été utilisées selon la nature de la solution courante (réalisable, irréalisable, admissible ou inadmissible).

Dans l'heuristique basée sur la RVV, l'examen de solutions irréalisables est autorisée. Pour pouvoir rétablir l'admissibilité, des procédures de correction sont effectuées. Cette heuristique à été testée sous deux variantes (*rapide* et *lente*).

La plupart des travaux de résolution du PmTS, dans la littérature, sont basées sur des algorithmes approchés. Boussier *et al.* (2007) ont proposée une méthode exacte, basée sur la technique de génération de colonnes, pour la résolution du PmTS.

1.5.2.2 Comparaison des méthodes proposées dans la littérature

Chao *et al.* (1996b) ont proposé un jeux de 353 instances pour évaluer leur méthode de résolution du PmTS. En comparant les résultats de l'heuristique proposée par Chao *et al.* (1996b) par rapport celle de Tsiligirides (1984) modifiée sur ces instances, elle produit la meilleure solution pour 60% des instances, la plus mauvaise solution pour 7% des instances et la même solution pour 33% des instances.

Les résultats obtenus par la méthode de Tang et Miller-Hooks (2005) ont été comparés par rapport aux résultats obtenus par l'heuristique de Chao *et al.* (1996b) et celle de Tsiligirides (1984) adaptée au TOP par Chao *et al.* (1996b). L'heuristique proposée a fourni 55 meilleures solutions non déjà connues contre 12 solutions moins bonnes que les meilleures solutions déjà connues.

Les heuristiques proposées par Archetti *et al.* (2007b) ont été testées sur les mêmes instances benchmark publiées dans (Chao *et al.*, 1996b). Les résultats obtenus ont été comparés par rapport aux résultats obtenus par Chao *et al.* (1996b) et Tang et Miller-Hooks (2005). Elle a fourni 199 nouvelles meilleures solutions. L'heuristiques *lente* basée sur la RVV est considérée comme la meilleure au point de vue qualité de solutions. Au point de vue du temps CPU, l'heuristique *rapide* basées

sur la RVV est considérée comme la meilleure. Elle a été choisie comme le meilleure compromis entre qualité de solution et temps d'exécution.

Finalement, la méthode exacte proposée par Boussier *et al.* (2007) a permis d'obtenir les solutions optimales sur les instances du benchmark, publiées par Chao *et al.* (1996b), pour $n = 21, 32$. Pour les autres instances, elle a fourni la solution optimale pour des longueurs des tournées maximales L relativement petites.

1.6 Conclusion

Dans ce chapitre, nous avons présenté le Problème de m-Tournées Sélectives PmTS ainsi que ses propriétés et caractéristiques. Ensuite, nous avons modélisé le PmTS sous forme d'un graphe et sous forme d'un programme linéaire en 0-1. Enfin, nous avons effectué une revue de la littérature du PmTS et celle des problèmes voisins. Nous avons présenté, dans cette partie, un aperçu rapide des techniques et méthodes déjà utilisées pour résoudre le PmTS.

Nous avons constaté que le PmTS est un problème NP-difficile. Sa résolution de façon exacte n'est pas actuellement possible pour des instances de grande taille. La majorité des travaux, existants dans la littérature, sont basées sur des méthodes approchées.

Dans les chapitres suivants, nous allons décrire les méthodes que nous avons proposées pour résoudre le PmTS. Nous montrons aussi comment la combinaison des différentes approches permet d'améliorer les comportement de chacune d'elles.

Chapitre 2

Approche à deux phases basée sur une méthode de classification faisant appel à la descente à voisinages variables

2.1 Introduction

Dans ce chapitre, nous présentons une méthode à deux phases pour résoudre PmTS basée sur la résolution d'un problème de classification et d'un problème d'élaboration des tournées. Nous définissons tout d'abord le problème de classification, puis nous présentons la techniques utilisée pour le résoudre. Plus précisément, cette première phase consiste à partitionner l'ensemble des sommets en m sous-ensembles de sommets distincts qui ne s'intersectent qu'aux deux sommets de départ et d'arrivée. Pour ce faire, nous avons utilisé une technique basée sur la recherche à voisinages variables (Hansen et Mladenović, 2001b). Elle consiste à construire une partition initiale et à effectuer ensuite des améliorations en échangeant ou déplaçant k sommets entre les classes. Suite à la description de cette technique nous présentons les résultats numériques obtenus lors de son utilisation au sein de la méthode à deux phases. Nous comparons cette méthode avec d'autres méthodes de la littérature.

2.2 Principe général de la méthode à deux phases

La méthode à deux phases suit le principe de "*Cluster first - Route second*". Ce principe a été utilisé fréquemment, dans la littérature, pour résoudre le PTV selon différentes approches (Wren et Holliday, 1972; Gillet et Miller, 1974; Fisher et Jaikumar, 1981; Taillard., 1993; Ryan *et al.*, 1993; Osman, 1993; Renaud et Boctor,

2002). Il est à noter qu'il existe, dans la littérature, le principe inverse appelé "*Route first - Cluster second*" utilisé encore pour la résolution du PTV (Beasley, 1983; Hachicha *et al.*, 2000; Prins, 2004; Amaya *et al.*, 2006).

Dans le cas du PmTS, une telle méthode consiste, en premier lieu, à partitionner l'ensemble des clients (sommets) en sous-ensembles (classes) de clients, puis à appliquer une procédure de routage sur chaque classe pour avoir enfin une solution au PmTS. La procédure de routage consiste à résoudre le Problème du Voyageur de Commerce Sélectif PVCS (Gendreau *et al.*, 1998b; Feillet *et al.*, 2005) sur chaque sous-ensemble de sommets. La diversification des partitions de classes permet à la méthode à deux phases de fournir plusieurs solutions au problème en appliquant, alternativement, les procédures de classification et de routage. L'algorithme 1 présente le schéma général de la méthode à deux phases.

Nous nous intéressons dans ce chapitre, uniquement, aux techniques utilisées dans la première phase (techniques de classification). Concernant la deuxième phase nous nous limitons à appliquer une procédure tabou pour la résolution du PVCS proposée par Gendreau *et al.* (1998b). Le problème à ce niveau se résume par le fait que cette procédure fournit, sur chaque classe de sommet, une solution pour le PVCS sous forme d'un cycle fermé. Afin de remédier à ce problème, nous avons fixé un gain infini p_∞ sur les dépôts v_1 et v_n et une distance nulle entre v_1 et v_n. La procédure de résolution du PVCS choisit obligatoirement les deux sommets v_1 et v_n dans la solution de façon adjacente puisque leurs gains sont élevés et qu'il a une distance nulle entre eux. A la fin, il ne reste que de retrancher la quantité $2mp_\infty$ du gain total de la solution du PmTS obtenue.

La variable $iterMax$ définit le nombre des itérations des deux phases. Bien évidement, le paramétrage de cette variable influe sur la qualité de la solution obtenue ainsi que sur le temps d'exécution de l'algorithme. La fixation de ce paramètre dépend du temps d'exécution nécessaire de chacune des deux phases et par suite de la taille de l'instance du problème.

Algorithme 1 Schéma général de la méthode à deux phases pour le PmTS

Étape 1 : $iter = 0$, initialiser $iterMax$.
Étape 2 : Répéter les étapes de 3 à 4 jusqu'à $iter = iterMax$.
Étape 3 : **Phase1** Générer une partition en classes de l'ensemble des clients à l'aide d'une méthode de classification spécifique.
Étape 4 : **Phase2**
- Appliquer une procédure de routage sur chaque classe de sommets générée par l'étape 3,
- iter++.

2.3 Le problème de classification

2.3.1 Définitions

La classification est une problématique importante dans l'analyse et l'exploration des données. Elle est présentée dans de nombreux domaines d'application tels que la finance, le marketing, le diagnostic médical, etc. Selon le petit Larousse, le mot classification signifie "*la distribution par classes, par catégories, selon un certain ordre et une certaine méthode*". Généralement, la classification consiste à construire, à partir d'un ensemble d'objets, des sous-ensembles distincts dont le contenu de chacun vérifie certaines contraintes d'intégrité.

Dans la littérature, nous pouvons trouver le terme *groupage* (ou *clustering* selon la terminologie anglaise) au lieu de classification. Même si ces deux termes sont similaire sur le plan linguistique, ils sont différents sur le plan technique. En effet, les techniques de classification utilisent une approche supervisée, c'est à dire, les objets sont affectés à un certain nombre de classes connues auparavant. Par contre, les techniques de groupage utilisent une approche non supervisée. Elle consiste à chercher un certain nombre de groupes potentiels dont les objets de chacun seront plus similaires entre eux que par rapport aux objets des autres groupes. Des survols de ces techniques existantes ont été réalisées par (Dubes, 1987; Jain et Dubes, 1988; Brandes *et al.*, 1989; Jain *et al.*, 1999, 2004; Essaquote *et al.*, 2005)

2.3.2 Formulation générique

D'une manière générale le problème de classification peut être défini comme suit : Soit un ensemble de n objets $O = \{o_1, o_2, ..., o_n\}$. Chaque objet o_i est caractérisé par un ensemble de p critères $C_i = \{c_{i1}, c_{i2}, ..., c_{ip}\}$. Le problème de classification consiste à diviser l'ensemble d'objets O en m sous-ensembles O_r (*classes*) distinctes, avec m est connu auparavant. Les objets d'une même classe O_r doivent avoir une certaine *similarité* définie par l'ensemble des évaluations $C = \{C_1, C_2, ..., C_n\}$.
$\{O_1, \ldots, O_m\}$ doit constituer une partition de O c'est à dire respecter les trois conditions suivantes :

$$\forall r \in \{1, ..., m\}, O_r \subset O \tag{2.1}$$

$$\forall r, r' \in \{1, ..., m\}, r \neq r', O_r \cap O_{r'} = \oslash \tag{2.2}$$

$$\sum_{r=1}^{m} |O_r| = n \tag{2.3}$$

2.3.3 Formulation spécifique

Dans le cas du Problème de m-Tournées Sélectives PmTS, nous définissons le problème comme suit :
Soit $G = (V, A)$ le graphe complet et non orienté sur lequel nous allons résoudre le PmTS où $V = \{v_1, v_2, ..., v_n\}$ est un ensemble de n sommets. v_1 et v_n représentent, respectivement, le point de départ et le point d'arrivée de chaque tournée construite et A est l'ensemble des arêtes. Soient $p_i > 0$ le gain associé à chaque sommet $v_i \in V$ (avec $p_1 = p_n = 0$) et $d_{ij} > 0$ la distance associée à chaque arête $(v_i, v_j) \in A$.

Appliquer une technique de classification pour la méthode à deux phases consiste à construire m classes de sommets. Chaque classe contient les sommets v_1 et v_n et doit être le plus compacte que possible du point de vue du profit et de la distance. Autrement dit, il faut tenir compte de la maximisation du profit résultant et de la minimisation de la distance totale, respectivement, des sommets et des arêtes associés aux sommets de chaque classe (voir figure 2.1).

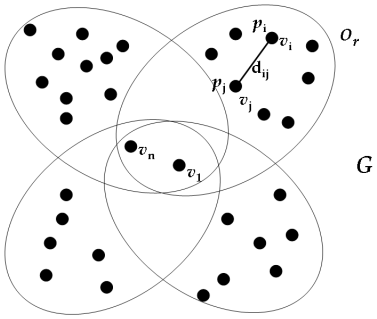

FIGURE 2.1 – Exemple de classification spécifique pour $k = 4$

Nous avons formulé le problème de classification comme suit. Déterminer $\{O_1, \ldots, O_m\}$ tel que :

$$\min \sum_{r=1}^{m} \frac{\sum_{i,j \in O_r} d_{ij}}{(|O_r| - 1) \sum_{i \in O_r} p_i} \tag{2.4}$$

$$\forall r \in \{1, \ldots, m\}, O_r \subset V \tag{2.5}$$

$$\forall r, r' \in \{1, \ldots, m\}, r \neq r', O_r \cap O_{r'} = \{v_1, v_n\} \tag{2.6}$$

$$\sum_{r=1}^{m} |O_r| = n + 2(m - 1) \tag{2.7}$$

L'équation (2.4) est la fonction objectif, elle consiste à maximiser la somme des gains moyens récolté par les m classes et de minimiser la somme des distances moyennes

entre les sommets de chaque classe. Notons qu'une autre fonction objectif pouvant être utilisé est la fonction suivante ($\max \sum_{r=1}^{m} \frac{(|O_r|-1)\sum_{i\in O_r} p_i}{\sum_{i,j\in O_r} d_{ij}}$). La contrainte (2.5) assure que chaque classe O_r soit un sous ensemble de sommets de G. L'équation (2.6) assure que l'ensemble $\{v_1, v_n\}$ représente la seule intersection entre toutes les classes. L'équation (2.7) garantit que la somme des cardinalités des classes soit exactement égale à $n + 2(m-1)$.

2.4 Recherche à voisinage variable

Dans cette section nous présentons une approche de classification, basée sur la recherche à voisinage variable RVV (Mladenović et Hansen, 1997; Hansen et Mladenović, 1999, 2001b) qui a été proposée comme une extension de la *recherche locale* standard.

Le schéma de base de la *descente simple* ou *recherche locale* est représenté par l'algorithme 2. Cette heuristique s'arête aussitôt qu'un optimum local est atteint. Elle utilise un seul type de mouvement, ou, ce qui est équivalent, une seule structure de voisinage.

Une première amélioration de la recherche locale est la *descente multiple*, la descente

Algorithme 2 Schéma général de la recherche locale

Étape 1 : Sélectionner une structure de voisinage N qui sera utilisée dans la recherche : Identifier une solution initiale x.
Étape 2 : Trouver le meilleur voisin $x' \in N(x)$ de x ;
Étape 3 : Si x' n'est pas meilleur que x, fin. Sinon, poser $x = x'$ et retourner en 2.

simple est alors réitérée plusieurs fois à partir de solutions initiales tirées au hasard et la meilleure solution obtenue est conservée. Il est cependant clair que la descente multiple disperse ses efforts en explorant de nombreuses régions sans se concentrer sur les régions les plus prometteuses. Il est donc important, après avoir découvert un optimum local, d'exploiter l'information ainsi obtenue. La descente multiple peut alors être modifiée pour explorer les voisinage proches et de plus en plus éloignés. C'est précisément ce qui est fait par la recherche à voisinage variable, lorsqu'elle utilise des structures de voisinages les plus simples, c'est à dire des voisinages imbriqués.

Nous présentons dans l'algorithme 3, une expression générale de la recherche à voisinage variable (Mladenović et Hansen, 1997; Hansen et Mladenović, 1999, 2001b). Notons par $N_k\,(k = 1, 2, \ldots, k_{max})$ un ensemble fini de structures de voisinages présélectionnées et par $N_k\,(x)$ l'ensemble de solutions du k^{ime} voisinage de x.

Nous pouvons choisir comme condition d'arrêt, parmi d'autres, le temps CPU maximal alloué, le nombre maximal d'itérations ou le nombre maximal d'itérations depuis la dernière amélioration de la meilleure valeur connue.

Un moyen facile d'obtenir un ensemble de structures de voisinage est de considérer un mouvement élémentaire et de définir $N_k(x)$ comme l'ensemble des solutions obtenus en appliquant successivement k mouvements à x (sans mouvement inverse).

Le schéma de l'algorithme 3 utilise une méthode de recherche locale simple. Cette méthode peut être remplacée par une méthode complexe afin d'avoir une meilleure exploration du voisinage. De plus l'exploration de voisinage peut être rapide ou lente selon la simplicité ou la complexité des mouvements définis. La méthode de recherche locale *descente simple* de l'algorithme 3 peut être remplacée par *une descente à voisinage variable* DVV décrite par l'algorithme 4.

Algorithme 3 Schéma de base de la recherche à voisinage variable

Étape 1 : Sélectionner l'ensemble de structure de voisinages N_k (avec $k = 1, 2, \ldots, k_{max}$) qui sera utilisé dans la recherche.
Étape 2 : Déterminer (ou choisir) une solution initiale x.
Étape 3 : Choisir une condition d'arrêt.
Étape 4 : Répéter Les étapes de 5 à 6 jusqu'à ce que la condition d'arrêt soit satisfaite.
Étape 5 : Poser $k = 1$.
Étape 6 : Répéter les étapes de 7 à 9 jusqu'à ce que $k = k_{max}$.
Étape 7 : Perturbation : Générer un point x' au hasard dans le k^{ime} voisinage de x ($x' \in N_k(x)$).
Étape 8 : Recherche locale : Appliquer une méthode de recherche locale avec x' comme solution initiale ; soit x" > l'optimum local ainsi obtenu.
Étape 9 : Intensification ou changement de voisinage : Si l'optimum local est meilleur que la meilleure solution connue, recentrer la recherche à ce point ($x \leftarrow x$") et retourner au premier voisinage $N_1(k \leftarrow 1)$; sinon, poser $k \leftarrow k + 1$.

2.5 Méthode de classification par la recherche à voisinage variable

Notre méthode de classification à voisinage variable consiste à générer à partir de l'ensemble des sommets du graphe G un ensemble de m classes différentes. Le seul point en commun de ces classes sont les deux sommets de départ et d'arrivée v_1 et v_n (voir section 2.3.3). La procédure de classification sera ensuite combinée avec une procédure de routage pour résoudre le PmTS (voir l'algorithme 1).

Algorithme 4 Schéma de base de la descente à voisinage variable

Étape 1 : Sélectionner l'ensemble de structure de voisinages N'_k (avec $k = 1, 2, \ldots, k'_{max}$) qui seront utilisées dans la descente.
Étape 2 : Déterminer une solution initiale x (si elle n'est pas déjà fournie).
Étape 3 : Répéter les étapes de 4 à 5 jusqu'à ce qu'il n'y ait plus d'améliorations.
Étape 4 : Poser $k = 1$.
Étape 5 : Répéter les étapes de 6 à 7 jusqu'à ce que $k = k_{max}$.
Étape 6 : Exploration du voisinage. Déterminer le meilleur voisin x' du point x ($x' \in N'_k(x)$).
Étape 7 : Si la solution x' ainsi obtenue est meilleure que la solution x précédente , poser $(x \leftarrow x')$; sinon, poser $k \leftarrow k + 1$.

Le principe général de notre méthode de classification est basé sur une descente à voisinage variable (voir Algorithme 5). Les rôles des différentes étapes sont décrites dans la suite.

Algorithme 5 Schéma général de la méthode de classification à voisinage variable

Étape 1 : Construire sur le graphe G une partition initiale de m classes en utilisant une approche gloutonne.
Étape 2 : Améliorer la partition initiale par une méthode de recherche locale.
Étape 3 : Répéter les étapes de 4 à 6 tant que la condition d'arrêt n'est pas vérifiée.
- *Étape 4 :* Évaluer la partition actuelle et mettre à jour, si nécessaire, la meilleure partition trouvée.
- *Étape 5 :* Générer une nouvelle partition choisie parmi les solutions définies par la structure du voisinage utilisée.
- *Étape 6 :* Améliorer la partition actuelle par une méthode de recherche locale.

2.5.1 Construction de la partition de classes initiale

La méthode de recherche à voisinage variable commence par une partition initiale de m classes. Ces classes sont obtenues par une méthode gloutonne. Elle consiste à insérer les sommets v_1 et v_n dans chaque classe. Ensuite, les m classes sont construites en parallèle en insérant, à tour de rôle, deux sommets du graphe à chaque classe. A chaque étape, les deux sommets les plus proches de la classe concernée sont insérés. Ce processus se termine lorsque chaque sommet de $V - \{v_1, v_n\}$ sera affecté à une classe.

2.5.2 Évaluation d'une partition de classes

Durant le procédure de recherche à voisinage variable, la qualité d'une partition de classes doit être déterminée. En conséquence, nous devons évaluer la compacité de chaque classe en tenant compte de sa distance moyenne et de son gain (profit) total de ses sommets.
Pour cela nous avons adapté l'index de dispersion ξ introduit par Tricot et Donegani (1989) pour calculer la distance moyenne d'un ensemble de sommets O. Cet index est définit comme suit :

$$\xi(O) = \begin{cases} \frac{1}{|O|(|O|-1)} \sum_{v_i, v_j \in O} d_{ij} & \text{si } |O| > 1 \\ 0 & \text{si } |O| = 1 \end{cases}$$

Pour notre cas, l'évaluation de la compacité d'une classe en tenant compte uniquement de la distance moyenne ne suffit pas. Donc, et pour tenir compte du gain récolté par chaque classe O_r, nous avons introduit l'index de dispersion $\gamma(O_r)$. Cet index combine la distance moyenne d'une classe (*i.e.* la compacité de la classe) et son profit moyen. Nous avons :

$$\gamma(O_r) = \begin{cases} \frac{\sum_{v_i, v_j \in O_r} d_{ij}}{(|O_r|-1) \sum_{v_i \in O_r} p_i} & \text{si } |O_r| > 1 \\ 0 & \text{si } |O_r| = 1 \end{cases}$$

En minimisant l'index $\gamma(O_r)$ de chaque classe O_r nous obtenons une classe compacte avec un profit total élevé. La valeur de l'index total γ des m classes sera calculée pour évaluer la compacité totale de chaque partition obtenue, $\gamma = \sum_{r=1}^{m} \gamma(O_r)$.

2.5.3 Structure du voisinage

Soit une partition de m classes, les voisins sont des partitions obtenues par élimination de k sommets de chaque classe et leur insertion dans une autre classe. Plus précisément, nous déterminons pour chaque classe O_r les deux plus proches classes O_{r1} et O_{r2} (i.e. les classes dont leurs centres de gravité sont le plus proches du centre de gravité de la classe O_r). Les k sommets de O_r, qui sont les plus proches de O_{r1} sont éliminés de O_r et insérés dans O_{r1}. Nous procédons, de la même façon pour éliminer k sommets de O_{r2} et les insérer dans O_r.

En appliquant tous les échanges de sommets, une nouvelle partition de m classes est générée à partir de laquelle la méthode de recherche locale, décrite dans la section suivante, peut être appliquée. Puisque les mouvements effectués dans la recherche

locale peuvent ramener à la décomposition initiale, les sommets déplacés en utilisant cette structure de voisinage ne sont pas considérés au cours de la recherche locale. En effet, ils sont déclarés interdits et sont stockés dans une liste tabou.

2.5.4 Méthode de recherche locale

Pour une partition de m classes, nous appliquons une méthode de recherche locale pour améliorer la valeur de l'index total. Dans la recherche locale, les voisinages d'une partitions sont obtenus par une *déplacement* d'un sommet d'une classe à une autre ou bien par une *permutation* de deux sommets entre deux classes. Chaque nouvelle partition obtenue par ces mouvements, de mutation ou de migration, sera évaluée par son index de dispersion total. Dès que l'index total diminue, la nouvelle partition correspondante est considérée. Plus précisément :

- *Mouvement de permutation :* deux sommets entre deux classes sont échangés afin de décroître la somme des valeurs de leurs index. Si cela n'est pas le cas, le mouvement de permutation est annulé.
- *Mouvement de déplacement :* un sommet est éliminé d'une classe et inséré dans une autre. Ce mouvement est validé si la somme des valeurs de leurs index diminue et si les classes ont presque la même cardinalité. Plus précisément et pour deux classes O_i et O_j, la différence entre leurs cardinalités ne doit pas dépasser une valeur fixée ϵ (*i.e.* $||O_i| - |O_j|| \leq \epsilon$). Nous avons imposé cette condition afin de ne pas avoir un déséquilibre entre les cardinalités des classes et de ne pas avoir, au pire des cas, des classes vides.

Ces mouvements sont effectués sur les sommets non interdits de $V - \{v_1, v_n\}$. En effet, les sommets interdits sont mémorisés dans une liste tabou qui sera décrite dans la sous-section suivante. Pour une itération de la recherche locale, le mouvement de déplacement est évalué en premier lieu. Si la valeur de l'index total obtenue n'est pas améliorée, le mouvement de déplacement est considéré. La procédure de recherche locale s'arête lorsque la valeur de l'index total ne peut plus être améliorée.

2.5.5 Taille de voisinage

Nous avons choisi la taille de voisinage k de façon à éviter le cas d'une permutation ou d'un déplacement d'un grand nombre de sommets entre les classes. Pour cela, nous avons fixé un nombre maximal de sommets échangés $k_{max} = \frac{n}{2m}$. Pour un déplacement ou une permutation, le nombre maximal de sommets échangés ne peut pas dépasser environ la moitié de la cardinalité d'une classe. En effet, si nous dépassons k_{max} échanges ou déplacements de sommets entre les classes nous pouvons revenir à une partition déjà rencontrée. Par exemple, pour une partition donnée et si nous effectuons les déplacements de κ sommets de chaque classes avec

$k_{max} < \kappa \approx |O_i|, \forall i \in \{1, ..., m\}$, nous pouvons considérer de nouveau la même partition que précédemment.

2.5.6 Nombre maximal d'itérations

Nous avons mentionné auparavant que le nombre d'applications de la phase de classification influe sur la qualité de la solution du PmTS. Le nombre des itérations dépend, aussi, de la taille du problème à résoudre. Ainsi, nous avons choisi de limiter le nombre maximal d'itération $IterMax = \frac{n}{k \times m}$. L'intervention de la taille de voisinage k assure que plus la taille du voisinage est petite plus l'exploration du voisinage est intense, et inversement dans le cas contraire.

2.5.7 Description de l'algorithme de classification à voisinage variable

Nous pouvons maintenant procéder à la description détaillée de l'algorithme de classification à voisinage variables. Pour être explicite, l'algorithme 6 présente l'hybridation de la procédure de classification avec celle de routage (Gendreau *et al.*, 1998b), notée RTT, afin de résoudre le PmTS.

2.6 Résultats numériques

Afin de valider la méthode de classification par la recherche à voisinage variable, nous avons testé l'heuristique (décrite à la section 2.5) sur 353 instances benchmark du PmTS, publiées par Chao et al. (Chao *et al.*, 1996b), de différentes tailles ($n = 102, 100, 66, 64, 62, 33, 32, 21$) et pour $m = 2, 3, 4$ tournées. Pour n et m fixés un sous-ensembles d'instances est obtenu en variant la longueur maximale L d'une tournée.

Lors de l'examen des résultats numériques obtenus et en les comparant avec ceux retrouvés par Chao *et al.* (1996b), Tang et Miller-Hooks (2005) et Archetti *et al.* (2007b), nous avons constaté que les solutions sont compétitives pour les instances de $n = 102$ et $n = 66$. En visualisant les graphes des différentes grandeurs, nous avons observé que les dépôts sont de différentes dispositions. Pour $n = 102$ les dépôts sont confondus et pour $n = 66$ ils sont proches, par contre pour toutes les autres instances les dépôts sont éloignés l'un de l'autre. Tout cela nous a permis d'affirmer la cause principale de l'hétérogénéité de la qualité des solutions obtenues est due à l'éloignement excessive des dépôts pour la majorité des instances.

En conséquence, la méthode de classification par la recherche à voisinage variable de la méthode à deux phases s'avère satisfaisante pour les instances à dépôts confondus ou proches (voir tableau 5.5 de Annexe1).

Algorithme 6 Schéma détaillé de la méthode de classification à voisinage variable

Étape 1 : Pré-traitement : Éliminer les sommets v_i qui ne peuvent pas figurer dans la solution finale tel que $(d_{1i} + d_{in} > L)$.
Étape 2 : Partitionnement initial : Construire une partition initiale en utilisant une heuristique gloutonne.
Étape 3 : Soient : p^* le meilleur profit, γ^* la meilleure valeur de l'index total et γ la valeur de l'index total courante.
Étape 4 : Initialiser $p^* = 0$, $\gamma^* = +\infty$ et $k = 1$.
Étape 5 : Amélioration de la partition : améliorer la partition initiale en moyen de la méthode de recherche locale décrite dans la section 2.5.
Étape 6 : Répéter les étapes de 7 à 8 pour $k = \{1, 2, ..., \frac{n}{2m}\}$.
Étape 7 : Soit $IterMax$ le nombre maximal d'appel à la recherche locale pour une valeur de k. $IterMax = \frac{n}{km}$.
Étape 8 : Répéter les étapes de 9 à 11 pour $IterMax$ itérations.
Étape 9 : Détermination des tournées : calculer la valeur de l'index total γ pour la partition courante. Si $\gamma^* > \gamma$, construire une tournée sur chaque classe au moyen de la procédure RTT et calculer le profit total p de la solution du PmTS associée.
Étape 10 : Mise à jour de la meilleure solution : si $p > p^*$, $p^* = p$, $IterMax = 1.2 \times IterMax$ et mémoriser la solution actuelle.
Étape 11 : Nouvelle partition : générer une nouvelle partition en échangeant k sommets entre chaque deux classes adjacentes comme expliqué dans la section 2.5. Améliorer la partition au moyen de la méthode de recherche locale décrite dans la section 2.5.

Le tableau 2.1 présente un résumé du tableau 5.5. Il présente les totaux des résultats numériques obtenus, sur 132 instances, par cette méthode (notée CRVV) et ceux obtenus par Chao *et al.* (1996b), Tang et Miller-Hooks (2005) et la méthode, basée sur la recherche à voisinage variable, considérée comme le meilleur compromis par Archetti *et al.* (2007b) notées (respectivement) par CGW, TMH et AHS.

Le tableau 2.2 représente les temps d'exécution maximal et moyens, sur les deux tailles d'instances, des différente méthodes à savoir :

- **CGW** : Les résultats de la méthode de (Chao *et al.*, 1996b).
- **TMH** : Les résultats de la méthode de (Tang et Miller-Hooks, 2005).
- **AHS** : Les résultats de valeur maximale après 3 exécutions de la méthode à voisinage variables rapide, considérée comme le meilleur compromis, de (Archetti *et al.*, 2007b).
- **CRVV** : Les résultats obtenus par notre méthode.

Les notations suivantes sont considérées dans le tableau 2.2 :

- n : le nombre de sommets y compris les dépôts,
- T_{moy} : le temps moyen d'exécution d'une instance de taille n correspondante,

– T_{max} : le temps maximal d'exécution d'une instance de taille n correspondante,

n	m	**CGW**	**TMH**	**AHS**	**CRVV**
100	4	9450	9778	9822	9695
	3	11124	11258	11344	11010
	2	12678	12669	12812	12646
66	4	16705	16775	17010	16300
	3	19415	19395	19590	19300
	2	22265	22170	22395	21590

TABLE 2.1 – *Résultats numériques moyens de la méthode à deux phases avec CRVV*

	n	T_{moy}	T_{max}
CGW [a]	100	*	841,4
TMH [b]		84,4	432,6
AHS [c]		10,3	90
CRVV [d]		225,5	449,1
CGW	66	*	193,7
TMH		16,8	71,3
AHS		34,2	30
CRVV		73,9	360,6

a. SUN 4/370 Workstation
b. DEC Alpha XP1000 Computer
c. Intel Pentium 4, 2.8 Ghz, 1048 Mo Ram
d. Intel Pentium 4, 2.4 Ghz, 256 Mo Ram

TABLE 2.2 – *Temps d'exécution moyens et maximums*

Nous remarquons qu'en général, les solutions obtenues par CRVV sont en moyenne moins bonnes que les solution obtenues par les autres méthodes. Cependant, en consultant le tableau 5.5 de l'Annexe 1 nous pouvons constater que sur les 132 instances, CRVV donne la même solution que TMH, CGW pour 30 instances et où elle fournit 26 fois la meilleure solution.

En comparant CRVV avec AHS qui est considérée comme la meilleure méthode, nous pouvons constater que CRVV fournit 18 fois une meilleure solution que AHS et qu'elles sont identiques pour 19 instances.

Au point de vue temps d'exécution, il est clair que la plupart des méthodes (CGW,TMH et AHS) fournissent des solutions au problème en un temps nettement inférieur à celui mis par CRVV. La différence entre les machines, sur lesquelles, les différentes méthodes, ont été testées (voir tableau 2.1), ne peut pas être une cause de cette différence de temps d'exécution. En effet, la diversité des voisinages

($k_{max} = \frac{n}{2m}$) et l'exploration intensive du voisinage ($IterMax = \frac{n}{km}$) effectués par CRVV nécessitent un temps d'exécution significatif lors du processus de classification (voir Algorithme 6).

Après l'analyse des résultats numériques obtenus par CRVV et mis à part les nouvelles solutions qu'elle a, nous avons remarqué qu'elle présente plusieurs inconvénients. En effet, d'une part elle est efficace uniquement pour un sous-ensemble d'instances dans lesquelles les dépôts sont confondus ou proches. D'autre part, CRVV requiert des temps CPU élevés par rapport aux autres méthodes.

Cela nous a donc conduit à considérer, pour résoudre le PmTS, d'autres méthodes de classification au sein de la méthode à deux phases.

2.7 Conclusion

Nous avons présenté, dans ce chapitre, une méthode à deux phases basée sur une technique de classification faisant appel à la recherche à voisinages variables CRVV. Cette méthode a fourni des bons résultats, pour résoudre le Problème de m-Tournées Sélectives PmTS, sur des instances de problème où les dépôts sont confondus ou proches. CRVV a fourni des solutions meilleures que les méthodes de Chao *et al.* (1996b) et Tang et Miller-Hooks (2005). Plusieurs nouvelles solutions ont été trouvées par rapport aux résultats d'Archetti *et al.* (2007b) qui est considérée comme la meilleure méthode de résolution du PmTS.

Ceci nous a conduit à considérer d'autres méthodes de classification permettant de fournir, à la phase de routage, des classes plus diversifiées. La description de ces différentes méthodes fait l'objet du chapitre suivant.

Les résultats obtenus par cette approche ont été publiés dans Khemakhem et Semet (2005); Khemakhem *et al.* (2005).

Chapitre 3

Approche à deux phases basée sur la méthode d'agrégation autour des centres mobiles

3.1 Introduction

Dans le chapitre précédent, nous avons présenté une méthode à deux phases, pour la résolution du PmTS, basée sur la classification à voisinage variable. Cette méthode a montré ses performances que pour un sous-ensemble d'instances où les dépôts sont proches ou confondus. Cet inconvénient, nous a conduit à proposer d'autres techniques de classification pour la méthode à deux phases.

Nous présentons dans ce chapitre, l'adaptation d'une technique de classification appelée *agrégation autour des centres mobiles* (Thorndike, 1953; MacQueen, 1967; Jain *et al.*, 1999; Hansen et Mladenović, 2001a; Likasa *et al.*, 2003). Nous avons utilisé cette méthode, au sein de la méthode à deux phases, avec trois métriques différentes à savoir les distance *Euclidienne*, de *Profilage* et de *Mahalanobis*. Après avoir définir cette méthode avec ces trois métriques, nous présentons les résultats numériques obtenus lors de leurs utilisations au sein de la méthode à deux phases que nous comparons à ceux obtenus par les méthodes classiques de la littérature.

3.2 Principe général de la méthode d'agrégation autour des centres mobiles

La méthode d'agrégation autour des centres mobiles est attribuée principalement à Forgy (1965), bien que de nombreux travaux (parfois antérieurs : Thorndike (1953)), le plus souvent postérieurs (MacQueen (1967) ; Ball et Hall (1967)) furent

menés parallèlement et indépendamment pour introduire des variantes ou des généralisations. Cette méthode peut être considérée comme un cas particulier de techniques connues sous le nom de nuées dynamiques étudiées dans un cadre formel par Diday (1971).

Cette méthode, connue aussi sous le nom de l'algorithme de *k-means* ou *k-médianes*, commence par un choix aléatoire de k objets parmi les objets d'un ensemble O. Les objets choisis sont considérés comme des centres q_r des classes en vue de leur construction. Ensuite, une phase d'affectation permet d'attribuer à chaque classe un sous-ensemble d'objets de O. Cette phase consiste à affecter chaque objet o_i à une classe O_r dont son centre q_r est le plus *adéquat* (le plus proche dans un contexte de distance) par rapport à o_i en tenant compte de l'évaluation de l'ensemble des critères C_i. Une fois que l'étape précédente est effectuée, le centre de chaque classe est mis à jour au moyen d'une *procédure spécifique* (détermination du barycentre dans un contexte de distance). Les deux étapes précédentes sont répétées jusqu'à ce que toutes les classes ne changent plus durant deux itérations consécutives. Nous obtenons alors une partition de k classes qui vérifient des exigences de *similarité* (la compacité dans un contexte de distance).

Il faut signaler que le choix des objets initiaux influe sur la partition obtenue par l'algorithme *k-médianes*, c'est à dire que deux choix différents des objets initiaux peut mener à deux partitions différentes. L'algorithme 7 et la figure 3.1 illustrent le fonctionnement de l'algorithme *k-médianes*. Nous expliquons la démarche de l'algorithme *k-médianes* selon la figure 3.1 comme suit :

① Cette étape consiste à choisir k centres initiaux, pour la figure $k = 2$.

② Cette étape consiste à affecter chaque sommet du graphe au plus proche centre. Cette affectation nous fourni une partition de classes initiale.

③ Cette étape consiste à identifier un nouveau centre à chaque classe construite à l'étape précédente. Les anciens centres seront ignorés.

④ Cette étape consiste à réaffecter chaque sommet à son nouveau plus proche centre.

Les deux dernière étapes sont répétées jusqu'à l'obtention de deux partitions successives identiques.

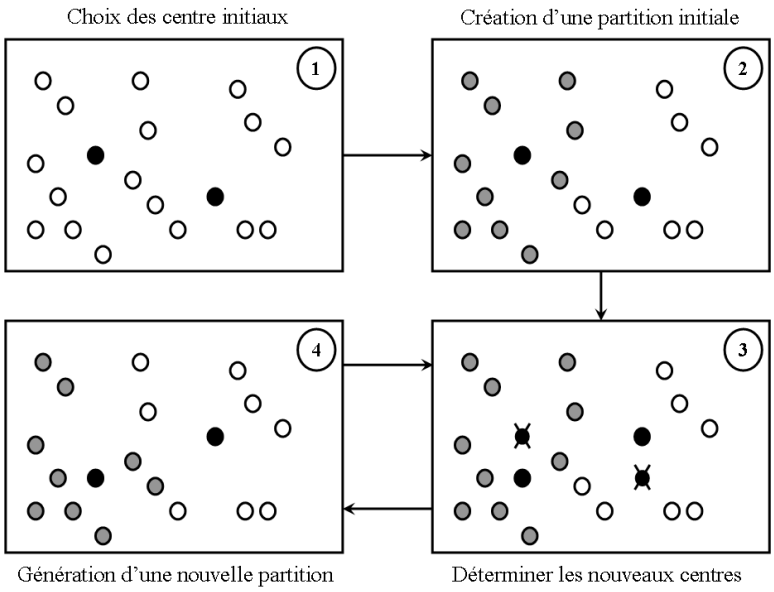

FIGURE 3.1 – Processus de l'algorithme *k-médianes* pour deux classes

Algorithme 7 Schéma général de l'algorithme *k-médianes*

Étape 1 : Choisir k centres $q_r \in O$ pour chaque classes O_r, avec $r \in \{1, \ldots, k\}$.
Étape 2 : Affecter chaque objet o_i de O à la classe O_r dont son centre q_r est le plus proche de o_i d'où la génération d'une partition P, $P' = \oslash$.
Répéter les étapes 3 à 5 Tant que $P \neq P'$
Étape 3 : Si $P' \neq \oslash$ faire $P = P'$. Calculer les centres de chaque classe O_r.
Étape 4 : Réaffecter de nouveau chaque objet o_i de O à une classe O_r dont le centre q_r est le plus proche de o_i d'où la génération d'une nouvelle partition P'.

Nous avons utilisé l'algorithme *k-médianes* afin de générer des classes de sommets pour la méthode à deux phases. Dans les sections suivantes nous détaillons l'algorithme *k-médianes* en l'adaptant au besoin de la phase de classification, ensuite, nous présentons nos choix d'initialisation et les métriques que nous avons utilisées pour tenir compte de la spécificité du PmTS ainsi que de la diversité de ses instances.

3.3 Algorithme *k-médianes* pour la méthode à deux phases

Nous avons défini, dans le chapitre précédent, le problème de classification spécifique au PmTS. En effet, dans le contexte du PmTS, la classification s'effectue

sur les n sommets d'un graphe G caractérisés par des gains p_i et des distances euclidiennes d_{ij}. Cette classification présente deux caractéristiques. D'une part, les k classes s'interfèrent au niveau des deux dépôts v_1 et v_n. D'autre part, la compacité d'une classe doit tenir compte des distances et des gains.

Nous avons donc à spécifier des éléments tels que le choix initial des sommets, la distance entre les sommets et le calcul des centres des classes dans notre adaptation de l'algorithme. L'algorithme générique de *k-médianes*, que nous avons utilisé pour assurer la phase de classification dans la méthode à deux phases pour résoudre le PmTS, est présenté par l'algorithme 8.

Algorithme 8 Schéma général de l'algorithme *k-médianes* pour la méthode à deux phases

Étape 1 : Choisir k sommets distincts $q_r \in V - \{v_1, v_n\}$ pour chaque classe O_r, avec $Q = \{q_1, \ldots, q_k\}$.
Étape 2 : Affecter les sommets v_1 et v_n à chaque classe O_r de façon définitive.
Étape 3 : Affecter chaque sommet v_i de $V - (Q \cup \{v_1, v_n\})$ à la classe O_r dont son centre q_r est le plus proche de v_i en terme de distance et de gain, d'où la génération d'une partition P, $P' = \oslash$.
Répéter les étapes 4 à 6 Tant que $P \neq P'$
Étape 4 : Si $P' \neq \oslash$ faire $P = P'$.
Étape 5 : Mettre à jour l'ensemble des centres Q.
Étape 6 : Mettre à jour les classe O_r en tenant compte des sommets de $(Q \cup \{v_1, v_n\})$, d'où la génération d'une nouvelle partition P'.

3.4 Algorithme *k-médianes* utilisant la métrique Euclidienne

La métrique la plus simple, pour appliquer l'algorithme *k-médianes* sur un graphe dont on connaît les coordonnées des sommets, est la distance Euclidienne. Nous présentons dans la suite les caractéristiques de cet algorithme à savoir le choix des centres initiaux, les distances entre les sommets et la détermination des centres des classes à chaque mise à jour.

3.4.1 Choix initial des centres

Une partition de classes générée par l'algorithme *k-médianes* dépend du choix initial des centres des classes. La variation des choix des centres initiaux des classes nous permet de diversifier les partitions de classes obtenues et par la suite celle des

solutions du PmTS. Lors de l'utilisation de la métrique actuelle nous avons choisi les centres initiaux parmi les sommets formant l'enveloppe convexe des sommets du graphe. Ce choix a été validé suite à des expérimentations qui nous ont permis de constater qu'il offre une convergence plus rapide à l'algorithme *k-médianes* et qu'il fournit des classes plus équilibrées. En effet, le choix de centres initiaux trop proches peut fournir une partition avec certaines classes quasi-vides.

3.4.2 La distance, index de proximité

Comme il a été déjà mentionné, nous avons utilisé dans cette approche la distance Euclidienne. Lors du choix des sommets candidats d'une classe et afin de tenir compte des contraintes distance et gain nous avons remplacé la distance Euclidienne par un index de proximité de_{ij} entre deux sommets v_i et v_j décrit comme suit :

$$de_{ij} = \frac{d_{ij}}{g_i+g_j} \quad \forall v_i, v_j \in V$$

Dans le cas de la compacité spatiale, l'algorithme *k-médianes* tend à déterminer les classes les plus compactes en minimisant la distance moyenne entre les sommets de chaque classe. Dans notre cas, vue la nature du PmTS, la distance est remplacée par un index de proximité défini comme un quotient entre la distance et le gain. Cela revient à minimiser la distance et maximiser le gain.

Puisque nous avons pris en considération, lors de la définition de l'index de proximité de_{ij}, la distance euclidienne et le gain, nous considérons les mêmes critères lors de la détermination d'un centre q_r d'une classe o_r. Il est à noter qu'un nouveau centre défini à chaque itération ne représentent pas un sommet du graphe. Nous le définissons par ses coordonnées xq_r et yq_r comme suit :

$$xq_r = \frac{\sum_{v_i \in o_r} g_i x_i}{\sum_{v_i \in o_r} g_i}, yq_r = \frac{\sum_{v_i \in o_r} g_i y_i}{\sum_{v_i \in o_r} g_i}$$

Après le calcul des coordonnées de chaque centre de classe, il faut mettre à jour les index de proximité des autres sommets du graphe par rapport à ces centres. Cet index de_{ir} doit tenir compte aussi du gain, nous l'avons défini comme suit :

$$de_{ir} = \frac{\sqrt{(x_i - xq_r)^2 + (y_i - yq_r)^2}}{g_i} \quad \forall v_i \in V, q_r \in o_r$$

3.5 Algorithme *k-médianes* utilisant la métrique de Mahalanobis

Nous avons également utilisé, dans l'algorithme *k-médianes*, la distance dite de Mahalanobis (P.C. Mahalanobis, 1936). Cette métrique correspond à une évaluation

de la vraisemblance entre les sommets. Dans la suite, nous présentons cette métrique, nous justifions son choix, et nous montrons son application dans l'algorithme *k-médianes* pour la première phase de notre méthode pour résoudre le PmTS.

3.5.1 La distance de Mahalanobis

Pour deux sommets $v_i(x_i, y_i)$ et $v_j(x_j, y_j)$ du graphe G, la distance de Mahalanobis dm_{ij} est définie comme suit :

$$dm_{ij} = \sqrt{\begin{pmatrix} x_i - x_j \\ y_i - y_j \end{pmatrix}^t COV^{-1} \begin{pmatrix} x_i - x_j \\ y_i - y_j \end{pmatrix}}$$

où COV est la matrice de covariance définit par :

$$COV = \begin{pmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{pmatrix}$$

$$\text{avec } cov(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \ / \ \overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \ , \ \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Supposons que la distance Euclidienne d'un sommet v_i par rapport à un sommet v_k est inférieure à la distance Euclidienne d'un sommet v_j par rapport au sommet v_k alors cette supposition n'est pas nécessairement vérifiée avec la distance de Mahalanobis. Inversement, supposons que la distance Mahalanobis d'un sommet v_i par rapport à un sommet v_k est inférieure à la distance Mahalanobis d'un sommet v_j par rapport au sommet v_k alors cette supposition n'est pas nécessairement vérifiée avec la distance de Euclidienne :

$$d_{ik} < d_{jk} \nLeftrightarrow dm_{ik} < dm_{jk}$$

En effet, la distance de Mahalanobis entre deux sommets prend en compte l'emplacement et la densité des autres sommets (voir figure 3.2).

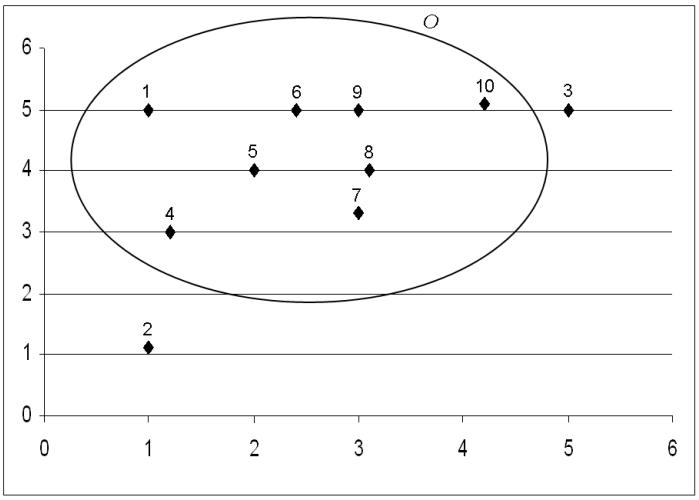

FIGURE 3.2 – Illustration de la distance de Mahalanobis

Dans la figure 3.2, considérant que nous sommes dans le cas d'un choix d'insertion d'un sommet dans une classe O de centre v_1. L'ensemble des sommets présenté dans la figure 3.2 est composé par les sommets $v_1, \ldots, v_{10}$ de coordonnées définies par la matrice *coord* suivante :

$$coord = \begin{pmatrix} 1 & 1 & 5 & 1,2 & 2 & 2,4 & 3 & 3,1 & 3 & 4,2 \\ 5 & 1,1 & 5 & 3 & 4 & 5 & 3,3 & 4 & 5 & 5,1 \end{pmatrix}$$

La classe O étant composée par l'ensemble les sommets $\{v_1, v_4, v_5, \ldots, v_{10}\}$. Les deux matrices des distances, Euclidienne et de Mahalanobis *de* et *dm* sont les suivantes :

$$de = \begin{pmatrix} 0,00 & \mathbf{3,90} & \mathbf{4,00} & \ldots \\ \mathbf{3,90} & 0,00 & 5,59 & \ldots \\ \mathbf{4,00} & 5,59 & 0,00 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}$$

$$dm = \begin{pmatrix} 0,00 & \mathbf{7,14} & \mathbf{6,93} & \ldots \\ \mathbf{7,14} & 0,00 & 6,44 & \ldots \\ \mathbf{6,93} & 6,44 & 0,00 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}$$

Nous constatons que $de_{12} < de_{13}$ par contre $dm_{12} > dm_{13}$. En effet, le sommet v_3 est considéré plus proche du sommet v_1 que le sommet v_2 en considérant la distance de Mahalanobis. Cela est expliqué du fait que v_3 est aussi plus proche des autres sommets de la classe O. Nous constatons ainsi que considérer la distance de Mahalanobis modifie le choix des sommets par rapport à la distance Euclidienne.

L'application de l'algorithme *k-médianes* avec la distance de Mahalanobis conduit le processus de classification à choisir l'insertion de v_3 puisqu'il est plus proche que v_2 des sommets de la classe O.

3.5.2 Choix initial des centres

Comme pour la distance euclidienne, nous avons choisi les centres initiaux parmi les sommets formant l'enveloppe convexe du graphe G.

3.5.3 La distance, index de proximité

Lors de la mise à jour d'un centre q_r d'une classe o_r, la distance est remplacée par un index de proximité. Les sommets de plus petits index de proximité λ_i^r seront choisis pour être insérés dans cette classe. λ_i^r est défini comme suit :

$$\lambda_i^r = \frac{dm_{ir}}{g_i}$$

3.5.4 Centres des classes

Pour calculer les coordonnées du centre d'une classe, nous considérons, en plus de la distance, les gains associés aux sommets de cette classe. Un centre q_r d'une classe o_r est défini par ses coordonnées xq_r et yq_r comme suit :

$$xq_r = \frac{\sum_{v_i \in o_r} g_i x_i}{\sum_{v_i \in o_r} g_i}, yq_r = \frac{\sum_{v_i \in o_r} g_i y_i}{\sum_{v_i \in o_r} g_i}$$

Après le calcul des coordonnées de chaque centre de classe, il faut mettre à jour les distances de Mahalanobis des autres sommets du graphe par rapport à ces centres en utilisant la formule déjà mentionnée pour déterminer dm_{ij}.

3.6 Algorithme *k-médianes* utilisant une métrique basée sur le profilage

Nous avons adapté une nouvelle métrique non conventionnelle, dans l'algorithme *k-médianes*, basée sur un concept de classification multi-critère. Cette approche est utilisée par DeSmet et Montano-Guzmán (2004) pour résoudre un problème de classification du risque d'un ensemble de pays et un autre sur le diagnostic du risque dans des sociétés.

Dans notre cas, la classification doit tenir compte de deux critères qui sont la compacité selon la distance et la maximisation du gain récolté dans chaque classe.

Pour cela nous avons adapté des notions utilisées par DeSmet et Montano-Guzmán (2004), pour nos objectifs de classification, afin de les utiliser au sein de l'algorithme de *k-médianes*.

3.6.1 La structure de préférence

Pour les problèmes multi-critères, particulièrement bi-critères, nous devons définir une structure de préférences entre les composants du problème à résoudre. Pour ce faire, considérons les relation suivantes :

- *Préférence P*
- *Indifférence I*
- *Incomparabilité J*

tels que pour deux sommets $v_i, v_j \in V$ nous avons :

- $v_i P v_j$ si v_i est préféré à v_j
- $v_i I v_j$ si v_i est indifférent à v_j
- $v_i J v_j$ si v_i est incomparable à v_j

Ces relations traduisent les situations de préférence, d'indifférence et d'incomparabilité. Elles satisfont les conditions suivantes :
$\forall v_i, v_j \in V$

- $v_i P v_j \Rightarrow v_j \neg P v_i$: *P est asymétrique*
- $v_i I v_i$: *I est réflexive*
- $v_i I v_j \Rightarrow v_j I v_i$: *I est symétrique*
- $v_i \neg J v_i$: *J est irréflexive*
- $v_i J v_j \Rightarrow v_j J v_i$: *J est symétrique*

Définition 1 : Les trois relations $\{P, I, J\}$ effectuent une structure de préférence sur V si elles satisfont les conditions précédentes et si pour deux sommets $v_i, v_j \in V$, une et une seule de ces propriétés soit vraie : $v_i P v_j$,$v_j P v_i$,$v_i I v_j$,$v_i J v_j$.

3.6.2 Le profil

Le problème de classification est basé sur l'idée que tous les sommets d'une même classe sont similaires dans le sens où ils sont préférables, indifférents et incomparables aux mêmes sommets. Pour déterminer cette similarité potentielle, nous introduisons la notion de profil à chaque sommet.

Définition 2 : Chaque sommet $v_i \in V$ est caractérisé par un profil $P(v_i)$.
$P(v_i)$ est défini par un 4-uple $\langle J(v_i), P^-(v_i), I(v_i), P^+(v_i)\rangle$ avec :

- $P_1(v_i) = J(v_i) = \{v_j \in V / v_i J v_j\}$
- $P_2(v_i) = P^-(v_i) = \{v_j \in V / v_j P v_i\}$

– $P_3(v_i) = I(v_i) = \{v_j \in V / v_i I v_j\}$
– $P_4(v_i) = P^+(v_i) = \{v_j \in V / v_i P v_j\}$

Supposons que nous avons trois points v_i, v_j et v_k. La figure 3.3 montre leurs dispositions par rapport à deux objectifs que l'on souhaite minimiser.

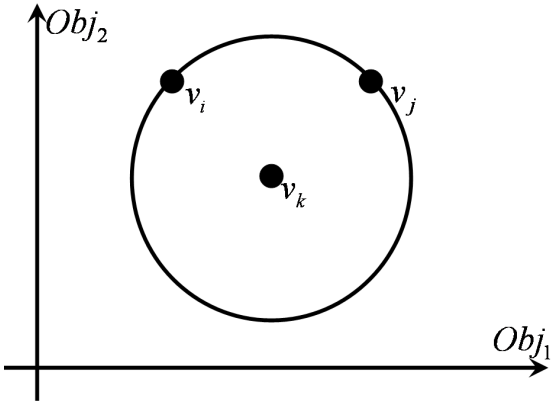

FIGURE 3.3 – Illustration des profils des sommets

Nous remarquons que :

– le point v_i est comparable avec v_j et non comparable avec v_k. En effet, v_i et v_j ont la même valeur de l'objectif Obj_2 et des valeurs différentes de Obj_1 ; donc, nous pouvons dire que v_i est meilleur que v_j (comparable). Tandis que v_i est meilleur que v_k par rapport à Obj_1 et que v_k est meilleur que v_i par rapport à Obj_2, donc nous ne pouvons pas conclure sur la performance de chaque point par rapport à l'autre (non comparable).
– le point v_j est comparable avec v_i et avec v_k. En effet, v_i et v_j ont la même valeur de l'objectif Obj_2 et des valeurs différentes de $Obj1$. Donc, nous pouvons dire que v_j est moins bon que v_i (comparable). Pour v_j et v_k, les valeurs des objectifs Obj_1 et Obj_2 de v_k sont moins bonnes que les valeurs des objectifs Obj_1 et Obj_2 de v_k ; donc, nous pouvons dire que v_j est moins bon que v_k (comparable).
– le point v_k est comparable avec v_j et non comparable avec v_i. En effet, les valeurs des objectifs Obj_1 et Obj_2 de v_k sont meilleures que les valeurs des objectifs Obj_1 et Obj_2 de v_j ; donc, nous pouvons dire que v_k est meilleur que v_j (comparable). Tandis que v_k est meilleur que v_i par rapport à Obj_2 et inversement par rapport à Obj_1, donc nous ne pouvons pas conclure quant à la performance de chaque point par rapport à l'autre (non comparable).

Si nous essayons de déterminer le profil de chaque point, selon le 4-uple $\langle J, P^-, I, P^+ \rangle$, nous obtenons :

– $P(v_i) = \langle \{v_k\}, \{v_j\}, \{v_i\}, - \rangle$
– $P(v_j) = \langle -, -, \{v_j\}, \{v_i, v_k\} \rangle$

– $P(v_k) = \langle \{v_i\}, \{v_j\}, \{v_k\}, - \rangle$

Profils des sommets pour la méthode à deux phases : Après avoir défini le profil d'un point en cas général, nous définissons ici le profil d'un sommet dans le contexte de la phase de classification pour la résolution du PmTS.
Soient deux sommets v_i et v_j et les deux dépôts v_1 et v_n de la figure 3.4.

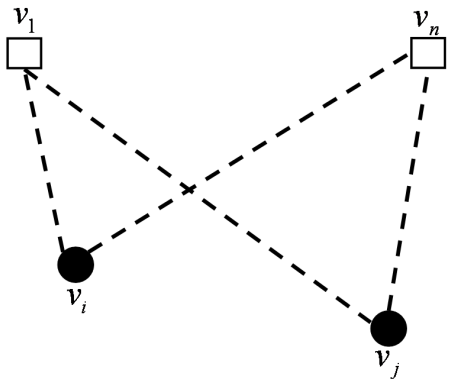

FIGURE 3.4 – Profils des sommets dans le cas du PmTS

Afin de définir le profit d'un sommet v_i, nous définissons les relations suivantes :

① $J(v_i) = \{v_j \in V / v_i J v_j\} = \{v_j \in V / (v_i \neq v_j) \wedge ((d_{0i} + d_{ij} + d_{jn} > L) \vee (d_{0j} + d_{ji} + d_{in} > L))\}$

② $I(v_i) = \{v_j \in V / v_i I v_j\} = \{v_j \in V / (v_i = v_j) \vee (\frac{d_{0i}+d_{in}}{g_i} = \frac{d_{0i}+d_{ij}+d_{jn}}{g_i+g_j} = \frac{d_{0j}+d_{ji}+d_{in}}{g_j+g_i})\}$

③ $P^+(v_i) = \{v_j \in V / v_i P^+ v_j\}$ $\{v_j \in V / (v_i \neq v_j) \wedge ((\frac{d_{0i}+d_{in}}{g_i} < \frac{d_{0i}+d_{ij}+d_{jn}}{g_i+g_j}) \vee (\frac{d_{0i}+d_{in}}{g_i} < \frac{d_{0j}+d_{ji}+d_{in}}{g_j+g_i}))$

④ $P^-(v_i) = \{v_j \in V / v_j P^- v_i\} = \{v_j \in V / (v_i \neq v_j) \wedge ((\frac{d_{0i}+d_{in}}{g_i} > \frac{d_{0i}+d_{ij}+d_{jn}}{g_i+g_j}) \vee (\frac{d_{0i}+d_{in}}{g_i} > \frac{d_{0j}+d_{ji}+d_{in}}{g_j+g_i}))$

Les significations des relations précédentes sont les suivantes :

① Deux sommets v_i est v_j sont incomparable *si est seulement si* ils sont différents et ils ne peuvent jamais figurer dans une même tournée. En effet, si la chaîne composée par les sommets v_i et v_j et d'extrémité v_1 et v_n dépasse la longueur limite L, v_i et v_j ne peuvent pas être présent simultanément dans une même tournée.

② Deux sommets v_i et v_j sont identique dans seulement deux cas : le premier est le cas où ils sont confondus $v_i = v_j$, le deuxième cas se présente lors de

l'insertion du sommet v_j dans la chaîne composée de v_i et d'extrémité v_1 et v_n. Si cette insertion n'influe pas le rapport distance sur gain, le sommets v_i est considéré identique à v_j et vise-versa. Il est à noté que le deuxième cas est rarement rencontré en pratique.

③ Un sommet v_j est non préférable à un sommet v_i dans le cas où l'insertion du sommet v_j dans la chaîne composée de v_i et d'extrémité v_1 et v_n engendre une perte sur le rapport distance sur gain.

④ Un sommet v_j est préférable à un sommet v_i dans le cas où l'insertion du sommet v_j dans la chaîne composée de v_i et d'extrémité v_1 et v_n engendre un gain sur le rapport distance sur gain.

3.6.3 La distance

Vu l'aspect bi-critère, la notion de distance utilisée dans les techniques de classification n'est pas appropriée. Cette observation nous a conduit à étendre la méthode de classification *k-médianes*.

Il faut définir un index de proximité entre les sommets en tenant compte de leur nature bi-critère. Pour ce faire, il suffit de faire intervenir les profils des sommets lors du calcul des distances entre les sommets.

Définition 3 : Soit $P(v_i)$ et $P(v_j)$ les profils des deux sommets v_i et v_j de V, l'index de proximité dp_{ij} entre les sommets v_i et v_j est défini comme suit :

$$dp_{ij} = 1 - \frac{\sum_{k=1}^{4} |P_k(v_i) \cap P_k(v_j)|}{n},$$

Cet index prend en compte, implicitement, une ressemblance entre chaque deux sommets v_i et v_j. Cette caractéristique est assurée par la prise en considération de l'intersection des profils des deux sommets en question.

3.6.4 Centres des classes

Nous définissons ici la méthode qui permet de déterminer l'élément central d'une classe. Cette phase est équivalente au calcul du centre de gravité dans le contexte d'une distance Euclidienne.

Dans notre cas, la détermination de l'élément central est basée sur une procédure de vote introduite par De Smet et Guzmán (DeSmet et Montano-Guzmán, 2004). L'idée est que le profil de la r^{ieme} classe est déterminé de façon à ce qu'il soit le plus semblable que possible de celui des sommets de la r^{ieme} classe.

Définition 4 : Soit $\{v_{r1}, v_{r2}, ..., v_{rp}\}$ les p sommets de la r^{ieme} classe. Le profil du r^{ieme} centre q_r, noté $P(q_r) = \{P_1(q_r), P_2(q_r), P_3(q_r), P_4(q_r)\}$, peut être définit par la procédure de vote définie par DeSmet et Montano-Guzmán (2004) :

$$v_j \in P_k(q_r) \Leftrightarrow k = argmax \sum_{l=1}^{p} 1_{\{v_j \in P_k(v_{rl})\}}$$

avec
$1_{\{\Delta\}} = 1$ **si** la condition Δ est vraie, 0 **sinon**
S'il y a différentes valeurs de k qui satisfassent la condition précédente, la valeur finale est choisie aléatoirement.

3.7 Résultats numériques

Afin de comparer la méthode de classification avec l'algorithme *k-médianes*, présenté par l'Algorithme 8, avec les trois métriques différentes, nous avons testé la méthode à deux phases sur l'ensemble de 353 instances tests décrites dans le chapitre précédent. Nous avons codé nos algorithmes en Java et nous avons effectué les tests sur une machine Intel Pentium4, 2.4Ghz et de 256Mo de mémoire.

Le tableau 3.1 présente un résumé du tableau 5.6 de l'Annexe 2. Il présente les résultats numériques moyens obtenus, sur les 353 instances, par les différentes métriques. Les notations suivantes sont considérées dans le tableau 3.1 :

- Euc : les résultats obtenus avec la distance Euclidienne
- Mah : les résultats obtenus avec la distance de Mahalanobis
- Pro : les résultats obtenus en utilisant la notion de profil.
- n : la nombre des sommets y compris les dépôts,
- m : le nombre des tournées exigées par la solution,
- G_{tot} : le gain total moyen obtenu par les différentes solutions des instances correspondantes,
- T_{cpu} : le temps moyens en secondes mis pour résoudre les instances correspondantes.

En examinant les résultats rapportés dans le tableau 5.6 de l'annexe 2, nous constatons que les différentes méthodes de classification (Euc, Mah et Pro) fournissent des résultats comparables. Par contre, si nous comptons le nombre des meilleures solutions obtenues par chacune nous trouvons que, sur les 353 instances, Mah donne 213 fois la meilleure solution contre 162 et 163 pour Euc et Pro. Nous constatons encore que pour 121 instances Mah est la seule approche à fournir la meilleure solution alors que Euc et Mah obtiennent la meilleure solution pour uniquement 57 et 46 instances. Dans le tableau 3.1, nous constatons que Mah fournit la meilleure moyenne sur l'ensemble des instances.

		Euc		**Mah**		**Pro**	
n	m	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
100	4	737,18	9,7	723,29	10,7	726,76	13,8
	3	809,37	12,7	794,79	16,2	798,79	16,3
	2	877,75	15,1	879,3	21,0	875,75	24,2
102	4	482,11	5,6	505,11	4,9	475,21	6,1
	3	573,37	7,0	579,63	7,9	555,05	8,7
	2	611,95	7,7	633,15	9,0	620,55	10,4
66	4	660	4,4	688,54	4,5	677,71	5,0
	3	755,4	5,5	772,2	6,2	765,2	5,7
	2	871,4	7,5	884,2	7,4	878,4	7,7
64	4	630	6,2	642	7,3	622,8	6,7
	3	750	5,7	758,25	9,1	751,5	7,1
	2	793,64	7,8	793,64	11,0	794,18	8,9
33	4	313,5	1,4	309,5	1,0	311	1,4
	3	398,5	1,6	392	1,8	399	1,7
	2	481	2,5	472,5	2,5	476	2,6
32	4	90	1,1	95	0,8	90,67	0,9
	3	115,63	1,5	121,56	1,2	115	1,4
	2	143,53	2,0	141,47	2,0	141,47	2,4
21	4	90,45	0,7	90,91	0,4	90,45	0,5
	3	130,91	0,7	124,55	0,5	130,91	0,5
	2	179,55	1,0	182,73	0,6	180,45	0,7
Moy		**499,77**	**5,1**	**504,01**	**6,0**	**498,90**	**6,3**

TABLE 3.1 – *Résultats numériques moyens de la méthode à deux phases avec k-médianes*

Les temps d'exécution sont minimes par rapport à $CRVV$ (voir chapitre précédent) puisque nous appliquons, dans la méthode à deux phases, une seule fois les procédures de classification et de routage.

En comparant les résultats fournis par ces Euc, Pro et Mah par rapport à ceux obtenus par CGW, TMH et AHS (voir 5.5), nous pouvons tirer la constatations suivantes.

Euc a fourni 75 fois la même solution que CGW, 77 fois que TMH et 67 fois que AHS. Elle a fourni (respectivement) 58, 19 et 4 fois une meilleure solution que CGW, TMH et AHS.

Pro a fourni 80 fois la même solution que CGW, 83 fois que TMH et 75 fois que AHS. Elle a fourni (respectivement) 51, 22 et 4 fois une meilleure solution que

CGW, TMH et AHS.
Mah a fourni 107 fois la même solution que CGW, 105 fois que TMH et 97 fois que AHS. Elle a fourni (respectivement) 59, 33 et 11 fois une meilleure solution que CGW, TMH et AHS.

Au point de vue qualité des solutions moyennes, Euc, Pro et Mah fournissent (respectivement) des solutions à 92.16%, 92.41% et 96.52% par rapport aux solutions fournies par CGW. Elles fournissent (respectivement) des solutions à 91.41%, 91.67% et 95.8% par rapport aux solutions fournies par TMH. Elles fournissent (respectivement) des solutions à 90.73%, 91.67% et 95.8% par rapport aux solutions fournies par AHS.

Au point de vue temps d'exécution, Euc, Pro et Mah fournissent des solutions pour le PmTS en des temps nettement plus réduits que les temps mis par les autres méthodes. En conclusion, nous pouvons dire que Mah est la méthode la plus adéquate pour être utilisée dans la méthode à deux phase pour le PmTS.

3.8 Conclusion

Nous avons présenté dans ce chapitre, trois métriques différentes pouvant être utilisées dans l'algorithme d'agrégation autour des centres mobiles (*k-médianes*) qui est intégrée dans la méthode à deux phases pour la résolution du PmTS. Les résultats obtenus par l'algorithme *k-médianes* sont plus performants que les résultats obtenus par la méthode de classification à voisinage variable au point de vue qualité de solutions et du temps d'exécution. En comparant les différentes métriques définies, nous avons constaté que la distance de Mahalanobis est la mieux adaptée pour la classification au sein de la méthode à deux phases pour le PmTS. Afin d'améliorer la méthode à deux phases et de la rendre plus compétitive par rapport aux autres méthodes de résolution du PmTS existantes, nous avons choisi d'utiliser la classification avec *k-médianes* et la distance de Mahalanobis en améliorant la phase de routage. Ces travaux font l'objet du chapitre suivant.

Les résultats obtenus par ces approches ont été publiés dans Khemakhem *et al.* (2006b,c).

Chapitre 4

Techniques de routage et approches Tabou

4.1 Introduction

Nous sommes intéressés, dans les deux chapitres précédents, à la phase de classification de la méthode à deux phases pour la résolutions du PmTS. Les approches que nous avons proposé sont caractérisées par leurs temps d'exécution réduits mais elles présentent des faiblesses au point de vue qualité de solutions.

Dans ce chapitre, nous nous intéressons à la deuxième phase de routage. Nous proposons des approches basées sur la recherche tabou. Ces approches nécessitent toujours une solution initiale générée par la méthode à deux phases décrite dans le deuxième et le troisième chapitre.

La première approche est basée sur la recherche tabou avec une structure de voisinage étendu. La deuxième consiste, à réduire cette structure de voisinage, et à changer et raffiner certains paramètres et composants utilisés dans la première approche.

Nous présentons, après la description de chaque approche, les résultats numériques obtenus par nos méthodes afin de les comparer aux résultats obtenus par les méthodes de la littératures pour la résolution du PmTS.

4.2 La recherche tabou

La recherche tabou RT est une métaheuristique originalement développée par Glover (1986) et indépendamment par Hansen (1986), sous l'appellation de "*steepest ascent mildest descent*". Elle est basée sur des idées simples, mais elle est néanmoins très efficace. Cette méthode combine une procédure de recherche locale avec un certain nombre de règles et de mécanismes permettant à celle-ci de ne pas rester

bloqué dans un optimum local, tout en évitant de cycler. Elle a été appliquée avec succès pour résoudre de nombreux problèmes difficiles d'optimisation combinatoire (Pirlot et Teghem, 2003).

4.2.1 Principe de base

Dans une première phase, la méthode de recherche tabou peut être vue comme une généralisation des méthodes de recherche locale. En effet, soit un problème de fonction objectif $Minf(x)$ avec $x \in X$, en partant d'une solution quelconque x appartenant à l'ensemble de solutions X, nous se déplaçons vers une solution $s(x)$ située dans le voisinage $V(x)$ de x. Donc l'algorithme explore itérativement l'espace de solutions X.

Afin de choisir le meilleur voisin $s(x)$ dans $V(x)$, l'algorithme évalue la fonction objectif f en chaque point $s(x)$, et retient un voisin qui améliore la valeur de la fonction objectif f, ou au pire celui qui la dégrade le moins.

L'originalité de la méthode de recherche tabou, par rapport aux méthodes de recherche locale, qui s'arrêtent dès qu'il n'y a plus de voisin $s(x)$ permettant d'améliorer la valeur de la fonction objectif f, réside dans le fait que nous retenons une solution voisine, même si celle-ci est plus mauvaise que la solution courante. Ce critère autorisant les dégradations de la fonction objectif évite à l'algorithme d'être piégé dans un minimum local. Par contre, il induit un risque de cyclage. En effet, lorsque l'algorithme a quitté un minimum local par acceptation de la dégradation de la fonction objectif, il peut visiter une solution déjà considérée à l'itération suivante.

Pour régler ce problème, l'algorithme a besoin d'une mémoire pour conserver pendant un nombre d'itérations la trace des dernières meilleures solutions déjà visitées. Ces solutions sont déclarées tabou, d'où le nom de la méthode. Elles sont stockées dans une *liste* de longueur Tl donnée, appelée liste tabou. Une nouvelle solution n'est acceptée que si elle n'appartient pas à cette liste tabou. Ce critère d'acceptation d'une nouvelle solution évite le cyclage de l'algorithme, durant la visite d'un nombre de solutions au moins égal à la longueur de la liste tabou, et il tend à diriger l'exploration de la méthode vers des régions du domaine de solutions non encore visitées.

La liste tabou est généralement gérée comme une liste de type $FIFO$: Nous éliminons à chaque itération la solution tabou la plus ancienne, en la remplaçant par la nouvelle solution retenue. Le codage d'une telle liste est lourd, car il faudrait garder en mémoire tous les éléments qui définissent une solution. Pour pallier cette contrainte, nous remplaçons la liste tabou comportant des solutions interdites par une liste de transformations interdites appelées *attributs*, en interdisant la transformation inverse d'une transformation faite récemment.

4.2.2 Critère d'aspiration

Le remplacement de la liste tabou des solutions visitées par la liste des attributs $\{x, s(x)\}$ conduit non seulement à l'interdiction de revenir vers des solutions précédentes, nous évitons le cyclage court, mais aussi vers un ensemble de solutions dont plusieurs peuvent ne pas avoir été visitées jusqu'ici. Il est donc primordial de corriger ce défaut et de trouver un moyen de lever l'interdiction de l'acceptation d'une transformation élémentaire $\{x, s(x)\}$ déjà effectuée (donc appartenant à la liste tabou), sous un certain critère, appelé critère d'aspiration. Cette idée est développée dans (Glover, 1989, 1990; Glover et Laguna, 1993).

Le critère d'aspiration le plus simple et le plus couramment utilisé consiste à tester si la solution produite de statut tabou présente un coût inférieur à celui de la meilleure solution trouvée jusqu'a présent. Si cette situation se produit, le statut tabou de la solution est levé. Ce critère est évidemment très sévere, il ne devrait pas être vérifié très souvent. Il apporte donc peu de changements au comportement de la méthode. D'autres critères d'aspiration plus complexes peuvent être envisages. L'inconvénient de recourir trop souvent à l'aspiration est qu'elle peut détruire, dans une certaine mesure, la protection offerte par la liste tabou vis-a-vis du cyclage.

4.2.3 Intensification

L'intensification consiste à approfondir la recherche dans certaines régions du domaine, identifiées comme susceptibles de contenir un optimum global. Cette intensification est appliquée périodiquement, et pour une durée limitée. Pour mieux intensifier la recherche dans une zone bien localisée, plusieurs stratégies sont proposées dans la littérature.

La plus simple consiste à retourner à l'une des meilleures solutions trouvée jusqu'à présent, puis de reprendre la recherche à partir de cette solution, en réduisant la longueur de la liste tabou pour un nombre limité d'itérations. Dans ce cas, nous adaptons la procédure de recherche tabou, en élargissant le voisinage de la solution courante. Nous pouvons aussi remplacer simplement l'heuristique tabou par une autre méthode plus puissante, ou mieux adaptée, pour une recherche locale.

4.2.4 Diversification

La diversification permet à l'algorithme de bien explorer l'espace des solutions, et d'éviter que le processus de recherche soit trop local et laisse de grandes régions de l'espace des solutions totalement inexplorées. La plus simple des stratégies de diversification consiste à interrompre périodiquement le déroulement normal de la procédure tabou, et à la faire redémarrer à partir d'une autre solution, choisie *aléatoirement* ou *intelligemment*.

Une autre méthode consiste à biaiser la fonction d'évaluation f, en introduisant un terme qui pénalise les transformations effectuées fréquemment, afin de favoriser des transformations nouvelles. Ce type de stratégie de diversification peut être utilise de façon continue, sans interrompre la procédure de recherche tabou.

Il est à noter que que la diversification et l'intensification sont des concepts complémentaires, qui enrichissent la méthode de recherche tabou et la rendent plus robuste et plus efficace.

4.3 Approche Tabou à voisinage étendu

Dans cette section nous présentons une approche tabou pour la résolution du PmTS. L'algorithme proposé initie la recherche avec une solution initiale fournie par la méthode à deux phases décrite dans le chapitre précédent. Nous avons testé cet algorithme avec trois différentes solutions initiales produites par la méthode à deux phases avec la métrique Euclidienne, de Mahalanobis et de profilage. Nous commençons la présentation de cette approche par l'exposition de son schéma général, ensuite nous détaillons ses caractéristiques principales et enfin nous analysons les résultats numériques obtenus par l'approche tabou avec les différentes solutions initiales en les comparant aux meilleures solutions obtenues par (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005; Archetti *et al.*, 2007b).

4.3.1 Schéma général de l'approche tabou à voisinage étendu

L'algorithme 9 présente le schéma général de l'approche tabou pour la résolution du PmTS. L'algorithme commence par générer une solution initiale Π à l'aide de la méthode à deux phases décrite dans le troisième chapitre avec l'une des métriques déjà mentionnées. Ensuite il initialise les paramètres de l'algorithme tels que le nombre d'itérations maximal $iterMax$, la taille de la liste tabou Tl, les fréquences d'aspiration, d'intensification et de diversification fa, fi et fd et déclare Π comme la meilleure solution obtenue Π^*.

Après les étapes d'initialisation, l'algorithme répète une procédure tabou tant que le nombre d'itérations $iter$ n'atteint pas $iterMax$. La procédure tabou consiste à choisir le meilleur voisin Ω^* de la solution Π dans son voisinage. Si Ω^* est meilleure que la meilleure solution rencontrée Π^* alors Ω^* sera considérée comme la meilleure solution rencontrée et le compteur des itérations sera mis à zéro.

Des procédures d'aspiration, d'intensification et de diversification seront appliquées si (respectivement) le nombre d'itération enregistré $iter$ est divisible par fa, fi, fd.

Enfin, *iter* doit être incrémenté de 1 et Π est remplacer par sa meilleure solution voisine Ω^* même si cette dernière n'améliore pas la meilleure solution rencontrée Π^*.

Algorithme 9 Schéma général de l'approche tabou pour le PmTS

Étape 1 : Générer une solution initiale Π de gain total $G(\Pi)$ pour le PmTS avec la méthode à deux phases.
Étape 2 : Initialiser
– la meilleure solution rencontrée $\Pi^* = \Pi$,
– le nombre d'itération maximal $iterMax$,
– fréquence d'aspiration fa,
– fréquence de diversification fd,
– fréquence d'intensification fi,
– la taille Tl de la liste tabou *liste*.
Répéter les étapes de 3 à 9 jusqu'à $iter = iterMax$.
Étape 3 : Sélectionner le meilleur voisin Ω^* de Π dans le voisinage défini,
Étape 4 : Déclarer comme tabou les attributs du mouvement pris en considération lors du passage de Π à Ω^*,
Étape 5 : Si $G(\Omega^*) > G(\Pi^*)$ alors $\Pi^* = \Omega^*$, $iter = -1$,
Étape 6 : Si $iter\%fa == 0$ alors appliquer aspiration,
Étape 7 : Si $iter\%fi == 0$ alors appliquer intensification,
Étape 8 : Si $iter\%fi == 0$ alors appliquer diversification,
Étape 9 : $\Pi = \Omega^*$, $iter++$

4.3.2 Composants de l'approche tabou à voisinage étendu

4.3.2.1 Liste tabou

Nous avons développé la liste tabou *liste* comme étant une file (stratégie FIFO) de taille Tl. Lors de la déclaration tabou d'un élément, il suffit d'insérer ses attributs dans la liste. Après Tl itérations, sans intervention de la procédure d'aspiration, les attributs de cet élément seront exclus de la liste tabou et par la suite il est considéré non tabou. Les attributs mémorisés dans la liste tabou représentants un couple (v, π) indiquant que le sommet v ne peut pas être inséré dans la tournée π pendant Tl itérations sauf exception produite par la procédure d'aspiration.

4.3.2.2 Structure du voisinage

Le passage d'une solution Π à une solution voisine Π_v suit une structure de voisinage bien précise. Nous avons défini cette structure comme suit :

Un voisin $\Omega = \{\pi'_1, \ldots, \pi'_k, \ldots, \pi'_m\}$ d'une solution $\Pi = \{\pi_1, \ldots, \pi_k, \ldots, \pi_m\}$ est déterminé en libérant un sommet v d'une tournée π_k. Après la libération une tentative d'amélioration avec permutation sera appliquée dans le but d'augmenter les gain des tournées. L'amélioration consiste à effectuer des échange des sommets libres (non visités) avec les sommets des tournée de la solution Π. Cette permutation est appliquée sur chaque sommet de chaque tournée avec tous les sommets libres en commençant par la tournée de plus faible gain.

Si la permutation aboutit à une augmentation du gain de la tournée, lors d'un échange d'un sommet libre avec un sommet de cette tournée, elle sera validée et la procédure passe à un autre sommet libre ; sinon elle est annulée et la procédure passe à un autre sommet de la tournée courante ou d'une autre tournée si nécessaire.

La procédure de permutation est suivi par une autre procédure d'amélioration avec insertion. Elle consiste à essayer d'insérer l'un des sommets restants dans l'une des tournées de la solution obtenue après la permutation.

Il est à noter que la validation d'une permutation ou d'une insertion n'est effectuée que dans le cas où ces dernières ne violent la contrainte de longueur et/ou le gain de la tournée actuelle augmente (ce qui est toujours le cas lors d'une insertion).

La libération d'un sommet ainsi que les procédures d'amélioration seront appliquées sur chaque sommet de la solution Π. La solution voisine Ω obtenue sera sauvegardée dans Ω^*, s'il s'agit du meilleur voisin courant de la solution Π. Le couple d'attributs (v, π_k) sera sauvegardé momentanément dans le couple d'attributs $v_t, \pi_t)$,$(v_t = v, \pi_t = \pi_k)$. Enfin, et lors de la sélection du meilleur voisin Ω^*, son couple (v_t, π_t) est inséré dans la liste tabou pour interdire le sommet v_t d'être inséré dans la tournée π_t pendant Tl itérations.

L'algorithme 10 décrit en détail la structure de voisinage que nous avons utilisé dans cette approche tabou.

4.3.2.3 Procédure d'aspiration

La procédure d'aspiration consiste, tout simplement, à appliquer les mouvements tabou sur la solution actuelle dans l'espoir d'améliorer la meilleur solution rencontrée. En effet, à chaque fa itérations sans amélioration la procédure tabou fait appel à l'aspiration, cette dernière consiste à prendre chaque couple d'attributs tabou (v_t, π_t) et effectue des tentatives de permutation du sommet v_t avec les sommets de la tournée π_t de la solution actuelle Π, ou bien des tentatives d'insertion si cela ne viole pas la contrainte de longueur maximale. La procédure d'aspiration est décrite par l'algorithme 11.

Algorithme 10 Schéma de recherche locale

Pour chaque sommet v de chaque tournée π_k de la solution courante $\Pi = \{\pi_1, \ldots, \pi_k, \ldots, \pi_m\}$*, faire les étapes de 1 à 14*,
Étape 1 : Enregistrer la solution courante Π dans la solution Ω, $\Omega = \Pi$,
Étape 2 : Libérer le sommet v de la tournée π_k et mettre à jour la liste des sommets non visités,
Étape 3 : Trier les tournées de Ω dans l'ordre croissant de leur gain total,
Tentatives de permutation
Étape 4 : Sélectionner à tour de rôle un sommets v_r de la liste des sommets non visités par les tournées de la solution Ω.
Étape 5 : Sélectionner à tour de rôle une tournée π_c de Ω en respectant l'ordre croissant des gains,
Étape 6 : Si (v_r, π_c) est non tabou, sélectionner à tour de rôle un des sommets v_c de la tournée π_c. Sinon aller à l'étape 5,
Étape 7 : Si $v_r \neq v$ ou $\pi_c \neq \pi_k$
– Permuter v_r et v_c,
– Si la contrainte de longueur n'est pas violée et le gain de la tournée π_c augmente, garder la permutation et aller à l'étape 4,
– Sinon, annuler la permutation et aller à l'étape 5
Étape 8 : Mettre à jour la liste des sommets non visités,
Étape 9 : Trier les tournées de Ω dans l'ordre croissant de leur longueur,
Tentatives d'insertion
Étape 10 : Sélectionner à tour de rôle un sommet v_r de la liste des sommets non visités par les tournées de la solution Ω,
Étape 11 : Sélectionner à tour de rôle une tournée π_c de Ω en respectant l'ordre croissant de longueurs,
Étape 12 : Si (v_r, π_c) est non tabou, sélectionner à tour de rôle un des sommets v_c de la tournée π_c ; Sinon aller à l'étape 11,
Étape 13 : Si $v_r \neq v$ ou $\pi_c \neq \pi_k$)
– Insérer v_r après v_c,
– Si la contrainte de longueur n'est pas violée, garder l'insertion et aller à l'étape 10,
– Sinon, annuler l'insertion et aller à l'étape 12,
Étape 14 : Si $G(\Omega)$ est le plus grand gain rencontré dans le voisinage alors $\Omega^* = \Omega$, $v_t = v$ et $\pi_t = \pi_k$,
Étape 15 : Mettre le couple (v_t, π_t) dans la liste tabou, $\Pi = \Omega^*$.

Algorithme 11 Schéma de la procédure d'aspiration de l'approche tabou à voisinage étendu et réduit

Étape 1 : Pour chaque couple d'attributs (v_t, π_t) de *liste*, faire les étapes 2 et 3,
Insertion
Étape 2 : Pour chaque sommet v de la tournée π_t de la solution Π,
- Insérer v_t après v,
- Si la longueur de la tournée π_t est inférieure à la longueur maximale, garder l'insertion, éliminer le couple (v_t, π_t) de liste, aller à l'étape 1,
- Sinon annuler l'insertion,

Permutation
Étape 3 : Pour chaque sommet v de la tournée π_t de la solution Π,
- Permuter v avec v_t,
- Si le gain de la tournée π_t augmente, garder la permutation, éliminer le couple (v_t, π_t) de *liste* et insérer le couple (v, π_t), aller à l'étape 3,
- Sinon annuler la permutation,

Étape 4 : Si $G(\Pi) > G(\Pi^*)$ alors $\Pi^* = \Pi$ sinon remettre *liste* et Π sur leurs états initiaux.

4.3.2.4 Procédure d'intensification

La procédure d'intensification est appelée par la procédure tabou juste avant l'appel à la procédure de diversification. Cette procédure est appliquée sur la solution courante afin d'intensifier la recherche dans l'espoir de trouver une solution plus prometteuse que la meilleure solution trouvée. Le principe de cette procédure ressemble à la procédure de recherche dans le voisinage d'une solution décrite par l'algorithme 10 mis à part quelques différences. La première différence consiste à ce que la procédure d'intensification n'accepte un voisin que s'il améliore la meilleure solution connue. La deuxième différence consiste au fait qu'à chaque itération, et au lieu de libérer un seul sommet v d'une tournée π_k, elle libère le sommet v et son successeur. La libération permet, dans un certain nombre de cas, et après les phase de permutation et d'insertion, de faire des échanges plus bénéfiques entre les tournées d'une solution en explorant plus profondément le voisinage.

4.3.2.5 Procédure de diversification

Après fd itérations sans amélioration, la procédure de diversification est appelée pour s'échapper de la zone de l'espace de recherche actuelle. Ce principe permet à la procédure tabou de ne pas demeurer longtemps dans une zone non prometteuse ou dans la zone d'un optimum local. Cette procédure consiste à faire des échanges de sommets entre les tournées en conservant la contrainte de longueur maximale et en ignorant la qualité de la solution obtenue après chaque permutation. La nouvelle

solution obtenue avec la diversification possède le même profit total que la solution précédente, uniquement les tournées qui changent. Il est à noter que la liste tabou *liste* sera totalement purgée lors de la diversification. L'algorithme 12 décrit plus précisément la démarche de cette procédure.

Algorithme 12 Schéma de la procédure de diversification de l'approche tabou à voisinage étendu

Etape 1 : Pour i allant de 1 à $(m-1)$, prendre la tournée π_i de la solution courante Π et faire les étapes de 2 à 4,
Etape 2 : Pour j allant de $(i+1)$ à m, prendre la tournée π_j de la solution courante Π et faire les étapes 3 et 4,
Etape 3 : Pour chaque sommet v_k de la tournée π_i faire l'étape 4,
Etape 4 : Pour chaque sommet v_h de la tournée π_j permuter les sommets v_k et v_h
- Si les tournées π_i et π_j ne violent pas la contrainte de longueur maximale, valider la permutation et aller à l'étape 4,
- Sinon annuler la permutation.

4.3.3 Paramétrages de l'approche tabou à voisinage étendu

Nous précisons, dans cette section, les valeurs des paramètres que nous avons choisi pour notre approche.

4.3.3.1 Taille de la liste tabou

La taille d'une instance du PmTS dépend de plusieurs facteurs jouant un rôle important lors du processus de sa résolution. Nous avons mentionné dans le premier chapitre que l'augmentation des paramètres n, m et L complique d'avantage la résolution du PmTS. Pour cela, la taille Tl de la liste tabou devrait être en fonction de ces paramètres $Tl = f(\overline{n}, m, L)$, où $\overline{n} = n - \underline{n}$, $\underline{n}$ est le nombre des sommets qui ne peuvent pas figurer dans la solution (i.e. $\underline{n} = |\underline{V}|$ où $\underline{V} = \{\forall v_i \in V, d_{1i} + d_{in)} > L\}$) (voir section 1.2.2 du chapitre 1).

Après plusieurs tests numériques et vue que la taille du problème peut être réduit, en s'appuyant sur la deuxième propriété du PmTS définie dans la premier chapitre, nous avons ignoré le paramètre L. Par la suite, nous avons fixé Tl au rapport du nombre de sommets sur le nombre de tournées : $Tl = \frac{\overline{n}}{m}$.

4.3.3.2 Fréquence d'aspiration

La procédure d'aspiration est appelée à chaque fois où tous les éléments de la liste tabou sont renouvelés sans avoir une amélioration de la meilleure solution rencontrée, c'est à dire $fa = Tl$.

4.3.3.3 Fréquence d'intensification et de diversification

A chaque fois, et avant d'appliquer la procédure de diversification, la procédure d'intensification est appelée dans l'espoir de trouver une solution prometteuse dans la zone de recherche actuelle. Après plusieurs tests numériques, nous avons fixé les fréquences d'intensification et de diversification au nombre de sommets divisé par deux : $fi = fd = \frac{\overline{n}}{2}$.

4.3.3.4 Nombre d'itérations maximal

Après $iterMax$ itérations, sans amélioration, la procédure de recherche tabou s'arrête et fourni la meilleure solution trouvée. Nous avons fixé initialement $iterMax$ au nombre de tournées multiplié par la taille du problème ($iterMax = \overline{n} \times m$), mais nous avons constaté après les tests que ce choix pénalise les instances où le nombre de tournées est faible. Par la suite nous avons fixé $iterMax$ à quatre fois la taille du problème ($iterMax = 4\overline{n}$).

4.3.4 Résultats numériques

4.3.4.1 Contexte et environnement

Nous avons codé notre approche tabou, pour la résolution du PmTS, en JAVA et nous l'avons testé avec des différentes solutions initiales générées par la méthode à deux phases avec les différentes métriques décrites dans le troisième chapitre. Nous avons exécuté les différentes méthodes sur une machine Intel Pentium 4, 2.4 Ghz et de 256 Mo Ram. Nous avons utilisé les 353 instances benchmark du PmTS, publiées par Chao *et al.* (1996b).

Le tableau 4.1 présente un résumé des résultats numériques obtenus par notre méthode (voir tableau 5.7 de l'Annexe 3) comparés avec ceux des méthodes proposées par (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005; Archetti *et al.*, 2007b).

Il présente, pour chaque n fixé, les gains totaux obtenus par l'approche pour des différentes valeurs de m et L. Les notations suivantes sont considérées dans le tableau 4.1 :

- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigé par la solution
- **CGW** : les résultats des solutions obtenues avec l'approche de Chao *et al.* (1996b),
- **TMH** : les résultats des solutions obtenues avec l'approche de Tang et Miller-Hooks (2005),
- **AHS** : Les résultats de valeur maximale après 3 exécutions de la méthode à voisinage variables rapide, considérée comme le meilleur compromis, de (Archetti *et al.*, 2007b),

		CGW	**TMH**	**AHS**	**TEuc**	**TPro**	**TMah**
n	m	G_{tot}	G_{tot}	G_{tot}	G_{tot}	G_{tot}	G_{tot}
100	4	13024	13339	13629	13386	13379	13503
	3	15486	16041	16207	16007	15947	15956
	2	17514	17901	18279	17921	18007	17925
102	4	9450	9778	9822	9749	9768	9799
	3	11124	11258	11344	11288	11180	11294
	2	12678	12669	12812	12642	12620	12696
66	4	16705	16775	17010	16780	16975	16910
	3	19415	19395	19590	19530	19500	19505
	2	22265	22170	22395	22280	22285	22370
64	4	3582	3564	3570	3570	3570	3570
	3	6300	6264	6342	6330	6324	6336
	2	8964	9000	9012	8934	8934	8898
33	4	6650	6700	6730	6730	6700	6670
	3	8060	8160	8230	8160	8160	8150
	2	9770	9840	9920	9840	9830	9860
32	4	1490	1515	1515	1515	1515	1510
	3	2010	1995	2000	2000	1990	2005
	2	2525	2530	2535	2525	2520	2495
21	4	1040	1040	1040	1040	1040	1040
	3	1495	1495	1500	1500	1500	1500
	2	2090	2090	2095	2085	2095	2095
Total		191637	193519	195577	193812	193839	194087

TABLE 4.1 – *Les sommes des gains des solutions obtenus par l'approche Tabou avec voisinage complet organisés par classes d'instances*

- **TEuc** : les résultats des solutions obtenues avec l'approche tabou avec solution initiale à distance Euclidienne
- **TPro** : les résultats des solutions obtenues avec l'approche tabou avec solution initiale à distance de profilage
- **TMah** : les résultats des solutions obtenues avec l'approche tabou avec solution initiale à distance de Mahalanobis
- G_{tot} : le gain total obtenu par la solution des instances correspondantes avec des différentes longueurs maximales
- $Total$: Les gains et les temps totaux obtenus par toutes les méthodes avec toutes les instances

	n	T_{moy}	T_{max}	n	T_{moy}	T_{max}
CGW	100	*	935	102	*	841
TMH		184,5	797		84,4	433
AHS		22,5	121		10,3	90
$Teuc$		266,5	786		123,3	522
$Tpro$		270,6	609		159	863
$Tmah$		250,5	700		101	543
CGW	66	*	194	64	*	150
TMH		16,8	71		14,1	46
AHS		34,2	30		8,7	20
$Teuc$		24,6	89		24,8	42
$Tpro$		23,5	61		25,1	42
$Tmah$		21,7	56		29,3	64
CGW	33	*	15	32	*	15
TMH		2,5	7		1,4	3
AHS		0,2	1		0,1	1
$Teuc$		2,7	5		2,1	5
$Tpro$		2,7	5		2,2	5
$Tmah$		2,6	6		1,9	6
CGW	21	*	1			
TMH		*	*			
AHS		0	0			
$Teuc$		0,8	2			
$Tpro$		0,9	1			
$Tmah$		0,6	1			

TABLE 4.2 – *Temps CPU obtenus par les approches $TEuc$, $TPro$ et $TMah$ par rapport à ceux obtenu par CGW, TMH et AHS*

Le tableau 4.2 présente les temps CPU moyens et maximums des différentes versions de l'approche tabou à voisinage étendu, ceux obtenus par la méthode considérée comme meilleur compromis proposée par Archetti *et al.* (2007b), ceux obtenus par Chao *et al.* (1996b) et Tang et Miller-Hooks (2005). Les notations suivantes sont considérées dans le tableau 4.2 :

- T_{moy} : Le temps moyen de résolution des instances de taille n correspondantes.
- T_{max} : Le temps maximal de résolution d'une instance de taille n.

4.3.4.2 Analyse des résultats

En consultant le tableau 4.1, nous constatons que presque pour toutes les instances $TEuc$, $TPro$ ou $TMah$ récolte un gain total plus important que ceux obtenus par Chao *et al.* (1996b) et Tang et Miller-Hooks (2005). Par contre, nous constatons, à l'exception de quelques instances, que notre approche fournit généralement un gain total moins important que les meilleures résultats obtenus par Archetti *et al.* (2007b).

En consultant la ligne $Total$ du tableau 4.1 et en comparant les différentes métriques que nous avons utilisé pour notre approche, nous constatons que $TMah$ fournit plus de gain par rapport à $TEuc$ et $TPro$. Ces deux dernières sont similaires.

En consultant le tableau 5.7 de l'Annexe 3, nous remarquons que $TEuc$, $TPro$ et $TMah$ trouvent (respectivement) 238, 232 et 240 fois les meilleures résultats connus ; et qu'elles fournissent 11,6 et 10 meilleure(s) solution(s) non déjà connue(s) indiquées par (*).

Concernant le temps d'exécution, les trois méthodes sont similaires. Chacune necessite environ 412 minutes pour résoudre la totalité des instances. Au point de vue temps CPU (voir tableau 4.2), nous constatons que notre approche consomme mois de temps CPU par rapport au meilleur compromis proposé par Archetti *et al.* (2007b) malgré que notre machine est moins performance que celle utilisée par eux. Par contre, elle consomme environ le double du temps consommé par la méthode de Tang et Miller-Hooks (2005) ce qui est compensé par la qualité de solution fournie par notre approche. En se comparant par rapport à la méthode de Chao *et al.* (1996b), cette dernière fournit environ le double du temps consommé par la notre.

Nous avons intégrer la méthode tabou, présentée dans cette section, au sein de deux processus différents. Le premier est basé sur une stratégie d'oscillation et le deuxième est basé sur la notion de mémoire adaptative. Nous avons remarqué que ces deux processus consomme énormément de temps CPU. Pour cela, nous avons proposé une deuxième approche tabou avec une structure de voisinage plus réduite que nous la présenterons dans la section suivante.

4.4 Approche Tabou à voisinage réduit

Dans cette section nous présentons une deuxième approche tabou pour la résolution du PmTS. Cette approche est caractérisée par une structure de *voisinage réduit* par rapport à celle utilisée dans l'approche décrite dans la section précédente. Comme dans la dernière approche, cet algorithme initie la recherche avec une solution initiale fournie par la méthode à deux phases décrite dans le troisième chapitre. Nous commençons la présentation de cette approche par l'exposition de son schéma

général, ensuite nous détaillons ses principaux composants tabou ainsi que leurs paramétrages et enfin nous analysons les résultats numériques obtenus par cette approche. La solution initiale utilisée est celle avec la distance de Mahalanobis tout en comparant les résultats fournis par rapport aux résultats de la première approche et aux solutions obtenues par (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005; Archetti *et al.*, 2007b).

4.4.1 Schéma général de l'approche tabou à voisinage réduit

La procédure de recherche, de cette approche, suit le même algorithme générale décrit par l'algorithme 9. Il commence à générer une solution initiale Π à l'aide de la méthode à deux phases décrite dans le chapitre précédent avec la distance de Mahalanobis. Nous considérons uniquement la distance de Mahalanobis car elle a montrée sa supériorité par rapport aux autres métriques dans la première approche.

Ensuite, l'algorithme initialise les paramètres tels que le nombre d'itérations maximal $iterMax$, la taille de la liste tabou Tl, les fréquences d'aspiration, d'intensification et de diversification fa, fi et fd et déclare la solution initiale comme la meilleure solution obtenue.

Après les étapes d'initialisation, l'algorithme répète une procédure tabou tant que le nombre d'itérations $iter$ n'atteint pas $iterMax$. La procédure tabou consiste à choisir le meilleur voisin Ω^* de la solution Π dans son voisinage. Si Ω^* est meilleure que la meilleure solution rencontrée Π^* alors Ω^* est considérés comme la meilleure solution rencontré et le compteur des itérations $iter$ sera mis à zéro.

Des procédures d'aspiration, d'intensification et de diversification sont appliquées si (respectivement) le nombre d'itération enregistré $iter$ est divisible par fa, fi, fd.

Enfin, $iter$ doit être incrémenté de 1 et Π est remplacé par sa meilleure solution voisine Ω^* même si cette dernière n'améliore pas la meilleure solution rencontrée Π^*.

4.4.2 Composants de l'approche tabou à voisinage réduit

Nous détaillons d'avantage, dans cette section, les différents composants de la procédure tabou définis dans la section précédente. Nous commençons par la liste tabou *liste*, ensuite nous présentons la structure de voisinage réduit que nous avons utilisé puis les procédures d'aspiration, d'intensification et de diversification.

4.4.2.1 Liste tabou

Comme dans l'approche précédente, les attributs mémorisés dans la liste tabou *liste* sont représentés sous forme d'un couple (v, π) indiquant que le sommet v ne peut pas être inséré dans la tournée π pendant Tl itérations. Nous avons développé cette liste en utilisant une matrice de taille $(n \times m)$. Lors de la déclaration d'un

attribut (v, π) comme un élément tabou $liste(v, \pi)$ est mis à $(iter + Tl)$. Tant que $liste(v, \pi) > iter$ le mouvement défini par l'attribut (v, π) est considéré tabou. De cette façon la vérification de l'interdiction d'un mouvement est plus rapide que l'utilisation d'une liste FIFO utilisée dans l'adaptation de la recherche tabou à voisinage complet (voir la section 4.3.2.1). En effet, si nous utilisons une liste FIFO la vérification d'un attribut, s'il est déclaré tabou ou non, nécessite une recherche dans la liste tabou *liste*. Par contre, en utilisant la matrice de taille $(n \times m)$, la vérification se fait par un simple accès à la case $liste(v, \pi)$ correspondante.

4.4.2.2 Structure du voisinage réduit

Afin de réduire la structure de voisinage utilisée dans l'approche tabou précédente, nous avons choisi une structure plus restreinte. Au début de l'algorithme tabou, une procédure est appliquée une seule fois pour déterminer pour chaque sommet $v_i \in V - \{v_0, v_{n-1}\}$ l'ensemble $\widetilde{V}_i$ contenant ses *pp* plus proches sommets.

Pour une solution courante $\Pi = \{\pi_1, \ldots, \pi_k, \ldots, \pi_m\}$, un voisin Ω est obtenu en libérant un sommet v de sa tournée π_k puis en libérant un autre sommet v' de son ensemble $\widetilde{V}_v$ s'il appartient à une tournée π_h de la solution courante. Après la libération, une tentative d'amélioration avec permutation sera appliquée dans le but d'augmenter les gain des tournées. La permutation est effectuée avec les sommets non visités par Π y compris le sommet libéré.

Si la permutation aboutit à une augmentation du gain de la tournée, elle sera validé et la procédure passe à un autre sommet restant. Sinon, elle est annulée et la procédure passe à un autre sommet de la tournée courante ou d'une autre tournée si nécessaire.

La procédure de permutation est suivi par une autre procédure d'amélioration avec insertion. Elle consiste à essayer d'insérer l'un des sommets restants dans l'une des tournées de la solution récupérée après la permutation.

Il est à noter que la validation d'une permutation ou d'une insertion n'est effectuée que dans le cas où ces dernières ne violent la contrainte de longueur et si le gain de la tournée actuelle augmente, ce qui est naturellement le cas dans le cas d'insertion. La libération d'un sommet ainsi que les procédures d'amélioration sont appliquées sur chaque sommet v_i de chaque tournée de la solution Π avec tous les sommets de son ensemble $\widetilde{V}_i$.

A chaque fois le voisin Ω obtenu sera examiné, dans le cas où Ω fournit la meilleure solution du voisinage il sera sauvegardé dans Ω^* comme étant le meilleur voisin de la solution Π et ses deux couples d'attributs $(v_t = v, \pi_t = \pi_k)$ et $(v_{t'} = v', \pi_{t'} = \pi_h)$ comme étant les sommets v_t et $v_{t'}$ sont retirés (respectivement) des tournées π_k et π_h. Enfin, et lors de la sélection du meilleur voisin Ω^*, les couples (v_t, π_t) et $(v_{t'}, \pi_{t'})$ sont déclarés tabou pendant Tl itérations.

L'algorithme 13 décrit plus précisément la structure de voisinage réduite que nous avons utilisée pour cette approche.

.

4.4.2.3 Procédure d'aspiration

La procédure d'aspiration utilisée dans cette approche est la même utilisée dans l'approche à voisinage complet. En effet, à chaque fa itérations la procédure d'aspiration est appliquée et elle consiste à prendre chaque couple d'attributs tabou (v_t, π_t) et effectuer des tentatives de permutation du sommet v_t avec les sommets de la tournée π_t de la solution actuelle Π, ou bien des tentatives d'insertion si cela ne viole pas la contrainte de longueur maximale. La procédure d'aspiration est décrite par l'algorithme 11.

4.4.2.4 Procédure d'intensification

A chaque fi itérations sans amélioration, une procédure d'intensification est appliqué dans le but d'améliorer la solution courante. Tout simplement, cette procédure consiste à appliquer sur chaque tournée de la solution courante l'algorithme Genius de (Gendreau *et al.*, 1992) pour la résolution du Problème de Voyageur de Commerce. L'algorithme Genius permet de corriger, dans la mesure du possible, chaque tournée afin de réduire sa longueur. La réduction des longueurs des tournée peut permettre l'insertion d'autre sommets qui ne figurent pas dans la solution d'où une augmentation du gain total. Il est à mentionner que, lors de l'application de Genius sur une tournée, l'adjacence des dépôts v_0 et v_{n-1} peut ne pas être respectée. Pour éviter cela il faut, avant de lancer Genius sur une tournée, annuler la distance entre les dépôts pour être sûr qu'ils soient adjacents dans la tournée fournie par Genius (i.e. $d_{0(n-1)} = 0$).

4.4.2.5 Procédure de diversification

Après fd itérations sans amélioration et lors de l'échec de la procédure d'intensification, la procédure de diversification est appelée pour s'échapper de la zone de recherche actuelle (voir l'algorithme 14). Cette procédure consiste à choisir les deux tournées π_1 et π_2 de gains minimums de la solution actuelle. Ensuite, il s'agit de couper chaque tournée à son milieu et de les reconnecter en appliquant le principe de la recherche locale *2*-opt de (Lin, 1965) (voir figure 4.1). Après cette transformation, si l'une des deux tournées modifiées π_k dépasse la longueur maximale L, une procédure de correction sera appliquée. Cette procédure consiste à éliminer le sommet v_i de la tournée π_k tant que la longueur de la tournée viole la longueur maximale L, avec $i = \underset{j}{argmax}\{\frac{d_{(j-1)j}+dj(j+1)-d(j-1)(j+1)}{g_j}, \forall v_j \in \pi_k, v_j \neq v_1, v_j \neq v_n\}$. Après chaque éli-

Algorithme 13 Schéma de recherche dans un voisinage réduite

Pour chaque sommet v de chaque tournée π_k de la solution courante Π *et pour chaque sommet v', de l'ensemble de ses pp plus proche voisins, de la tournée π_h si elle existe, faire les étapes de 1 à 13*,
Étape 1 : Enregistrer la solution courante Π dans la solution Ω, $\Omega = \Pi$,
Étape 2 : Retirer le sommet v de la tournée π_k, et le sommet v' de la tournée π_h,
Étape 3 : Mettre à jour la liste des sommets non visités par Ω,
Tentatives de permutation
Étape 4 : Sélectionner à tour de rôle un sommets v_r de la liste des sommets non visité par la solution courante,
Étape 5 : Sélectionner un sommet v_c apparaissant sur π_h et appartenant à $\widetilde{V}_r$
Étape 6 : Si $liste(v_r, \pi_h) > iter + Tl$ aller à l'étape 5,
Étape 7 : Si $v_r \neq v$ ou $\pi_h \neq \pi_k$
– Permuter v_r et v_c,
– Si la contrainte de longueur n'est pas violée et le gain de la tournée π_h augmente garder la permutation et aller à l'étape 4,
– Sinon annuler la permutation et aller à l'étape 5
Étape 8 : Mettre à jour la liste des sommets non visités,
Tentatives d'insertion
Étape 9 : Sélectionner à tour de rôle un sommets v_r de la liste des sommets restants,
Étape 10 : Sélectionner à tour de rôle une tournée π_h,
Étape 11 : Si $liste(v_r, \pi_h) < iter + Tl$, sélectionner à tour de rôle un des sommets v_c de la tournée π_h, sinon aller à l'étape 10,
Étape 12 : Si $v_r \neq v$ ou $\pi_h \neq \pi_k$
– Insérer v_r après v_c,
– Si la contrainte de longueur n'est pas violée garder l'insertion et aller à l'étape 9,
– Sinon annuler l'insertion et aller à l'étape 10,
Étape 13 : Si $G(\Omega)$ est le plus grand gain rencontré dans le voisinage alors $\Omega^* = \Omega$, $v_t = v$, $\pi_t = \pi_k$, $v_{t'} = v'$ et $\pi_{t'} = \pi_h$,
Étape 14 : Mettre à jour les éléments tabou $l(v_t, \pi_t) = iter + Tl$, $l(v_{t'}, \pi_{t'}) = iter + Tl$, $\Pi = \Omega^*$.

mination d'un sommet, l'algorithme Genius est appliqué sur la tournée dans l'espoir de réduire la longueur en corrigeant la tournée.

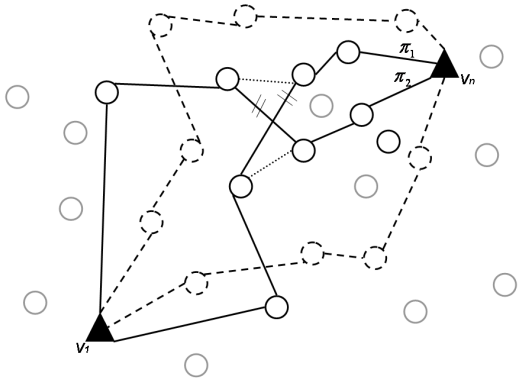

FIGURE 4.1 – Procedure de diversification avec 2-opt

Algorithme 14 Schéma de la procédure de diversification de l'approche tabou à voisinage réduit

Etape 1 : Choisir les deux tournées π_1 et π_2 de gains minimums,
Etape 2 : Appliquer *2*-opt sur le cycle composé par π_1 et π_2,
Etape 3 : Pour chaque $\pi_k, k = \{1, 2\}$ faire les étapes suivantes,
Etape 4 : Appliquer Genius sur π_k
Etape 5 : Tant que la longueur de π_k excède la longueur maximale L faire
– Éliminer le sommet v_i le plus coûteux,
– $i = \underset{j}{argmax}\{\frac{d_{(j-1)j}+dj(j+1)-d(j-1)(j+1)}{g_j}, \forall v_j \in \pi_k, v_j \neq v_1, v_j \neq v_n\}$
– Appliquer Genius sur π_k.

4.4.3 Paramétrage de l'approche tabou à voisinage réduit

Nous détaillons d'avantage, dans cette section, les paramètres des différents composants que nous avons choisi dans notre approche.

4.4.3.1 Initialisation

Afin de tenir compte de la taille du problème au point de vu du nombre de sommets et de tournées, nous avons fixé pour notre approche *pp* à $\frac{\overline{n}}{2m}$, i.e. $pp = \frac{\overline{n}}{2m}$ où $\overline{n} = n - \underline{n}$, $\underline{n}$ est le nombre des sommets qui ne peuvent pas figurer dans la solution (i.e. $\underline{n} = |\underline{V}|$ où $\underline{V} = \{\forall v_i \in V, d_{1i} + d_{in)} > L\}$) (voir section 1.2.2 du chapitre 1).

4.4.3.2 Taille de la liste tabou

Après plusieurs tests de l'algorithme de cette approche sur différentes instances, nous avons fixé la taille de la liste tabou à $Tl = \frac{\sqrt{\overline{n} \times m}}{2}$.

4.4.3.3 Fréquence d'aspiration

Dans cette approche nous avons choisis d'appliquer la procédure d'aspiration à chaque itération de l'algorithme, c'est à dire $fa = 1$.

4.4.3.4 Fréquence d'intensification

A chaque $\frac{\overline{n}}{2}$ itérations une procédure d'intensification est appelée dans l'espoir de trouver une solution prometteuse dans la zone de recherche actuelle de l'espace des solutions réalisables (i.e. $fi = \frac{\overline{n}}{2}$).

4.4.3.5 Fréquence de diversification

La procédure de diversification est appelée à chaque $\overline{n}$ itérations (i.e. $fd = \overline{n}$). Les procédures de diversification et d'intensification se coïncident une fois sur deux, dans ce cas la procédure d'intensification est appliquée en premier lieu, si cette dernière permet d'améliorer la meilleur solution connue la procédure de diversification ne sera pas effectuée.

4.4.3.6 Nombre d'itérations maximal

Après $iterMax$ itérations, sans amélioration, la procédure de recherche tabou s'arrête et fournit la meilleure solution trouvée. Nous avons fixé $iterMax$ à quatre fois la taille du problème sans tenir compte des sommets qui ne peuvent pas figurer dans la solution ($iterMax = 4\overline{n}$).

4.4.4 Résultats numériques

4.4.4.1 Contexte et environnement

Nous avons codé notre approche tabou avec voisinage réduit, pour la résolution du PmTS, en JAVA et nous l'avons testé sur une machine Intel Pentium 4 - 2.4 Ghz - 256 Mo Ram sur les 353 instances benchmark publiées par (Chao *et al.*, 1996b). Nous avons comparé les résultats obtenus par cette approche par rapport aux résultats obtenus par (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005; Archetti *et al.*, 2007b) et par rapport au résultats obtenus par notre approche avec le voisinage complet.

Le tableau 4.3 présente un résumé du tableau 5.8 de l'Annexe 4. Il présente pour chaque approches les résultats numériques totaux obtenus pour un ensemble de groupes d'instances. Les notations suivantes sont considérées dans le tableau 4.3 :

- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigées par la solution
- **CGW** : les résultats des solutions obtenues avec l'approche de (Chao *et al.*, 1996b)
- **TMH** : les résultats des solutions obtenues avec l'approche de (Tang et Miller-Hooks, 2005)
- **AHS** : les résultats de valeur maximale après 3 exécutions de la méthode à voisinage variables rapide, considérée comme le meilleur compromis, de (Archetti *et al.*, 2007b)
- **TE** : les résultats des meilleurs solutions obtenues avec l'approche tabou à voisinage complet
- **TR** : les résultats des meilleurs solutions obtenues avec l'approche tabou à voisinage réduit
- G_{tot} : le gain total obtenu par la solution des instances correspondantes avec des différentes longueurs maximales
- $Total$: Les gains totaux obtenus par toutes les méthodes avec toutes les instances

Le tableau 4.4 présente un résumé des temps CPU de l'approche tabou à voisinage réduit comparée par rapport à ceux obtenus par l'approche tabou à voisinage étendu et à ceux obtenus par (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005; Archetti *et al.*, 2007b). Les notations suivantes sont considérées dans le tableau 4.4 :

- T_{moy} : Le temps moyen de résolution des instances de taille n correspondantes.
- T_{max} : Le temps maximal de résolution d'une instance de taille n.
- $Ratio$: mesure la performance de la méthode en tenant compte de la machine sur laquelle est testée.

4.4.4.2 Analyse des résultats

En consultant le tableau des résultats moyens 4.3 nous remarquons que la somme des gain fournie par l'approche tabou à voisinage réduit TR est moins importante que la somme fournie par l'approche à voisinage complet TE. En effet, la réduction de la structure de voisinage empêche d'explorer d'avantage le champs de recherche. En revanche, TR présente une qualité au point de vue temps d'exécution du fait qu'elle consomme 40% de temps CPU de moins que TE pour une perte de 0.08% de gain total.

En comparant TR par rapport aux résultats obtenus le meilleur compromis proposé par AHS (Archetti *et al.*, 2007b), nous constatant que TR fournie un gain total moins important de 1% que celui obtenu par AHS. Nous constatons encore que TR

fournit un gain total plus important de 0.22% et de 1.20% (respectivement) à ceux obtenus par TMH (Tang et Miller-Hooks, 2005) et CGW (Chao *et al.*, 1996b).

En consultant le tableau 4.4, nous pouvons constater que pour la majorité des instances, TR requiert des temps CPU moyens et maximums les plus faibles par rapport à TE. Nous remarquons que la réduction de la structure de voisinage dans TR présente un apport par rapport à TE, du point de vue des temps d'exécution, que pour les instances de grandes et moyennes taille.

Vue la non-conformité des machines sur lesquelles les tests des autres méthodes (CGW, TMH AHS) ont été effectués, nous avons utilisé la relation, pour calculer le ratio entre les performances des machines, utilisée par Prins (2004) pour évaluer différents algorithmes de résolution du PTVC.

La machine utilisée pour tester TMH fonctionne à une performance estimée à 700 $MFlops$. Selon le logiciel d'évaluation de performance des machines *SiSoftware Sandra Standard*, notre machine fonctionne avec une performance de 3215 $Mflops$ et celle utilisée pour tester AHS fonctionne avec une performance de 3492 $Mflops$. Pour la machine utilisée pour tester CGW, nous n'avons pas pu savoir sa performance en $MFlops$.

Nous avons pris TMH comme méthode de référence et nous avons calculé le ratio suivant pour chaque méthode :

$$Ratio_a = \frac{T_a \times P_a}{T_b \times P_b}$$

Avec T_a et P_a sont le temps d'exécution et la performance de la machine où la méthode, que nous voulons évaluer, à été testée. T_b et P_b sont ceux de la méthode de référence TMH.

Mis à part CGW, le tableau 4.4 présente pour chaque méthode les valeurs des ratios qui correspondent aux temps moyens et maximums présenté dans le tableau 4.4. Nous avons calculé ces ratios avec la méthode citée si-dessus. En tenant compte de ces ratios, nous remarquons que TE consomme plus de temps CPU par rapport à TMH et AHS.

En consultant le tableau 5.8 de l'Annexe 4, nous constatons que TR est plus compétitive par rapport à TE sur les grandes instances. Encore, TR fournit 185 fois la meilleure solution connue sur 353 instances. Nous remarquons encore que TR fournit 13 fois les meilleures solutions notées avec (*). D'autre part, TR est en moyenne à 98.01% de la meilleure solution fournie par rapport aux méthodes autres que TE. Ce pourcentage peut être considéré comme acceptable par rapport au gain du temps de calcul obtenu.

		CGW	**TMH**	**AHS**	**TE**	**TR**
n	m	G_{tot}	G_{tot}	G_{tot}	G_{tot}	G_{tot}
100	4	13024	13339	13629	13503	13556
	3	15486	16041	16207	15956	16030
	2	17514	17901	18279	17925	18112
102	4	9450	9778	9822	9799	9727
	3	11124	11258	11344	11294	11229
	2	12678	12669	12812	12696	12745
66	4	16705	16775	17010	16910	16825
	3	19415	19395	19590	19505	19550
	2	22265	22170	22395	22370	22300
64	4	3582	3564	3570	3570	3564
	3	6300	6264	6342	6336	6342
	2	8964	9000	9012	8898	8940
33	4	6650	6700	6730	6670	6550
	3	8060	8160	8230	8150	8140
	2	9770	9840	9920	9860	9860
32	4	1490	1515	1515	1510	1460
	3	2010	1995	2000	2005	1960
	2	2525	2530	2535	2495	2510
21	4	1040	1040	1040	1040	1010
	3	1495	1495	1500	1500	1440
	2	2090	2090	2095	2095	2095
Total		191637	193519	195577	194087	193945

TABLE 4.3 – *Les sommes des gains des solutions obtenus par l'approche Tabou avec voisinage réduit organisés par classes d'instances*

CGW				
n	T_{moy}	$Ratio$	$Tmax$	$Ratio$
100	*	*	935,0	*
102	*	*	841,0	*
66	*	*	194,0	*
64	*	*	150,0	*
33	*	*	15,0	*
32	*	*	15,0	*
21	*	*	1,0	*
TMH				
n	T_{moy}	$Ratio$	$Tmax$	$Ratio$
100	184,5	1	797,0	1
102	84,4	1	433,0	1
66	16,8	1	71,0	1
64	14,1	1	46,0	1
33	2,5	1	7,0	1
32	1,4	1	3,0	1
21	*	*	*	*
AHS				
n	T_{moy}	$Ratio$	$Tmax$	$Ratio$
100	22,5	0,6	121,0	0,8
102	10,3	0,6	90,0	1,0
66	34,2	10,1	30,0	2,1
64	8,7	3,1	20,0	2,2
33	0,2	0,3	1,0	0,7
32	0,1	0,5	1,0	1,7
21	0,0	*	0,0	*
TE				
n	T_{moy}	$Ratio$	$Tmax$	$Ratio$
100	250,5	6,2	700,0	4,0
102	100,9	5,5	543,0	5,8
66	21,7	5,9	56,0	3,6
64	29,3	9,5	64,0	6,4
33	2,6	4,9	6,0	3,9
32	1,9	6,4	6,0	9,2
21	0,4	*	1,5	*
TR				
n	T_{moy}	$Ratio$	$Tmax$	$Ratio$
100	142,6	3,5	379,3	2,2
102	67,1	3,7	344,8	3,7
66	10,8	2,9	26,1	1,7
64	15,7	5,1	32,6	3,2
33	2,4	4,4	5,1	3,4
32	1,8	5,7	4,0	6,1
21	0,4	*	1,5	*

TABLE 4.4 – *Résumé des temps CPU obtenus par par l'approche Tabou complet TE et réduit TR et ceux obtenus parCGW, TMH, AHS*

4.5 Conclusion

Dans ce chapitre, nous avons présenté deux approches basées sur la recherche tabou. Nous utilisons dans la première approche TE une structure de voisinage étendue, par contre dans la deuxième approche nous avons essayer de réduire ce voisinage afin de minimiser les temps CPU.

Nous pouvons affirmer que TR est une approche fournissant des solutions de moyenne qualité raisonnable par rapport aux solutions fournis par les différentes méthodes de la littérature. L'intérêt de cette approche vient du fait qu'elle nécessite un temps CPU beaucoup moins réduit que les autres méthodes.

Notre approche tabou à voisinage réduit peut être améliorée pour qu'elle soit plus compétitive du point de vue de la qualité des solutions. L'une des stratégies est de raffiner d'avantage la procédure de recherche tabou en intégrant l'espace de recherche des solutions non réalisables. Par exemple nous pouvons intégrer la notion de terme de pénalité dans la fonction objectif (Gendreau *et al.*, 1994), en relaxant la contrainte de longueur maximale. Cette stratégie nous permet d'osciller entre les solutions réalisables et non réalisables dans le but maximiser d'avantage les chances de tomber sur un optimum global. Nous pouvons encore intégré la notion de la mémoire adaptative comme elle a été utilisée pour résoudre le PTVC par Rochat et Taillard (1995); Tarantilis et Kiranoudis (2002). Nous avons développé ces deux idées et elles font l'objet du chapitre suivant.

Les résultats obtenus par ces approches ont été publiés dans Khemakhem *et al.* (2006a, 2007a).

Chapitre 5

Stratégie d'oscillation et mémoire adaptative pour la recherche tabou

5.1 Introduction

Nous présentons, dans ce chapitre, deux approches plus sophistiquées de résolution du PmTS à l'aide de la recherche tabou. La première est basée sur une stratégie d'oscillation entre les solutions réalisables et non réalisables. Cette approche a été inspirée d'une heuristique de résolution du PTVC, appelé *TabouRoute*, proposée par (Gendreau *et al.*, 1994). La deuxième est basée sur le principe de la mémoire adaptative. Cette approche a été inspirée d'une heuristique de résolution du PTVC, appelée *BoneRoute*, proposée par Tarantilis et Kiranoudis (2002). Nos heuristiques utilisent la méthode tabou à voisinage réduit, présentée dans la section 4.4 du chapitre précédent.

Pour chaque méthode, nous commençons par présenter son schéma général puis nous la détaillons. Ensuite, nous précisons les paramétrages utilisés dans ces approches. Finalement, nous présentons les résultats numériques et les interprétons.

5.2 Approche Tabou avec stratégie d'oscillation

Dans cette section nous présentons une approche tabou pour la résolution du PmTS, elle est basée sur une stratégie d'oscillation entre solutions réalisables et non réalisables. Nous utilisons dans la fonction objectif un terme de pénalité en cas de non réalisabilité. Nous proposons une méthode pour normaliser la fonction objectif afin de tenir compte des ordres de grandeurs différentes. L'approche tabou, proposée ne comporte ni procédure d'intensification ni de diversification. Elle est composée principalement d'une exploration de la structure de voisinage réduit et une phase d'aspiration appelée de temps à autre. Nous avons testé cette approche sans et avec

normalisation et nous avons constaté que la normalisation a un effet positif sur le comportement de notre algorithme.

5.2.1 Schéma général de l'heuristique

Le processus de cette approche ressemble aux deux approches tabou décrites dans les deux sections précédentes. Il commence par une solution initiale générée par la méthode à deux phases avec la distance de Mahalanobis. Nous avons utilisée la structure de voisinage réduite utilisée dans l'approche de la section précédente. Pour les composants tabou, nous avons utilisé uniquement une procédure d'aspiration appelée dans le cas d'un meilleur voisin qui n'améliore pas la meilleure solution trouvée. L'une des caractéristiques de cette approche est qu'elle autorise les solutions non réalisables en relaxant la contrainte de longueur maximale. Le fait de visiter des champs de solutions non réalisables, permet au processus de recherche de diversifier les régions de l'espace de recherche visitées. L'algorithme proposé est basé sur une alternance entre régions admissible et non admissible. Ce principe est connu sous le nom d'oscillation (i.e. osciller entre solutions réalisables et non réalisables). La stratégie d'oscillation est étroitement liée aux origines de la recherche tabou. Elle a été efficacement utilisée par Glover et Kochenberger (1995); Hanafi et Fréville (1998) pour résoudre le problème de sac à dos multidimensionnel. La façon d'osciller peut prendre plusieurs formes. Une des plus simples est celle proposée par (Gendreau *et al.*, 1994) dans une méthode tabou pour résoudre le Problème de Tournées de Véhicules avec Capacités connue sous le nom de *Tabouroute*. Nous avons utilisé le principe d'oscillation utilisé dans *Tabouroute* qui consiste à modifier la fonction objectif de sorte qu'elle prend en considération la relaxation des contraintes lors de l'exploration du champs de recherche non réalisable.

L'algorithme 15 présente le schéma général de l'approche tabou avec stratégie d'oscillation. Dans cette approche, la non réalisabilité est défini par le fait qu'une solution Π de m tournées ($\Pi = \{\pi_1, ..., \pi_k, ..., \pi_m\}$) contienne au moins une tournée dépassant la longueur maximale L. Pour cela nous avons remplacé la fonction objectif $G(\Pi)$ par une autre fonction objectif avec pénalité $P(\Pi)$ définies comme suit :

$$G(\Pi) = \sum_{i=1}^{m} g(\pi_i) \tag{5.1}$$

$$P(\Pi) = \sum_{i=1}^{m} g(\pi_i) - \alpha \sum_{i=1}^{m} [l(\pi_i) - L]^+, \quad [x]^+ = max(0, x), \quad \alpha > 0 \tag{5.2}$$

Nous détaillons d'avantage, dans les prochaines sections, la nouvelle fonction objectif $P(\Pi)$ ainsi que le choix de la constante α.

La relaxation de la contrainte de longueur maximale est effectuée en acceptant des solutions possédant une ou plusieurs tournées de longueur plus grande que L. Le

dépassement de la longueur maximale L par une tournée est limité par un coefficient de tolérance β. Ce coefficient varie au cours du déroulement de l'algorithme.

Dans le cas où le meilleur voisin Ω^* sélectionné est réalisable, il est comparé à la meilleure solution réalisable trouvée Π^*. Lorsque Ω^* n'améliore pas Π^* une procédure d'aspiration définie dans les sections précédentes est appliqué (voir Algorithme 11). Dans le cas où Ω^* est non réalisable, elle est comparé par rapport à la meilleure solution non réalisable trouvée $\overline{\Pi^*}$. La procédure d'aspiration est appliquée aussi dans le cas de non réalisabilité. Si la meilleure solution trouvée est améliorée, *iter* prend la valeur nulle. L'algorithme s'arrête lorsque *iter* atteint $iterMax$.

Cette approche est décrite plus en détails par l'algorithme 15.

Algorithme 15 Schéma général de l'approche tabou avec pénalité

Étape 1 : Générer une solution initiale Π de gain total $G(\Pi)$ à l'aide de la méthode à deux phases.
Étape 2 : Initialiser
– la meilleure solution réalisable et non réalisable rencontrées $\Pi^* = \Pi$, $\overline{\Pi^*} = \Pi$,
– le nombre d'itération maximal $iterMax$,
– la taille Tl de la liste tabou *liste*,
– le facteur de pénalisation α,
– le facteur de relaxation de la contrainte de longueur β.
Répéter les étapes de 3 à 8 jusqu'à $iter = iterMax$.
Étape 3 : Sélectionner le meilleur voisin Ω^* de Π dans le voisinage réduit en évaluant par la fonction objectif $P(\Pi)$,
Étape 4 : Déclarer comme tabou les attributs du mouvement pris en considération lors du passage de Π à Ω^*, (les mettre dans la liste tabou *liste*),
Étape 5 : Mettre à jour α et β si nécessaire,
Étape 6 : Si Ω^* est réalisable
– Si $P(\Omega^*) > P(\Pi^*)$ alors $\Pi^* = \Omega^*$, $iter = -1$,
– Sinon appliquer aspiration,
Étape 7 : Si Ω^* est non réalisable
– Si $P(\Omega^*) > P(\overline{\Pi^*})$ alors $\overline{\Pi^*} = \Omega^*$,
– Sinon appliquer aspiration,
Étape 8 : $\Pi = \Omega^*$, $iter++$

5.2.2 Normalisation de la fonction objectif

La fonction objectif $P(\Pi)$, définit par l'équation 5.2, présente une somme de données hétérogènes. D'une part nous avons une sommation sur les gain des sommets visités, d'autre part nous avons une sommation sur les longueurs des tournées. Ces quantités non homogènes peuvent fausser l'évaluation d'une solution dans les zones

non réalisables. En effet, nous pouvons trouver des solutions non réalisables qui possèdent des valeurs de la fonction objectif négatives même si elle peuvent conduire à des régions de l'espace des solutions contenant des solutions de bonne qualité.

Pour éviter ce problème, nous avons proposé une normalisation de la fonction objectif utilisée lors de l'évaluation des voisins. La fonction objectif normalisée $\widehat{P}(\Pi)$ est décrite comme suit :

$$\widehat{P}(\Pi) = \sum_{i=1}^{m} \widehat{g}(\pi_i) - \alpha \sum_{i=1}^{m} [\widehat{l}(\pi_i) - \widehat{L}]^+ \tag{5.3}$$

avec

$$\widehat{g}(\pi_i) = \sum_{v_j \in \pi_i} \frac{g_j}{\underset{v_h \in V}{max}\{g_h\}}$$

$$\widehat{l}(\pi_i) = \sum_{(v_j,v_k) \in \pi_i} \frac{d_{jk}}{\underset{(v_h,v_p) \in A}{max}\{d_{hp}\}}$$

$$\widehat{L} = \frac{L}{\underset{(v_h,v_p) \in A}{max}\{d_{hp}\}}$$

Après la normalisation, les quantités sommés dans la fonction objectif ont des valeurs d'une part significatives et d'autre part homogènes. En effet, les quantités $\widehat{g}(\pi_i)$ et $\widehat{l}(\pi_i)$ sont des sommations de valeurs comprises entre 0 et 1.
(i.e. $0 \leq \frac{g_j}{\underset{v_h \in V}{max}\{g_h\}} \leq 1$ et $0 \leq \frac{d_{jk}}{\underset{(v_h,v_p) \in A}{max}\{d_{hp}\}} \leq 1$).

5.2.3 Paramétrage et initialisation

Les paramètres les plus importants de l'approche actuelle sont la taille de la liste tabou, le nombre d'itération maximal, fréquence d'aspiration, et les fréquences des changements des valeurs des paramètres α et β. Ces paramètres peuvent êtres statiques ou dynamiques. Nous détaillons dans ce qui suit les choix que nous avons adopté dans notre approche.

5.2.3.1 Liste tabou

Comme dans l'approche précédente, nous avons fixé la liste tabou à une taille $Tl = \frac{\sqrt{\overline{n} \times m}}{2}$ où $\overline{n} = n - \underline{n}$, $\underline{n}$ est le nombre des sommets qui ne peuvent pas figurer dans la solution (i.e. $\underline{n} = |\underline{V}|$ où $\underline{V} = \{\forall v_i \in V, d_{1i} + d_{in}) > L\}$) (voir section 1.2.2 du chapitre 1). Pour le codage de la liste tabou nous avons utilisé la matrice de taille $(n \times m)$ décrite dans la sous-section 4.4.2.1 du chapitre précédent.

5.2.3.2 Fréquence d'aspiration

La procédure d'aspiration est appelée que lorsqu'un voisin, réalisable ou non réalisable, n'améliore pas la meilleure solution (respectivement) réalisable ou non réalisable trouvée.

5.2.3.3 Nombre d'itérations maximal

Après $iterMax$ itérations, sans amélioration, la procédure de recherche tabou s'arrête et fourni la meilleure solution trouvée. Nous avons fixé $iterMax$ à quatre fois la taille du problème sans tenir compte des sommets qui ne peuvent pas figurer dans la solution ($iterMax = 4\overline{n}$).

5.2.3.4 Initialisation et fréquence de stabilité du paramètre α

Le paramètre de pénalisation α est initialisé à 1 pour commencer la recherche dans la zone réalisable. Après h itérations successives dans une zone réalisable, le paramètre α est divisé par deux dans le cas où α est strictement supérieur à 1. Après $\overline{h}$ itérations successives dans une zone non réalisable, le paramètre α est multiplié par deux. Nous avons fixé h à 30 et $\overline{h}$ à 10.

5.2.3.5 Initialisation et fréquence de stabilité du paramètre β

Le paramètre β est utilisé dans l'exploration du voisinage, il définit de combien une tournée peut excéder la longueur maximale L ($L = \beta \times L$). Le paramètre β est initialisé à 1 pour commencer la recherche dans une zone réalisable. Après h itérations successives dans une zone réalisable, le paramètre β est augmenté de 0.05. Après $\overline{h}$ itérations successives dans une zone non réalisable, le paramètre β est diminué de 0.05 dans le cas où β est strictement supérieur à 1.

5.2.4 Résultats numériques

5.2.4.1 Contexte et environnement

Nous avons codé l'approche tabou avec pénalité avec Java. Nous l'avons testé sur une machine Intel Pentium 4 - 2.4 Ghz - 256 Mo RAM sur les 353 instances benchmark publiées par (Chao *et al.*, 1996b). Afin de prouvé l'intérêt de la normalisation de la fonction objectif pénalisée, nous avons testé l'approche sans normalisation. Le tableau 5.1 présente un résumé du tableau 5.9 de l'Annexe 5. Il présente pour chaque approche les résultats numériques totaux obtenus pour un ensemble de groupes d'instances. Les notations suivantes sont considérées dans le tableau 5.1 :

– n : le nombre des sommets y compris les dépôts

		TP		$\widehat{TP}$	
n	m	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
100	4	13535	3838,5	13564	2713,9
	3	16027	6041,1	15996	3616,5
	2	17948	7340,0	17957	7705,4
102	4	9713	853,1	9716	876,8
	3	11133	1592,5	11115	1403,2
	2	12689	3294,0	12696	2940,5
66	4	16775	285,6	16805	272,6
	3	19510	437,3	19505	440,1
	2	22270	610,2	22265	568,2
64	4	3564	111,2	3564	111,0
	3	6342	209,6	6324	200,6
	2	8886	309,5	8886	295,7
33	4	6480	24,1	6530	18,7
	3	8120	45,9	8140	40,1
	2	9820	66,8	9800	60,9
32	4	1465	14,0	1465	9,6
	3	1960	23,1	1960	18,8
	2	2465	43,2	2460	38,2
21	4	1010	3,0	1010	1,4
	3	1410	4,4	1410	2,8
	2	2080	8,1	2080	6,1
Total		193202	25155,1	193248	21341,0

TABLE 5.1 – *Résumé des résultats numériques obtenus par l'approche Tabou avec pénalité*

- m : le nombre des tournées exigées par la solution
- TP : les résultats des meilleurs solutions obtenues avec l'approche tabou avec pénalité à fonction objectif non normalisée
- $\widehat{TP}$: les résultats des meilleurs solutions obtenues avec l'approche tabou avec pénalité à fonction objectif normalisée
- G_{tot} : le gain total obtenu par la solution des instances correspondantes avec des différentes longueurs maximales
- T_{cpu} : le temps total mis, en seconde, pour résoudre les instances correspondantes avec des différentes longueurs maximales
- $Total$: Les gains et les temps totaux obtenus par toutes les méthodes avec toutes les instances

	TP		$\widehat{TP}$	
n	T_{moy}	T_{max}	T_{moy}	T_{max}
100	307,5	830,5	250,6	838,0
102	99,0	501,5	90,0	418,1
66	18,0	45,5	17,3	44,6
64	26,3	49,9	25,3	47,1
33	2,3	5,3	2,0	5,0
32	1,7	4,7	1,4	4,2
21	0,5	1,0	0,3	0,8

TABLE 5.2 – *Résumé des temps CPU obtenus par l'approche Tabou avec pénalité*

Le tableau 5.2 présente un résumé des temps CPU de l'approche tabou avec pénalité à fonction objectif normalisée et non normalisée. Les notations suivantes sont considérées dans le tableau 5.2 :

- TP : la méthode tabou avec pénalité à fonction objectif non normalisée.
- $\widehat{TP}$: la méthode tabou avec pénalité à fonction objectif normalisée.
- n : la nombre des sommets y compris les dépôts.
- T_{moy} : Le temps moyen de résolution des instances de taille n correspondantes.
- T_{max} : Le temps maximal de résolution d'une instance de taille n.

5.2.4.2 Analyse des résultats

En consultant le tableau 5.1 nous pouvons constater que le gain total fourni par l'approche tabou avec la fonction objectif normalisée $\widehat{TP}$ est plus grand de quelque unités par rapport à l'approche avec la fonction objectif non normalisée TP. Cette constatation n'est pas suffisante pour affirmer l'intérêt de la normalisation. Par contre, en observant les temps de résolution de chaque ensemble d'instance (voir ligne *Total* de tableaux 5.1 et 5.2), nous pouvons constater que $\widehat{TP}$ arrive à résoudre toutes les instances du problème en un temps nettement moins important que TP. La différence entre les temps totaux de résolution des 353 instances des deux méthodes est d'environ une heure de calcul. Cette différence peut être expliquer du fait que lors de l'utilisation de la fonction objectif non normalisée, l'algorithme passe beaucoup de temps pour migrer de la zone de recherche non réalisable vers la zone de recherche réalisable. Cela est expliqué par le fait que l'algorithme accepte des solutions non réalisables de valeurs beaucoup plus importantes que la meilleure solution trouvée. Par la suite et lors de l'augmentation du paramètre de pénalisation α, l'algorithme effectue beaucoup plus d'itérations dans la zone non réalisable afin d'atteindre la zone de recherche réalisable.

Le tableau 5.9 de l'Annexe 5, présente les détails des résultats de cette approche avec et sans normalisation. Nous constatons que presque pour la majorité des ins-

tances les deux méthodes fournissent la même qualité de solution sauf que pour 31 et 37 instances (respectivement) $\widehat{TP}$, TP donne la meilleure solution.

Au point de vue comparaison par rapport au autres méthodes tabou décrites dans le sections précédentes et celles proposées par (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005; Archetti *et al.*, 2007b) (voir tableau 5.5 de l'Annexe 1), la méthode tabou avec pénalité présente une grande faiblesse au point de vue qualité de solution. En effet, elle est en moyenne à 97.41% de la meileures solution trouvée.

La méthode tabou avec pénalité n'est pas compétitive par rapport aux autres méthodes. Cela est dû à la relaxation de la contrainte de la longueur maximale. En effet, cette relaxation provoque un élargissement des régions de recherche et par conséquence l'éloignement, du processus de recherche, des zones prometteuses. Notre approche basée sur la recherche tabou peut être améliorer en changeant la structure de voisinage et en introduisant une mémoire adaptative comme dans la méthode nommé *BoneRoute* introduite par (Tarantilis et Kiranoudis, 2002) pour résoudre le Problème de Tournées de Véhicules avec Capacité.

5.3 Approche Tabou avec mémoire adaptative

Dans cette section, nous décrivons une approche pour la résolution du PmTS basée sur le principe de la mémoire adaptative. Cette approche a été inspirée de deux heuristiques de résolution du PTVC. La première a été proposée par Rochat et Taillard (1995) et la deuxième, appelée *BoneRoute*, a été proposée par Tarantilis et Kiranoudis (2002). Nous commençons par la présentation du schéma général de notre approche, ensuite nous détaillons ses différents composants et paramètres. Finalement, nous présentons les résultats numériques et les interprétons.

5.3.1 Schéma général de l'heuristique

Le processus général de notre approche est décrit comme suit : elle commence à générer un ensemble de TM solutions initiales en utilisant la méthode à deux phases, pour la résolution du PmTS, décrite dans le troisième chapitre. Partant de ces solutions de qualité moyenne, l'heuristique applique une procédure de recherche tabou avec voisinage réduit, décrite dans la section 4.4 du chapitre précédent, sur chacune d'elles afin d'améliorer d'avantage sa qualité. Les solutions améliorées par la procédure tabou sont triées selon leurs valeurs de la fonction objectif et stockées dans la mémoire des solutions. Tant que le critère d'arrêt n'est pas satisfait, les prochaine phases sont répétées.

En respectant les paramètres TC et FC, notre heuristique extrait les *sous-chaînes* de taille TC qui figurent au moins FC fois dans l'ensemble des solutions de la mémoire. En agrégeant les *sous-chaînes* en sommets, la méthode à deux phases

est appliquée pour générer une nouvelle solution du PmTS. Cette solution est améliorée par la procédure de recherche tabou utilisée lors de la génération des solutions initiales. La solution de plus mauvaise qualité, stockée dans l'ensemble de solutions, est remplacée par la nouvelle solution même si cette dernière est la plus mauvaise et à condition qu'elle ne soit pas déjà présente dans la mémoire. Ensuite, une nouvelle extraction des *sous-chaînes* a lieu. Le processus de notre heuristique s'arrête dés qu'un nombre d'itérations sans amélioration est atteint. L'algorithme 16 présente le schéma général de notre heuristique.

Algorithme 16 Schéma général de l'heuristique avec mémoire adaptative pour le PmTS

Étape 1 : Générer un ensemble de solutions pour le SVRP par la méthode à deux phases proposée décrite dans le troisième chapitre.
Étape 2 : Améliorer les solutions générées par la méthode de recherche tabou avec voisinage réduit décrite dans le chapitre précédent.
Étape 3 : Stocker les solutions générées dans la mémoire des solutions.
Répéter les étapes suivantes tant que le critère d'arrêt n'est pas satisfait
Étape 4 : Extraire les *sous-chaînes* en commun entre les solutions de la mémoire en respectant les paramètres TC et FC de taille et de fréquence minimale des *sous-chaînes*.
Étape 5 : Utiliser la méthode à deux phases pour générer une nouvelle solution en considérant les *sous-chaînes* extraits.
Étape 6 : Utiliser une méthode de recherche tabou pour améliorer la solution extraite.
Étape 7 : Éliminer de la mémoire la solution de plus mauvaise qualité.
Étape 8 : Insérer dans la mémoire la solution générée par l'Étape 6.

5.3.2 Détails de l'heuristique

La génération de la solution initiale, son amélioration, l'extraction des *sous-chaînes* et la construction d'une nouvelle solution constituent les points principaux de notre heuristique. Dans cette sous section, nous présentons en détail les éléments déjà mentionnées dans l'algorithme 16 et décrits plus en détails dans l'algorithme 17.

5.3.2.1 Génération des solutions initiales

Comme déjà mentionné, l'heuristique commence par la génération d'un ensemble de solutions du PmTS pour les stocker dans la mémoire, TM étant la taille de la mémoire. Cette phase est assurée par la méthode à deux phases pour la résolution

Algorithme 17 Schéma détaillé de l'heuristique avec mémoire adaptative pour le PmTS

Étape 1 : Initialisation des paramètres suivants : *iter*, compteur d'itérations. TM, la taille de la mémoire des solution notée *pools*. TC, la taille d'une *sous-chaîne*. FC, la fréquence minimale d'une *sous-chaîne* dans *pools*. $iterMax$, le nombre maximal des itérations sans amélioration de l'algorithme entier. $iterTMax$, le nombre maximal d'itérations de l'algorithme tabou lors de la génération des solutions initiales, modifiée lors de la construction d'une nouvelle solution avec les *sous-chaînes* extraites.

Répéter les étapes de 2 à 5 tant que *pools* ***contient moins de*** TM ***solutions différentes***.

Étape 2 : Classification d de V en m classes avec l'algorithme k-médianes avec la distance de Mahalanobis et un choix aléatoire uniforme des centres initiaux.

Étape 3 : Génération d'une solution Π en appliquant le routage avec la méthode tabou proposée par Gendreau *et al.* (1998b).

Étape 4 : Amélioration de Π avec la méthode tabou à voisinage réduit ($iterTMax$ itérations).

Étape 5 : Ajouter Π dans *pools* si elle n'y figure pas, sinon retourner à l'Étape 2.

Répéter les étapes de 6 à 14 tant que $iter < iterMax$.

Étape 6 : Extraction des *sous-chaînes* en commun entre les solutions de *pools*. Mettre chaque *sous-chaîne* extrait, de fréquence supérieure ou égale à FC, dans le pool de *sous-chaîne* noté *poolb*.

Étape 7 : En considérant que les *sous-chaînes* extraits sont des sommets agrégés, construire un nouveau graphe $G'(V', A')$ composé par *poolb* et les sommets de G privé des sommets composant les *sous-chaînes* de *poolb* .

Étape 8 : Classification de V' en m classes avec l'algorithme k-médianes avec la distance de Mahalanobis et un choix aléatoire uniforme des centres initiaux.

Étape 9 : Génération d'une solution Π avec le routage avec la méthode tabou proposée par Gendreau *et al.* (1998b).

Étape 10 : Transformer la solution Π en une solution Ω en considérant le graphe G au lieu de G'

Étape 11 : Appliquer sur chaque tournée de Ω l'heuristique *Genius* proposée par Gendreau *et al.* (1992),

Étape 12 : Amélioration de Ω avec la méthode tabou à voisinage réduit ($iterTMax$ itérations modifiée).

Étape 13 : $iter++$

Étape 14 : Ajouter Ω dans *pools*, a la place de la mauvaise solution, si elle n'y existe pas. $iter = 0$ dans le cas où Ω est classée comme la meilleure solution de *pools*.

du PmTS présentée dans le troisième chapitre. Brièvement, cette méthode consiste à construire sur le graphe G une partition de m classes en utilisant la méthode d'agrégation autour des centres mobiles (appelée *k-médianes*). Pour des raisons de performance nous avons utilisé la distance de Mahalanobis comme métrique d'évaluation. Une des caractéristiques de la méthode *k-médianes* est que la solution finale dépend des choix initiaux des centres des classes. En deuxième phase, l'heuristique applique une heuristique de routage sur chaque classe de la partition fournie. Cette procédure a été proposée par Gendreau *et al.* (1998b) pour résoudre le PVCS. En résolvant le PVCS sur chaque classe, nous obtenons une solution globale pour le PmTS en considérant toutes les tournées. Afin d'avoir différentes solutions, notre heuristique applique la méthode à deux phases TM fois avec des choix aléatoires uniformes des centres initiaux des classes lors de l'application de la méthode *k-médianes*.

5.3.2.2 Procédure d'amélioration avec la recherche tabou

Les solutions du PmTS fournies par la méthode à deux phases sont de qualité moyenne. En effet, l'efficacité de la méthode à deux phases a été montrée uniquement pour les problèmes du PmTS à un seul dépôt ou à deux dépôts très proches. Afin de remédier à ce problème, nous avons présenté, dans le chapitre précédent, des procédures basée sur la recherche tabou pour améliorer les solutions fournies par la méthode à deux phases. Afin de réduire le temps des calculs, nous avons choisi d'appliquer celle avec le voisinage réduit (voir section 4.4 du chapitre 4).

5.3.2.3 Extraction des *sous-chaînes*

L'extraction de l'ensemble des *sous-chaînes* en commun entre les solutions de la mémoire se fait de la façon suivante. En commençant par la solution Π_k de meilleure qualité et par sa tournée π_h^k de plus grand gain, les TC premiers sommets de la tournée π_h^k seront sélectionnés. Ensuite, une phase de recherche, dans les autres tournées, permet d'identifier la présence de la séquence des sommets sélectionnés. Si cette séquence des TC sommets existe au moins FC fois dans la mémoire de solution, elle est sélectionnée parmi les *sous-chaînes* en commun et la procédure passe au TC sommets suivants de la tournée π_h^k de la solution Π_k. Sinon, une autre séquence de sommets sera sélectionnée en décalant la séquence actuelle par un seul sommet. Tout ce traitement est appliqué sur toutes les tournées des solutions selon l'ordre décroissant de leurs qualités.

Il est important de noter que dans le cas où la solution générée, à la fin de chaque itération de l'heuristique, existe déjà dans la mémoire contenant les solutions, et afin d'éviter le problème de *cyclage*, la procédure d'extraction des *sous-chaînes*, à l'itération suivante, commence par la deuxième solution de meilleure qualité et non

plus par la première. La procédure d'extraction des *sous-chaînes* est décrite par l'algorithme 18.

Algorithme 18 Schéma de la procédure d'extraction des *sous-chaînes*

Étape 1 : Trier les TM solutions de la mémoire selon leurs qualités,
Étape 2 : Initialiser $Sol = 1$ indiquant le numéro de la solution selon l'ordre du triage,
Répéter les étapes de 3 à 11 tan que $Sol < TM$
Étape 3 : Trier les m tournées de la solution Sol selon l'ordre décroissant de leurs gains,
Étape 4 : Initialiser $k = 1$ indiquant la numéro de la tournée de la solution selon l'ordre du triage,
Répéter les étapes de 5 à 10 tan que $k \leq m$
Étape 5 : Initialiser $t = 1$ indiquant le début de la séquence de sommets à sélectionner,
Étape 6 : Sélectionner les TC sommets de la tournée k existants dans l'intervalle $[t, t + FC[$,
Étape 7 : Initialiser $f = 0$ indiquant le fréquence d'apparition actuelle,
Pour chaque solution d'ordre $h > Sol$ ***vérifier l'existence de la séquence de sommets sélectionnée (sous-chaîne) et faire l'étape suivante***
Étape 8 : Si la séquence existe dans la solution d'ordre h alors incrémenter f d'une unité,
Étape 9 : Si $f \geq FC$ alors mémoriser la *sous-chaîne* sélectionnée dans la mémoire des *sous-chaînes* et $t = t + TC$ sinon $t++$,
Étape 10 : Si $(t + TC - 1)$ dépasse le nombre des sommets de la tournée k alors $k++$,
Étape 11 : $Sol++$.

5.3.2.4 Construction d'une nouvelle solution

Après l'extraction des *sous-chaînes*, la construction d'une nouvelle solution est assurée par la méthode à deux phases en considérant les *sous-chaînes* extraits et les autres sommets du graphe. La particularité de cette phase réside dans le fait qu'il faut conserver l'information fournie par les *sous-chaînes* déjà extraits. De ce fait, il faut conserver dans la mesure du possible la structure de ces *sous-chaînes* dans la nouvelle solution générée. Pour cela et avant d'appliquer la méthode à deux phases, notre heuristique considère les *sous-chaînes*, privées de leurs extrémités $Ext1$ et $Ext2$, comme étant des sommets notés vb. Un nouveau graphe est donc généré (voir figure 5.1).

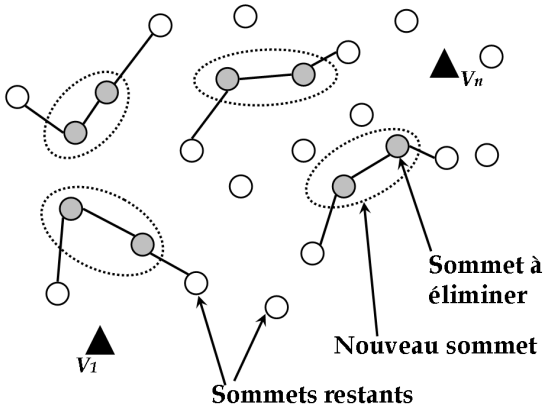

FIGURE 5.1 – Transformation du graphe

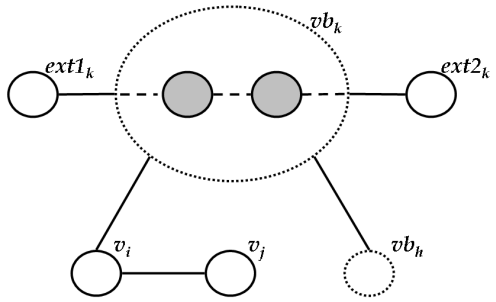

FIGURE 5.2 – Caractéristiques du graphe transformé

Ce nouveau graphe possède de nouvelles caractéristiques tels que les distances et les gains associés aux anciens et aux nouveaux sommets. Puisque chaque *sous-chaîne* privée de ses extrémités est considérée comme un sommet, sa visite par une tournée induit une distance supplémentaire dans sa longueur. Pour cela, cette information doit être introduite implicitement lors de la définition des caractéristiques du nouveau graphe. Afin de conserver les *sous-chaînes* dans la nouvelle solution, il faut faire de sorte que, si l'une des extrémités d'une *sous-chaîne* est visitée par une tournée, tous les autres sommets d'une *sous-chaîne* soient visités. Pour ce faire nous avons défini les nouvelles caractéristiques du nouveau graphe comme suit (voir la figure 5.2) :

- Les distances $d_{v_i v_j}$ entre les sommets non éliminés reste inchangées.
- Les distances $d_{vb_k v_j}$ entre chaque nouveau sommet vb_k et un sommet non éliminé v_i n'étant pas une des extrémités ou un nouveau sommet vb_h prennent la valeur infini ($d_{vb_k v_i} = d_{vb_k vb_h} = \infty$).

- Les distances $d_{vb_k ext1_k}$ et $d_{vb_k ext2_k}$ entre chaque nouveau sommet vb_k et ses extrémités $ext1$ et $ext2$ prennent la moitié de la distance de la chaîne formant le nouveau sommet vb_k.
- Les gains des sommets non éliminés sont inchangés.
- Le gain g_{vb_k} d'un nouveau sommet prend la somme des gains des sommets formant la *sous-chaîne* correspondante.
- Les gains des extrémités d'un nouveau sommet vb_k sont annulés ($g_{ext1_k} = g_{ext2_k} = 0$).

Lors de l'application de la méthode à deux phases sur le nouveau graphe, si l'un des sommets vb_k est sélectionné, cela n'est possible qu'après sélection de l'une de ses extrémité $ext1_k$ ou $ext2_k$. Après la visite de ce sommet la méthode choisit obligatoirement l'autre extrémité pour sortir. Tout cela est dû à l'introduction de la distance infini et du gain nul. De cette façon, nous conservons la forme des *sous-chaînes* dans la solution générée. La transformation d'un graphe G de taille n en un graphe G' présente un avantage dans certains cas. En effet, le nombre de sommets n' du nouveau graphe G' est déterminée comme suit : $n' = n + (NB \times (3 - TC))$ avec NB est le nombre des *sous-chaînes* extraites. Donc, dans le cas où TC est supérieur à 3 la taille du graphe est réduite.

5.4 Résultats numériques

Archetti *et al.* (2007b) ont proposé quatre heuristiques et ont montré que celles basées sur la recherche à voisinage variable comme étaient les meilleures heuristiques. En effet, celle nommé $FastVNS$ est considérée comme le meilleur compromis au point de vue qualité de solution et du temps de calcul. Par contre celle nommée $SlowVNS$ fournit des solutions meilleures mais au bout d'un temps de calcul important. Ces deux méthodes sont considérées comme les meilleures en se comparant aux méthodes déjà existantes dans la littérature. Pour cela nous comparons les résultats obtenus à ceux fournis par ces deux méthodes.

Nous avons codé notre heuristique en Java et nous l'avons testé sur une machine Intel Pentium 4, 3 Ghz et de 512 Mo Ram. Nous avons utilisé les 353 instances benchmark du publiées par Chao *et al.* (1996a). Après un certain nombre de tests numériques, nous avons fixé les paramètres de notre heuristique comme suit : $TM = 5$, $TC = \frac{n}{5 \times m}$, $FC = 2$, $Tl = \sqrt{\frac{n \times m}{2}}$, $iterTMax = 5$ lors de la génération des solutions initiales et $iterTMax = 2$ lors de l'amélioration de la nouvelle solution générée, $fa = 1$, $fd = \frac{iterTMax}{2}$, $fi = \frac{iterTMax}{4}$, le nombre d'itérations maximal $iterMax$ sans amélioration de l'heuristique est fixé à 10.

Le tableau 5.3 présente un résumé des résultats numériques fournis par notre méthode présentés en détails dans le tableau 5.10 de l'annexe 6. Il présente, pour chaque valeur de n fixée, les totaux des gains obtenus par l'approche pour des diffé-

rentes valeurs de m et L. Uniquement les valeurs maximales obtenues par $FastVNS$ et $SlowVNS$ sont considérées. Le tableau 5.4 présente pour chaque taille de problème le temps d'exécution moyen est maximum correspondant à chaque méthode. Nous considérons les notations suivantes :

- **SVNS** : Les résultats correspondant à $SlowVNS$.
- **FVNS** : Les résultats correspondant à $FastVNS$.
- **MA** : Les résultats correspondant à notre approche.
- G_{tot} : La somme des gains obtenus par l'ensemble des instances.
- T_{moy} : Le temps moyen pour la résolution de l'ensemble des instances de taille correspondante.
- T_{max} : Le temps maximum pour la résolution d'une instance de taille correspondante.
- $Ratio$: mesure la performance de la méthode en tenant compte de la machine sur laquelle est testée.
- $Total$: correspond au gain total obtenu en résolvant toutes les instances par la méthode correspondante.

En observant le tableau 5.3, nous remarquons que pour la majorité des classes d'instances notre approche est meilleure que **FVNS** et moins bonne que **SVNS**. En moyenne, la qualité des solutions obtenues par **MA** est à 99.93% des qualités des solutions obtenues par **SVNS** elle dépasse **FVNS** de 0.09%. Au point de vue temps d'exécution, nous remarquons que **MA** est nettement moins rapide que **FVNS** et plus rapide que **SVNS**, à l'exception de quelques problèmes de petites tailles.

Vue la non-conformité des machines sur lesquelles les tests ont été effectués, nous avons utilisé la relation utilisée dans la section 4.4.4.2 du chapitre précédent. Selon le logiciel d'évaluation de performance des machines *SiSoftware Sandra Standard*, notre machine fonctionne avec une performance de 3705 $Mflops$ et celle utilisée pour tester **FVNS** et **SVNS** une performance de 3492 $Mflops$. Nous avons pris **FVNS** comme méthode de référence et nous avons calculé le ratio pour chaque méthode (voir tableau 5.4).

En tenant compte de ces ratios et malgré la différence entre les machines, **MA** est considérée plus rapide que **SVNS** puisque ces ratios sont nettement très inférieurs à ceux de **SVNS** (voir tableau 5.4). En observant le tableau 5.10 de l'Annexe 6, nous remarquons qu'au pire des cas, notre approche nécessite quatre minutes pour résoudre une instance face à deux minutes pour **FVNS** et dix minutes pour **SVNS** (Archetti *et al.*, 2007b).

Afin d'élargir d'avantage la variété des instances de tests, nous avons testée cette approche sur un nouveau ensemble de 139 instances de taille $n = 121$ et $n = 151$. Nous avons généré ces instances en se basant sur des matrices de coordonnées prise des benchmark du PTVC [1]. Les demandes définies dans ces instances sont de valeurs

1. http ://or.ingce.unibo.it/research/cvrp-and-dcvrp/dcvrp.zip

		FVNS	**SVNS**	**MA**
n	m	G_{tot}	G_{tot}	G_{tot}
100	4	13629	13655	13649
	3	16207	16257	16237
	2	18279	18323	18292
102	4	9822	9859	9848
	3	11344	11387	11365
	2	12812	12850	12846
66	4	17010	17010	17005
	3	19590	19590	19590
	2	22395	22425	22430
64	4	3570	3594	3570
	3	6342	6342	6342
	2	9012	9012	9012
33	4	6730	6750	6730
	3	8230	8230	8230
	2	9920	9920	9920
32	4	1515	1515	1515
	3	2000	2000	2005
	2	2535	2535	2535
21	4	1040	1040	1040
	3	1500	1500	1500
	2	2095	2095	2095
Total		195577	195889	195756

TABLE 5.3 – *Résumé des résultats numériques de la méthode à mémoire adaptative*

trop proches d'où nous avons donc été obligé d'affecter à chaque sommets v_i un gain positif g_i aléatoirement choisi. Ensuite nous avons varier les longueurs maximales L avec un pas de 10 en commencent par 20 pour $n = 121$ et par 30 pour $n = 151$. L'Annexe 7 présente les résultats obtenus sur ces nouvelles instances.

5.5 Conclusion

Dans ce chapitre nous avons présenté deux approches heuristiques pour la résolution du PmTS. Une qui utilise une recherche tabou avec une stratégie d'oscillation et une qui combine une méthode à deux phases et une procédure de recherche tabou au sein d'un processus d'une recherche à mémoire adaptative. Expérimentalement, nous avons pu observer que la première approche est limité au point de vu de la

FVNS				
n	T_{moy}	$Ratio$	$Tmax$	$Ratio$
100	22,5	1	121,0	1
102	10,3	1	90,0	1
66	34,2	1	30,0	1
64	8,7	1	20,0	1
33	0,2	1	1,0	1
32	0,1	1	1,0	1
21	0,0	1	0,0	1
SVNS				
n	T_{moy}	$Ratio$	$Tmax$	$Ratio$
100	457,9	20,3	1118,0	9,2
102	309,9	30,0	911,0	10,1
66	158,9	4,7	394,0	13,1
64	147,9	16,9	310,0	15,5
33	10,2	67,9	19,0	19,0
32	7,8	59,8	22,0	22,0
21	0,0	*	1,0	*
MA				
n	T_{moy}	$Ratio$	$Tmax$	$Ratio$
100	130,7	6,2	329,6	2,9
102	69,2	7,1	197,1	2,3
66	37,1	1,2	91,0	3,2
64	46,8	5,7	97,1	5,2
33	8,3	58,7	23,5	25,0
32	7,2	58,5	24,1	25,6
21	2,9	*	5,7	*

TABLE 5.4 – *Résumé des temps d'exécution CPU*

qualité des solutions. Par contre la deuxième produit des solutions de bonne qualité et peut être considérée comme un nouveau compromis en la comparant aux autres méthodes déjà développées pour résoudre le PmTS.

Les résultats obtenus par ces approches ont été publiés dans Khemakhem *et al.* (2007b).

Conclusion et perspectives

Le travail que nous avons présenté concerne la résolution d'un problème d'optimisation rencontré dans le domaine du transport. Ce problème est appelé Problème de m-Tournées Sélectives PmTS ou Problème de Tournées de Véhicules Sélectives PTVS ou le plus souvent problème de course d'orientation par équipe. Le PmTS consiste à déterminer les tournées de m véhicules partant d'un dépôt et s'achevant à un autre. Chaque tournée doit desservir un sous-ensemble de clients dont chacun est caractérisé par une priorité ou un profit. Le problème consiste à maximiser le gain récolté par les m tournées sachant que chacune d'elles est limitée par une distance maximale.

Afin de résoudre le PmTS, nous avons conçu une approche à deux phases basée sur le principe connu sous le nom de "*Cluster first - Route second*". La première phase consiste à diviser l'ensemble des sommets, représentant les clients, sous forme de m sous-ensembles appelés classes. La deuxième phase consiste à appliquer sur chaque classe une procédure de routage afin d'avoir une solution au PmTS. Pour chaque phase, nous avons proposé différentes techniques qui sont enfin combinées pour avoir une solution au PmTS.

Dans la première approche, nous avons proposé une méthode, pour la première phase, basée sur la recherche à voisinage variable. Nous avons adapté, dans la deuxième phase, une procédure de routage proposée dans la littérature pour résoudre le problème du voyageur de commerce sélectif PVCS. La combinaison de ces deux méthodes nous a permit d'avoir des bonnes solutions du PmTS pour un nombre limité d'instances. Lors de l'examen de ces instances, nous avons constaté qu'elles sont caractérisées par le fait que les deux dépôts de départ et d'arrivée sont confondus ou très proches. Ce constat ainsi que des temps d'exécution significatifs nous ont conduit à concevoir une autre méthode pour assurer la première phase (phase de classification).

Nous avons adapté la méthode de classification basée sur l'agrégation autour des centres mobiles, connue sous le nom de *k-médianes*, afin d'assurer la phase de classification tout en gardant la même méthode de routage. Pour accroître la qualité des partitions de l'ensemble des sommets produites, nous avons testé cette méthode avec trois métriques différentes à savoir la distance *Euclidienne*, la distance de *Ma-*

halanobis et la distance basée sur le profilage. La rapidité d'exécution offerte par la méthode *k-médianes* nous permet de l'exécuter plusieurs fois avec différents choix initiaux fournissant différentes solutions. Cette caractéristique nous a permis par la suite de diversifier les solutions initiales de nos méthodes. Nous avons testé cette méthode avec les différentes métriques mentionnées. En examinant les résultats numériques, nous avons choisi d'utiliser la distance de *Mahalanobis* comme métrique pour la phase de classification avec la méthode *k-médianes*.

Dans la troisième approche, nous nous sommes intéressés à la phase de routage. Pour cela nous avons proposé une approche basée sur la recherche tabou. Cette approche considère la solution fournie par la deuxième méthode, avec la distance de *Mahalanobis*, comme étant la solution initiale. Ensuite et dans un processus de recherche locale de type recherche tabou, la méthode effectue des tentatives d'amélioration durant un nombre d'itérations sans amélioration fixé. Nous avons utilisé deux type de voisinage complet étendu et réduit. Comme leurs noms l'indiquent, la première structure explore un grand nombre de solutions voisines alors que la deuxième effectue une procédure de choix des solutions voisines avant l'exploration. Lors des tests numériques de cette approche avec les deux voisinages, nous avons constaté que le voisinage réduit fournit des solutions de qualité inférieur à celles obtenues par le voisinage étendu. Cette perte de qualité de solutions est compensée par le gain en temps d'exécution qui est moins important que celui requis par l'examen du voisinage complet.

Dans la quatrième approche, nous avons utilisé la méthode de recherche tabou à voisinage réduit au sein d'une méthode basée sur une stratégie d'oscillation. Elle consiste à considérer les solutions réalisables et non réalisables en utilisant la structure de voisinage réduite. Cette méthode a montré ses limites du point de vue qualité des solutions obtenues.

Dans la dernière approche, nous avons utilisé la méthode de recherche tabou à voisinage variable au sein d'une méthode basée sur le principe de la mémoire adaptative. Cette méthode utilise une mémoire de solutions de différentes qualités fournies par l'approche basée sur l'algorithme *k-médianes* avec la distance de *Mahalanobis* en considérant différents choix de centres initiaux. En utilisant la mémoire de solutions initialement générée, cette approche construit une solution en prenant des caractéristiques des solutions stockées dans la mémoire. Cette construction suit le principe de la méthode à deux phases, "*Cluster first - Route second*", en utilisant une classification d'un ensemble de sommets particulier. La nouvelle solution générée est ensuite stockée dans la mémoire de solutions. Ce processus est répété durant un certain nombre d'itérations fixé. Lors de l'évaluation de cette approche, nous avons constaté qu'elle constitue un nouveau compromis en la comparant aux solutions fournies par nos approches et par les approches proposées dans la littérature.

En conclusion, nous pouvons dire que la dernière méthode de résolution du PmTS a permis de fournir des résultats de bonne qualité en un temps raisonnable. Cette approche peut être améliorer en augmentant la taille du voisinage mais cela produit une augmentation du temps d'exécution.

Divers extensions du PmTS pourrais être considérées à l'avenir. Ainsi, afin de s'approcher d'avantage de la réalité et d'élargir les domaines d'application, nous pouvons ajouter la contrainte de capacité des véhicules. Ce problème a été traité récemment et pour la première fois par Archetti *et al.* (2007a) et nommé Capacitated Team Orienteering Problem CTOP. Nous pouvons encore ajouter la contrainte de temps de service ou la contrainte de fenêtre de temps.

La solution du PmTS est sous la fourme d'un planning de tournées journalier. Comme, parmi les caractéristiques du PmTS il n'est pas obligatoire de visiter tous les clients, les clients non visités peuvent l'être ultérieurement. Si nous voulons entrer dans un contexte plus général, nous pouvons proposer le planning des tournées pour plusieurs jours et non plus un planning journalier, tout en sachant que les gains associés aux clients deviennent variables au cours du temps (le gain récolté par un client servi le jour j diminue s'il est servi le jour $j + 1$). Nous pouvons appeler ce problème le Problème de m-Tournées Sélectives Périodiques PmTSP. L'étude de ce problème fait l'objet de nos perspectives de recherche.

Bibliographie

Amaya, A., A. Langevin et M. Trépanier, 2006. Un modèle de tournées de véhicules avec réapprovisionnements pour l'entretien de la signalisation routière. Dans *Proceedings of LT 2006, Mogistique et Transport, Hammamet, Tunisie*, pages 56–60.

Archetti, C., D. Feillet, A. Hertz et M. G. Speranza, 2007a. The capacitated team orienteering and profitable tour problems. Rapport technique, Les Cahiers du GERAD G-2007-31, École Polytechnique de Montréal, Montréal (Quebec) Canda.

Archetti, C., A. Hertz et M. G. Speranza, 2007b. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13 :49–76.

Aráoz, J., E. Fernández et C. Zoltan, 2006. Privatized rural postman problems. *Computers and Operations Research*, 33 :3432–3449.

Awerbuch, B., Y. Azar, A. Blum et S. Vempala, 1998. New approximation guarantees for minimum-weight k-trees and prize collecting salesman. *SIAM Journal on computing*, 28 :254–262.

Balas, E., 1989. The prize collecting traveling salesman problem. *Networks*, 19 :621–636.

Balas, E., 1995. The prize collecting traveling salesman problem ii : Polyhedral results. *Networks*, 25 :199–216.

Balas, E., 1999. New classes of efficiently solvable generalized traveling salesman problems. *Annals of Operations Research*, 86 :529–558.

Balas, E., 2002. *The Prize Collecting Traveling Salesman Problem and its applications*, pages 663–695. G. Gutin and A. Punnen, Kluwer Academic Publishers.

Balas, E. et G. Martin, 1985. Roll-a-round : Software package for scheduling the rounds of a rolling mill, copyright balas and martin associates.

Ball, G. H. et D. J. Hall, 1967. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12 :153–155.

Bar-Yehuda, R., G. Even et S. M. Shahar, 2005. On approximating a geometric prize-collecting traveling salesman problem with time windows. *Journal of Algorithms*, 55 :76–92.

Beasley, J. E., 1983. Route first-cluster second methods for vehicle routing. *Omega*, 11 :403–408.

Boulanger, C., F. Semet et P. V. Arellano, 2006. An integrated bicriteria model and a heuristic approach for a hazardous wastes transportation problem. Dans *Proceedings of Odysseus 2006, Third International Workshop on Freight Transportation and Logistics, Altea, Espagne*, pages 66–69.

Boulanger, C., F. Semet et P. V. Arellano, 2007. Un problème bi-objectif de localisation d'installations et d'élaboration de tournées de véhicules pour le transport de déchets dangereux. Dans *Proceedings de la conférence scientifique conjointe en Recherche Opérationnelle et Aide à la Décision, FRANCORO/ROADEF, Grenoble, France*, pages 105–106.

Boussier, S., D. Feillet et M. Gendreau, 2007. An exact algorithm for team orienteering problems. *4OR : A Quarterly Journal of Operations Research*, 5 :211–230.

Brandes, U., M. Gaertler et D. Wagner, 1989. Experiments on graph clustering algorithms. Dans S. LNCS, rédacteur, *Proceeding of 11th Europ. Symp. Algorithms ESA'03*.

Bérubé, J. F., M. Gendreau et J. Y. Potvin, 2007. An exact epsilon-constraint method for biobjective combinatorial optimization problems : Application to the travelling salesman problem with profits. *Soumit dans European Journal of Operational Research*.

Butt, S. E. et T. M. Cavalier, 1994. A heuristic for the multiple tour maximum collection problem. *Computers and Opérations Research*, 21 :101–111.

Chao, M., 1993. *Algorithms and Solutions to Multi-level Vehicle Routing Problem*. Thèse de doctorat, University of Maryland-College Park. MD. USA.

Chao, M., B. L. Golden et E. A. Wasil, 1996a. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88 :475–489.

Chao, M., B. L. Golden et E. A. Wasil, 1996b. The team orienteering problem. *European Journal of Operational Research*, 8 :464–474.

Dantzig, G. B. et J. H. Ramser, 1959. The truck dispatching problem. *Management Science*, 6 :80–91.

Deitch, R. et S. P. Ladany, 2000. The one-period bus touring problem : Solved by an effective heuristic for the orienteering tour problem and improvement algorithm. *European Journal of Operational Research*, 127 :69–77.

Dell'Amico, M., F. Maffioli et A. Sciomachen, 1998. A lagrangian heuristic for the prize-collecting travelling salesman problem. *Annals of Operations Research*, 81 :289–305.

DeSmet, Y. et L. Montano-Guzmán, 2004. Towards multicriteria clustering : An extension of the k-means algorithm. *European Journal of Operational Research*, 158 :390–398.

Diday, E., 1971. La méthode des nuées dynamiques. *Revue de Statistique Appliquée*, 19 :19–34.

Dillmann, R., B. Becker et V. Beckefeld, 1996. Practical aspects of a route planning for magazine and newspaper wholesale. *European Journal of Operational Research*, 90 :1–12.

Dubes, R. C., 1987. How many clusters arc best ? an experiment. *Pattern Recognition*, 20 :645–663.

Essaquote, H., N. E. Zahid, M. Limouri et A. Essaid., 2005. A new approach for unsupervised classification. *4OR : A Quarterly Journal of Operations Research*, 3 :39–49.

Feillet, D., 2001. *Problèmes de tournées avec gains : étude et application au transport inter-usines*. Thèse de doctorat, Ecole Centrale de Paris, Paris.

Feillet, D., M. Gendreau et P. Dejax, 2005. Traveling salesman problems with profits. *Transportation Science*, 39 :188–205.

Fischetti, M., J. J. Salazar-Gonzalez et P. Toth, 1998. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10 :133–148.

Fischetti, M. et P. Toth, 1988. *An additive approach for the optimal solution of the Prize-collecting Travelling Salesman Problem*, pages 319–343. B.L. Golden and A.A. Assad, Elsevier Science Publishers B.V., North-Holland.

Fisher, M. L. et R. Jaikumar, 1981. A generalized assignment heuristic for vehicle routing. *Networks*, 11 :109–124.

Forgy, E., 1965. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21 :768–769.

Garey, M. R. et D. S. Johnson, 1979. *A guide to the theory of NP-completeness*, page 47. Freeman, San Francisco.

Gendreau, M., A. Hertz et G. Laporte, 1992. New insertion and postoptimization procedures for the travelling salesman problem. *Operations Research*, 40 :1086–1094.

Gendreau, M., A. Hertz et G. Laporte, 1994. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40 :76–90.

Gendreau, M., G. Laporte et F. Semet, 1998a. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32 :263–273.

Gendreau, M., G. Laporte et F. Semet, 1998b. A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106 :539–545.

Gillet, B. E. et L. R. Miller, 1974. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22 :340–349.

Gimadi, E. K., N. I. Glebov et A. I. Serdyukov, 2004. On finding a cyclic tour and a vehicle loading plan yielding maximum profit. *Discrete Applied Mathematics*, 135 :105–111.

Glover, F., 1968. Surrogate constraints. *Operations Research*, 16 :741–749.

Glover, F., 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8 :156–166.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13 :533–549.

Glover, F., 1989. Tabu search, part i. *ORSA Journal on Computing*, 1 :190–206.

Glover, F., 1990. Tabu search, part ii. *ORSA Journal on Computing*, 2 :4–32.

Glover, F., 1996. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65 :223–253.

Glover, F. et G. A. Kochenberger, 1995. *Critical events tabu search for multidimensional knapsack problems*, pages 407–427. I.H. Osman, J.P. Kelly (Eds.), Kluwer Academic Publishers, Dordrecht.

Glover, F. et M. Laguna, 1993. *Tabu search, dans Modern Heuristic Techniques for Combinatorial Problems, C.R. Reeves*. Wiley.

Golden, B., G. Laporte et E. Taillard, 1997. An adaptive memory heuristic for a class of vehicle routing problems with min-max objective. *Computers and Operations Research*, 24 :45–52.

Golden, B. L., L. Levy et R. Vohra, 1987. The orienteering problem. *Naval Research Logistics*, 34 :307–318.

Golden, B. L., Q. Wang et L. Liu, 1988. A multi-faceted heuristic for the orienteering problem. *Naval Research Logistics*, 35 :359–366.

Golden, B. L. et E. A. Wasil, 1987. Computerized vehicle routing in the soft drink industry. *Operations Research*, 35 :6–17.

Gothe-Lundgren, M., F. Maffioli et P. Varbrand, 1995. A lagrangian decomposition approach for a prize collecting traveling salesman type problem. Rapport technique, LiTH-MATH-R-1995-10, Linkoping Institute of Technology, Linkoping, Sweden.

Gueguen, C., 1999. *Méthodes de résolution exacte pour les problèmes de tournées de véhicules*. Thèse de doctorat, Ecole Centrale de Paris, Paris.

Hachicha, M., M. J. Hodgson, G. Laporte et F. Semet, 2000. Heuristics for the multi-vehicle covering tour problem. *Computers and Operations Research*, 27 :29–42.

Hanafi, S. et A. Fréville, 1998. An efficient tabu search approach for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, 106 :659–675.

Hansen, P., 1986. The steepest ascent mildest descent heuristic for combinatorial programming. Dans *Proceedings of the Congress on Numerical Methods in Combinatorial Optimization, Capri, Italie*.

Hansen, P. et N. Mladenović, 1999. *An introduction to variable neighborhood search*, pages 433–458. Voss S. et al. Kluwer, Dordrecht.

Hansen, P. et N. Mladenović, 2001a. J-means : a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, 34 :405–413.

Hansen, P. et N. Mladenović, 2001b. Variable neiborhood search : Principles and applications. *European Journal of Operational Research*, 130 :449–467.

Hart, J. et S. Shogan, 1987. A semi-greedy heuristics : an empirical study. *Operations Research Letters*, 6 :7–14.

Hayes, M. et J. M. Norman, 1984. Dynamic programming in orienteering : Route choice and siting of controls. *Journal of the Operational Research Society*, 35 :791–796.

Jain, A. et R. C. Dubes, 1988. *Algorithms for clustering data*. Prentice Hall, Inc, Upper Saddle River, NJ, USA.

Jain, A. K., M. N. Murty et P. J. Flynn, 1999. Data clustering : a review. *ACM Computing Surveys*, 31 :264–323.

Jain, A. K., A. Topchy, M. H. C. Law et J. M. Buhmann, 2004. Landscape of clustering algorithms. Dans I. C. Society, rédacteur, *Proceeding of 17th International Conference on Pattern Recognition ICPR'04*.

Jozefowiez, N., F. Glover et M. Laguna, 2006. Multi-objective meta-heuristics for the traveling salesman problem with profits. Dans *Proceedings of Meta 2006, First workshop on metaheuristics, Hammamet, Tunisie*.

Jozefowiez, N., F. Glover et M. Laguna, 2007. A hybrid meta-heuristic for the traveling salesman problem with profits. Dans *Proceedings de la conférence scientifique conjointe en Recherche Opérationnelle et Aide à la Décision, FRANCORO/ROADEF, Grenoble, France*.

Kabadi, S. N. et A. Punnen, 1996. Prize-collecting traveling salesman problem. Dans *Proceedings of INFORMS Washington Conference*.

Kataoka, S., T. Yamada et S. Morito, 1998. Minimum directed 1-subtree relaxation for score orienteering problem. *European Journal of Operational Research*, 104 :139–153.

Keller, C. P., 1985. *Multiobjective routing through space and time : the MVP and TDVP problems*. Thèse de doctorat, Department of geography, the university of western Ontario, London, Ontario, Canada.

Keller, C. P., 1989a. Algorithms to solve the orienteering problem : A comparison. *European Journal of Operational Research*, 41 :224–231.

Keller, C. P., 1989b. A multiobjective vending problem : a generalization of the travelling salesman problem. *Environment and Planning B : Planning and Design*, 15 :447–460.

Khemakhem, M. et F. Semet, 2005. Nouvelles heuristique pour le problème de m-tournées sélectives. Dans *Proceeding de la Sixième conférence de la société Française de la Recherche Opérationnelle et d'Aide à la Décision ROADEF'05, Février, Tour, France*.

Khemakhem, M., F. Semet et H. Chabchoub, 2005. A hybrid heuristic for the selective vehicle routing problem. Dans IEEE, rédacteur, *Proceeding of the 6th Metaheuristics International Conference MIC'05, August, Vienna, Austria.*

Khemakhem, M., F. Semet et H. Chabchoub, 2006a. Le problème de tournées de véhicules sélectives : extensions de l'approche basée sur l'algorithme k-means. Dans *Proceeding de la conférence Métaheuristiques META'06, Novembre, Hammamet, Tunisie.*

Khemakhem, M., F. Semet et H. Chabchoub, 2007a. A classification based heuristic for the selective vehicle routing problem. *The Open Operational Research Journal, soumit.*

Khemakhem, M., F. Semet et H. Chabchoub, 2007b. Heuristique basée sur la mémoire adaptative pour le problème de tournées de véhicules sélectives. Dans SMC-IEEE, rédacteur, *Proceeding de la conférence en Logistique et Transport LT'07, Novembre, Sousse, Tunisie.*

Khemakhem, M., F. Semet, H. Chabchoub et M. Tmar, 2006b. Le problème de m-tournées sélectives : résolution par classification avec la méthode *k-means*. Dans *Proceeding des Sixièmes journées scientifiques des jeunes chercheurs en Génie Électrique et Informatique GEI'06, Mars, Hammamet, Tunisie.*

Khemakhem, M., F. Semet, H. Chabchoub et M. Tmar, 2006c. Le problème de m-tournées sélectives : une approche basée sur la méthode des centres mobiles. Dans SMC-IEEE, rédacteur, *Proceeding de la conférence en Logistique et Transport LT'06, Avril, Hammamet, Tunisie.*

Kleywegt, A. J., 1996. *Dynamic and stochastic models with freight distribution applications.* Thèse de doctorat, School of Industrial Engineering, Purdue University.

Kubo, M. et H. Kasugai, 1992. On symmetric subtour problems. *Japan Journal of Industrial and Applied Mathematics*, 9 :383–396.

Laporte, G., 1992a. The travelling salesman problem : An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59 :231–247.

Laporte, G., 1992b. The vehicle routing problem : An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59 :345–358.

Laporte, G. et S. Martello, 1990. The selective traveling salesman problem. *Discrete Applied Mathematics*, 26 :193–207.

Leifer, A. C. et M. Rosenwein, 1994. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research*, 73 :517–523.

Lenstra, J. K. et A. H. G. Rinnooy-Kan, 1981. Complexity of vehicle routing and scheduling problems. *Networks*, 11 :221–227.

Likasa, A., N. Vlassisb et J. J. Verbeekb, 2003. The global k-means clustering algorithm. *Pattern Recognition*, 36 :451–461.

Lin, S., 1965. Computer solutions of the travelling salesman problem. *Bell System Computer Journal*, 44 :2245–2269.

MacQueen, J. B., 1967. Some methods for classification and analysis of multivariate observations. Dans U. of California Press, rédacteur, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, tome 1, pages 281–297.

Mladenović, N. et P. Hansen, 1997. Variable neiborhood search. *Computers an Operations Research*, 24 :1097–1100.

Mullaseril, P. A., M. Dror et J. Leung, 1997. Split-delivery routing heuristics in livestock feed distribution. *Journal of the Operational Research Society*, 48 :107–116.

Osman, I. H., 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41 :421–451.

Pekny, J. F. et D. L. Miller, 1990. An exact parallel algorithm for the resource constrained travelling salesman problem with application to scheduling with an aggregate deadline. Dans *Proceedings of the ACM Eighteenth Annual computer Science Conference*.

Pirlot, M. et J. Teghem, 2003. *Résolution de problèmes de RO par les métaheuristiques*. Hermès, Paris.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31 :1985–2002.

Ramesh, P. et K. M. Brown, 1991. An efficient four-phase heuristic for the generalized orienteering problem. *Computers and Operations Research*, 18 :151–165.

Ramesh, P., Y. Yong-Seok et M. H. Karwan, 1992. An optimal algorithm for the orienteering problem. *ORSA Journal on Computing*, 4 :155–165.

Renaud, J. et F. F. Boctor, 2002. A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, 140 :618–628.

Rochat, Y. et E. Taillard, 1995. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1 :147–167.

Rodrigue, J. P., B. Slack, C. Comtois, M. Pouliot et J. Andrey, 2007. La géographie des transports dans. URL `http ://www.geog.umontreal.ca/Geotrans/`.

Rosenkrantz, D., R. Stearns et P. Lewis, 1974. Approximate algorithms for the traveling salesman problem. Dans *Proceedings of the 15th annual IEEE symposium of switching and automata theory*, pages 33–42.

Rosenkrantz, D. J., R. E. Stearns et P. M. Lewis, 1977. An analysis of several heuristics for the travelling salesman problem. *SIAM Journal on computing*, 6 :563–581.

Ryan, D. M., C. Hjorring et F. Glover, 1993. Extensions of the petal method for vehicle routing. *Journal of the Operational Research Society*, 44 :289–296.

Semet, F. et E. Taillard, 1993. Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, 41 :469–488.

Taillard., E., 1993. Parallel iterative search methods for vehicle routing problems. *Networks*, 23 :661–673.

Tang, H. et E. Miller-Hooks, 2005. A tabu search heuristic for the team orienteering problem. *Computers and Operations Research*, 32 :1379–1407.

Tarantilis, C. D. et C. T. Kiranoudis, 2002. Boneroute : An adaptive memory-based method for effective fleet management. *Annals of Operations Research*, 115 :227–241.

Tarantilis, C. D. et C. T. Kiranoudis, 2007. A flexible adaptive memory-based algorithm for real-life transportation operations : Two case studies from dairy and construction sector. *European Journal of Operational Research*, 179 :806–822.

Thorndike, R., 1953. Who belongs in a family ? *Psychometrika*, 18 :267–276.

Toth, P. et D. Vigo, 2001. *The Vehicle Routing Problem*. SIAM Society for Industrial and Applied Mathematics.

Tricot, M. L. et M. Donegani, 1989. Formules de réactualisation pour une famille d'indices de proximité inter-classe en classification hiérarchique. *RAIRO Recherche Opérationnelle*, 23 :165–192.

Tsiligirides, T., 1984. Heuristic methods applied to orienteering problem. *Journal of the Operational Research Society*, 35 :797–809.

Wang, Q., X. Sun, B. L. Golden et J. Jia, 1995. A using artificial neural networks to solve the orienteering problem. *Annals of Operations Research*, 61 :111–120.

Wren, A. et A. Holliday, 1972. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*, 23 :333–344.

Annexe1

Le tableau 5.5 présente les résultats numériques obtenus pour le PmTS avec la méthode de classification à voisinage variable (notée **CRVV**), ceux obtenus par Chao *et al.* (1996b) notée par **CGW**, par Tang et Miller-Hooks (2005) notée par **TMH** et le meilleure compromis par (Archetti *et al.*, 2007b) notée par **AHS**. Les notations suivantes sont considérées :

- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigées par la solution
- L : la longueur maximale d'une tournée
- G_{tot} : le gain total obtenu par la solution de l'instance correspondante
- T_{cpu} : le temps mis, en secondes, pour résoudre l'instance correspondante.

TABLE 5.5 – *Résultats numériques de la méthode à deux phases avec la classification par la méthode à voisinage variable (CRVV)*

			CGW	**TMH**	**AHS**	**CRVV**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}
102	4	100	1066	1067	1077	1073	246,0
		95	990	1019	1021	1022	248,5
		90	886	966	970	961	253,9
		85	882	905	899	907	258,5
		80	801	832	846	832	251,4
		75	728	776	770	781	209,5
		70	683	726	715	726	241,7
		65	610	643	646	639	213,2
		60	562	576	588	561	136,9
		55	516	503	518	502	137,0
		50	462	462	462	437	112,6
		45	338	359	366	354	85,6
		40	283	285	285	279	48,9
		35	209	217	217	217	25,9

à suivre...

TAB5.5 : *suite...*

			CGW	**TMH**	**AHS**	**CRVV**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}
		30	156	164	164	158	7,3
		25	123	123	123	91	2,6
		20	79	79	79	79	1,2
		15	46	46	46	46	1,0
		10	30	30	30	30	0,9
	3	133,3	1095	1098	1112	1111	303,6
		126,7	1064	1061	1079	1062	305,6
		120	1008	1011	1024	1020	327,7
		113,3	943	966	978	971	356,9
		106,7	919	922	926	914	353,4
		100	848	874	871	843	355,3
		93,3	813	789	814	784	360,7
		86,7	754	756	745	720	354,6
		80	676	681	681	653	346,9
		73,3	602	632	632	610	449,1
		66,7	539	563	562	522	290,6
		60	466	481	487	478	405,5
		53,3	419	416	425	416	242,5
		46,7	338	344	344	317	124,3
		40	235	247	247	231	84,7
		33,3	163	175	175	144	28,8
		26,7	117	117	117	105	19,9
		20	79	79	79	79	2,0
		13,3	46	46	46	30	1,4
	2	200	1173	1165	1168	1165	402,0
		190	1127	1116	1136	1119	405,3
		180	1082	1067	1094	1067	398,0
		170	1031	1017	1038	1022	397,5
		160	987	987	1002	982	434,6
		150	934	914	945	905	357,7
		140	864	864	883	864	387,4
		130	811	817	824	815	381,5
		120	758	767	759	767	321,9
		110	693	702	704	699	302,3
		100	633	638	643	638	303,0
		90	576	578	575	578	287,7
		80	517	521	521	520	297,1
		70	453	459	459	459	274,2

à suivre...

TAB5.5 : *suite...*

			CGW	**TMH**	**AHS**	**CRVV**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}
		60	379	382	387	382	221,7
		50	275	290	289	290	200,0
		40	190	190	190	170	178,6
		30	101	101	101	110	133,4
		20	64	64	64	64	100,7
		10	30	30	30	30	98,9
66	4	32,5	1545	1575	1620	1525	33,6
		31,2	1490	1520	1520	1520	34,0
		30	1420	1410	1450	1470	61,7
		28,8	1380	1380	1390	1370	46,9
		27,5	1310	1310	1320	1320	33,7
		26,2	1260	1275	1300	1250	33,7
		25	1160	1100	1160	1110	33,7
		23,8	1020	1000	1030	990	33,2
		22,5	950	960	960	910	33,7
		21,2	860	860	860	810	33,5
		20	750	760	765	720	33,6
		18,8	675	680	690	650	33,4
		17,5	620	620	620	580	33,1
		16,2	495	555	555	470	33,1
		15	430	430	430	390	32,9
		13,8	340	340	340	325	32,0
		12,5	340	340	340	240	30,8
		11,2	240	240	240	200	31,1
		10	140	140	140	170	31,1
		8,8	140	140	140	110	31,3
		7,5	80	80	80	80	31,3
		6,2	20	20	20	50	31,3
		5	20	20	20	20	31,3
		3,8	20	20	20	20	31,3
	3	43,3	1635	1635	1635	1615	101,6
		41,7	1580	1580	1595	1590	270,1
		40	1530	1530	1555	1520	94,5
		38,3	1450	1465	1485	1460	123,3
		36,7	1400	1410	1425	1395	84,1
		35	1330	1330	1345	1310	130,5
		33,3	1250	1240	1260	1255	360,6
		31,7	1175	1175	1190	1175	100,4

à suivre...

TAB5.5 : *suite...*

			CGW	**TMH**	**AHS**	**CRVV**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}
		30	1105	1115	1125	1125	100,3
		28,3	1060	1065	1070	1040	178,8
		26,7	990	990	990	940	128,7
		25	870	835	870	850	170,4
		23,3	755	755	755	740	112,4
		21,7	650	650	650	660	63,6
		20	595	575	595	600	83,8
		18,3	480	495	495	505	148,9
		16,7	470	470	470	460	84,1
		15	335	335	335	310	71,1
		13,3	255	260	260	260	74,4
		11,7	185	185	185	190	42,3
		10	110	110	110	135	135,0
		8,3	110	95	95	80	88,0
		6,7	60	60	60	55	72,6
		5	20	20	20	20	184,1
		3,3	15	15	15	10	197,3
	2	65	1680	1665	1670	1670	76,6
		62,5	1635	1630	1635	1630	57,6
		60	1600	1610	1590	1595	57,4
		57,5	1545	1560	1560	1555	57,3
		55	1490	1500	1500	1495	55,1
		52,5	1450	1445	1460	1435	54,9
		50	1380	1380	1400	1360	55,6
		47,5	1310	1310	1340	1285	55,6
		45	1250	1260	1260	1240	55,7
		42,5	1195	1185	1195	1050	55,9
		40	1150	1090	1150	1045	57,5
		37,5	1010	975	1020	935	55,1
		35	920	920	925	885	56,3
		32,5	855	860	860	800	55,2
		30	790	770	800	735	55,7
		27,5	670	670	670	670	56,4
		25	580	560	580	555	56,7
		22,5	480	480	480	460	56,3
		20	395	410	410	365	56,7
		17,5	315	320	320	315	57,2
		15	240	240	240	205	63,9

à suivre...

TAB5.5 : *suite...*

			CGW	**TMH**	**AHS**	**CRVV**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}
		12,5	175	180	180	155	57,7
		10	80	80	80	85	57,8
		7,5	50	50	50	50	57,2
		5	20	20	20	15	57,6

Annexe2

Le tableau 5.6 présente les résultats numériques obtenus pour le PmTS avec la méthode à deux phases avec classification par l'algorithme k-means. Cet algorithme est appliqué avec trois métriques différentes telles que Euclidienne, Mahalanobis et profilage. Il a été testé sur une machine (*Pentium 4, 512 Mo*). Les notations suivantes sont considérées :

- Euc : les résultats obtenus avec la distance Euclidienne
- Mah : les résultats obtenus avec la distance de Mahalanobis
- Pro : les résultats obtenus avec la distance de profilage
- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigé par la solution
- L : la longueur d'une tournée
- G_{tot} : le gain total obtenu par la solution de l'instance correspondante
- T_{cpu} : le temps mis, en secondes, pour résoudre l'instance correspondante.

TABLE 5.6 – *Résultats numériques de la méthode à deux phases avec la classification par k-means utilisant trois métriques différentes*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
102	2	200	1147	13,109	1159	22,522	1111	18,477
		190	1062	12,781	1121	15,683	1104	19,799
		180	1012	12,343	1080	14,821	1036	16,093
		170	939	10,828	1012	13,590	1004	16,423
		160	941	20,234	976	11,967	941	18,627
		150	852	8,984	915	19,218	898	15,953
		140	806	8,406	863	13,129	831	14,701
		130	818	12,0	818	13,509	818	18,397
		120	759	11,36	758	12,778	761	13,980
		110	705	7,641	705	8,602	705	10,645
		100	641	8,891	642	7,410	633	11,096

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		90	579	5,953	579	6,420	579	8,512
		80	517	4,891	517	5,398	517	7,441
		70	453	5,39	457	4,286	457	5,669
		60	380	4,187	387	4,176	387	4,336
		50	290	2,797	290	2,744	290	2,283
		40	190	1,765	190	1,593	190	1,552
		30	100	0,937	100	0,731	101	1,091
		20	32	0,859	64	0,601	32	0,941
		10	16	0,844	30	0,571	16	0,911
102	3	133,3	1092	15,719	1110	16,454	1019	13,309
		126,7	1052	15,547	1058	21,111	1041	18,146
		120	1019	14,375	1019	19,948	991	23,764
		113,3	964	10,015	950	14,350	954	17,285
		106,7	916	8,813	910	13,540	808	13,049
		100	844	14,484	859	12,588	766	10,325
		93,3	765	7,781	792	7,721	753	14,842
		86,7	745	9,11	735	7,641	734	13,489
		80	672	7,969	667	6,789	644	10,194
		73,3	614	5,5	626	7,411	612	8,532
		66,7	546	6,281	557	7,461	493	4,797
		60	478	5,063	478	5,037	450	5,218
		53,3	375	2,906	398	4,207	415	3,035
		46,7	293	2,672	307	1,742	307	2,113
		40	228	2,062	223	1,212	228	1,692
		33,3	152	1,156	157	0,871	166	1,162
		26,7	91	1,047	87	0,681	117	1,342
		20	32	1,0	48	0,611	32	1,022
		13,3	16	0,969	32	0,581	16	0,982
102	4	100	1045	12,578	1062	13,740	968	15,332
		95	992	15,25	1022	12,348	908	12,047
		90	895	8,391	953	11,326	935	12,228
		85	897	12,578	900	7,701	870	13,319
		80	812	11,781	832	7,170	785	16,123
		75	767	8,5	772	8,432	738	8,302
		70	680	7,438	716	7,300	676	8,212
		65	613	6,343	638	4,767	593	5,979
		60	571	5,281	567	5,488	541	5,688
		55	447	3,172	496	4,076	474	4,446

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		50	399	3,438	444	3,025	444	2,794
		45	294	2,0	342	2,173	330	2,604
		40	264	2,172	261	1,853	260	1,753
		35	184	1,328	203	0,871	197	1,182
		30	145	1,203	141	0,761	147	1,402
		25	91	1,172	109	0,701	99	1,412
		20	32	1,125	79	0,600	32	1,082
		15	16	1,125	30	0,570	16	1,101
		10	16	1,094	30	0,571	16	1,091
100	2	120	1282	25,937	1286	32,717	1289	44,043
		115	1281	26,969	1272	32,457	1266	52,775
		110	1235	26,313	1258	33,058	1248	50,322
		105	1203	30,359	1228	28,521	1234	45,255
		100	1152	20,234	1201	35,451	1150	29,332
		95	1138	18,797	1170	35,161	1150	29,742
		90	1120	18,922	1114	29,282	1148	54,639
		85	1083	19,062	1078	27,890	1083	26,668
		80	1024	18,344	1029	27,750	1010	22,672
		75	1004	16,187	959	24,035	976	21,180
		70	915	19,359	911	21,872	915	20,029
		65	892	12,937	859	17,265	865	21,160
		60	809	9,797	784	16,794	806	15,462
		55	716	8,375	725	15,773	680	8,953
		50	655	7,546	676	11,366	667	12,087
		45	591	6,313	597	10,015	589	10,345
		40	508	5,766	497	7,100	508	8,512
		35	448	4,39	440	5,708	426	4,927
		30	341	5,625	341	6,168	327	4,456
		25	158	1,407	161	1,231	178	1,662
100	3	80	1242	25,968	1224	31,034	1220	27,730
		76,7	1265	31,219	1204	28,571	1214	28,631
		73,3	1232	20,969	1161	27,770	1185	25,026
		70	1181	20,078	1114	23,243	1131	28,641
		66,7	1177	24,891	1184	25,397	1121	26,108
		63,3	1128	16,906	1093	22,322	1081	21,391
		60	1063	13,907	1034	25,227	1051	20,770
		56,7	953	12,797	997	18,727	1001	18,967
		53,3	919	11,532	907	16,964	897	15,462

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		50	858	10,063	872	19,589	886	26,418
		46,7	806	9,11	809	14,551	827	18,016
		43,3	747	8,5	754	15,812	799	15,922
		40	673	7,219	684	11,566	666	10,074
		36,7	646	8,656	594	9,374	627	9,133
		33,3	559	6,859	544	7,621	532	6,410
		30	446	7,25	428	4,467	442	4,867
		26,7	309	3,813	316	2,644	326	4,156
		23,3	148	1,25	144	1,141	145	1,482
		20	26	0,985	38	0,912	26	0,961
100	4	60	1211	17,062	1160	20,529	1168	28,511
		57,5	1185	15,859	1149	20,479	1133	24,205
		55	1099	13,0	1096	14,161	1132	29,313
		52,5	1080	17,0	1041	13,169	1040	21,140
		50	1015	11,938	977	16,133	994	20,149
		47,5	952	14,828	923	15,532	974	25,236
		45	904	15,11	885	13,619	918	13,539
		42,5	867	9,5	830	15,743	826	14,931
		40	827	9,328	787	14,140	824	10,626
		37,5	769	7,562	760	10,776	770	10,005
		35	703	8,61	680	8,032	698	9,223
		32,5	602	7,125	614	8,332	596	10,835
		30	515	8,203	508	3,685	491	6,569
		27,5	425	4,812	405	2,404	416	5,959
		25	229	2,703	284	2,984	226	1,913
		22,5	123	1,328	159	1,252	123	1,321
		20	26	1,109	38	0,731	26	1,052
66	2	65	1660	9,047	1680	7,892	1660	8,752
		62,5	1645	12,468	1640	9,333	1635	9,594
		60	1555	13,719	1610	10,045	1575	10,556
		57,5	1520	13,078	1565	10,776	1520	10,705
		55	1455	13,703	1500	10,485	1470	11,196
		52,5	1395	9,984	1460	10,365	1450	10,486
		50	1350	10,14	1360	9,944	1400	16,313
		47,5	1320	18,719	1315	13,069	1280	13,530
		45	1240	11,609	1250	18,847	1205	9,834
		42,5	1150	10,687	1190	8,993	1170	8,923
		40	1070	15,485	1060	24,024	1150	16,374

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		37,5	965	6,531	1010	9,925	960	10,105
		35	895	5,984	920	6,439	900	8,352
		32,5	835	5,579	825	6,009	835	5,848
		30	780	6,36	740	5,147	770	6,730
		27,5	670	7,266	645	4,707	670	7,631
		25	580	3,969	570	5,318	580	9,163
		22,5	480	3,266	480	4,537	480	4,576
		20	385	2,75	400	4,446	400	4,487
		17,5	315	2,235	315	2,213	320	3,044
		15	240	1,938	240	1,422	240	2,394
		12,5	175	1,156	180	0,761	180	1,652
		10	75	0,922	80	0,420	80	0,771
		7,5	20	0,5	50	0,441	20	0,450
		5	10	0,5	20	0,460	10	0,460
66	3	43,3	1595	11,265	1635	9,233	1610	8,092
		41,7	1550	11,937	1560	13,760	1560	8,111
		40	1440	8,875	1495	11,416	1515	14,831
		38,3	1420	11,281	1475	12,558	1455	12,308
		36,7	1410	8,282	1400	10,835	1365	11,426
		35	1275	7,547	1320	10,806	1315	11,156
		33,3	1215	7,125	1250	9,754	1245	8,072
		31,7	1165	6,89	1165	9,374	1175	7,411
		30	1120	8,703	1095	6,860	1120	6,950
		28,3	1055	6,39	1035	6,169	1050	8,563
		26,7	970	10,0	990	6,519	990	8,352
		25	870	7,516	870	9,243	845	8,802
		23,3	755	6,468	755	13,290	755	4,927
		21,7	635	4,063	650	4,216	650	4,096
		20	565	4,859	575	4,787	575	3,625
		18,3	480	2,906	485	3,014	495	4,046
		16,7	470	3,515	470	3,966	470	3,575
		15	330	2,5	335	3,235	325	2,524
		13,3	240	2,219	260	1,272	260	1,743
		11,7	160	1,453	185	1,362	185	1,272
		10	105	1,375	110	0,611	110	0,781
		8,3	25	0,656	95	0,681	25	0,521
		6,7	20	0,734	60	0,520	20	0,531
		5	10	0,735	20	0,401	10	0,531

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		3,3	5	0,703	15	0,401	5	0,521
66	4	32,5	1510	10,031	1545	12,098	1490	9,594
		31,2	1470	7,75	1520	7,170	1435	8,242
		30	1350	7,219	1440	7,782	1430	7,621
		28,8	1325	6,594	1355	7,050	1335	9,574
		27,5	1275	9,391	1320	6,369	1275	9,504
		26,2	1270	9,0	1300	6,780	1300	6,880
		25	1110	6,234	1150	11,076	1110	8,362
		23,8	1000	5,484	990	7,200	1005	10,074
		22,5	950	6,516	960	10,155	960	10,144
		21,2	805	4,609	825	7,601	825	7,721
		20	705	5,391	735	5,318	740	7,060
		18,8	690	6,297	670	3,645	690	5,197
		17,5	520	3,781	620	3,485	620	4,146
		16,2	525	3,579	485	2,754	520	3,576
		15	400	3,578	400	1,883	425	2,984
		13,8	290	2,641	340	1,722	340	2,053
		12,5	275	1,859	295	1,673	280	1,763
		11,2	185	1,234	200	1,001	190	1,182
		10	105	1,531	140	0,742	140	0,731
		8,8	35	0,75	95	0,921	110	0,591
		7,5	20	0,766	80	0,631	20	0,591
		6,2	10	0,75	20	0,601	10	0,591
		5	10	0,75	20	0,501	10	0,601
		3,8	5	0,766	20	0,470	5	0,581
64	2	40	1200	11,657	1188	21,832	1200	12,949
		37,5	1098	10,422	1068	14,652	1098	11,437
		35	1110	17,203	1104	21,571	1110	18,947
		32,5	1032	9,625	1032	13,239	1032	10,545
		30	936	8,234	936	11,296	936	9,223
		27,5	888	7,375	888	10,084	888	7,991
		25	774	6,093	780	8,522	774	6,539
		22,5	648	6,344	660	6,890	654	8,552
		20	528	3,797	534	5,808	528	5,628
		17,5	360	4,031	354	5,338	360	4,987
		15	156	0,813	186	1,903	156	0,791
64	3	26,7	1158	9,828	1134	17,435	1158	10,585
		25	1062	8,468	1026	11,286	1062	9,213

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		23,3	978	7,5	1002	14,481	978	8,012
		21,7	888	6,343	876	8,342	882	10,935
		20	756	5,516	750	6,990	762	7,681
		18,3	564	4,89	588	6,780	564	6,319
		16,7	396	1,969	438	5,608	408	2,854
		15	198	1,0	252	2,233	198	0,921
64	4	20	960	6,828	948	11,246	984	11,527
		18,8	840	13,36	846	11,407	804	12,188
		17,5	660	6,86	594	4,697	606	4,526
		16,2	450	2,75	492	7,050	480	3,976
		15	240	1,172	330	1,913	240	1,082
33	2	55	750	3,656	800	2,914	750	3,936
		52,5	750	3,375	800	2,945	750	3,635
		50	750	3,031	730	3,174	750	3,195
		47,5	730	3,187	710	3,445	730	3,334
		45	710	3,469	680	2,594	710	3,676
		42,5	690	3,297	660	3,766	690	3,465
		40	650	3,562	620	3,875	650	3,705
		37,5	610	3,578	570	3,015	610	3,756
		35	590	3,469	560	2,844	560	3,635
		32,5	520	3,109	540	3,695	520	3,014
		30	490	2,843	460	3,275	490	2,834
		27,5	460	2,704	420	2,774	460	2,734
		25	400	2,25	370	2,584	370	2,453
		22,5	350	2,015	360	2,363	330	2,063
		20	290	1,563	290	2,073	290	1,613
		17,5	250	1,375	250	1,312	250	1,802
		15	220	1,187	220	1,042	210	1,242
		12,5	170	1,094	170	0,731	170	0,681
		10	150	0,719	150	0,491	150	0,551
		7,5	90	0,422	90	0,370	80	0,280
33	3	36,7	730	2,5	690	2,213	730	2,954
		35	710	2,219	700	2,133	710	2,624
		33,3	710	2,062	660	2,304	710	2,434
		31,7	630	2,594	630	2,473	630	2,453
		30	590	2,125	620	2,073	600	2,404
		28,3	590	1,907	580	2,734	590	2,213
		26,7	550	1,828	530	2,774	530	3,145

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		25	480	2,297	490	2,594	500	2,103
		23,3	460	2,109	460	2,023	460	2,002
		21,7	430	1,922	430	3,645	430	1,682
		20	370	1,75	370	1,673	370	2,564
		18,3	330	1,5	330	1,662	330	1,422
		16,7	300	1,359	290	1,783	300	1,342
		15	270	1,266	270	1,152	270	1,332
		13,3	220	0,937	220	1,091	230	0,871
		11,7	200	0,781	200	0,862	200	0,891
		10	170	0,859	150	0,691	170	0,601
		8,3	120	0,75	110	0,601	110	0,471
		6,7	80	0,703	80	0,510	80	0,511
		5	30	0,515	30	0,370	30	0,350
33	4	27,5	590	1,922	620	1,242	640	2,814
		26,2	550	1,891	570	1,242	550	2,123
		25	560	1,781	540	1,272	540	1,793
		23,8	560	1,797	540	1,632	500	1,922
		22,5	510	2,062	470	1,402	520	1,503
		21,2	480	1,672	460	0,781	490	2,053
		20	410	1,859	410	0,951	430	1,962
		18,8	360	1,469	380	1,051	380	2,384
		17,5	380	1,578	370	1,311	370	1,812
		16,2	350	1,469	300	0,832	320	1,191
		15	300	1,343	300	0,811	300	1,022
		13,8	250	1,234	240	0,781	250	0,861
		12,5	230	1,266	240	0,821	230	1,312
		11,2	210	1,109	220	1,052	210	1,282
		10	190	1,0	170	0,781	150	0,741
		8,8	140	1,157	130	1,753	140	0,841
		7,5	90	0,672	90	0,441	90	0,481
		6,2	60	0,625	90	0,550	60	0,461
		5	30	0,609	30	0,381	30	0,420
		3,5	20	0,625	20	0,421	20	0,521
32	2	42,5	250	3,282	260	3,144	255	3,575
		40	255	2,781	245	3,616	265	3,506
		37,5	240	3,219	230	4,015	220	4,085
		36,5	240	3,11	225	2,845	220	4,116
		35	225	3,218	215	2,964	235	3,936

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		32,5	205	3,078	205	2,674	215	3,816
		30	190	2,547	190	2,904	180	3,665
		27,5	170	2,328	175	2,814	170	3,074
		25	155	2,453	155	2,454	145	2,153
		23	135	1,968	130	1,932	130	2,524
		20	110	1,469	110	1,372	110	1,702
		17,5	90	1,078	90	1,051	85	1,312
		15	80	1,312	80	0,821	80	1,252
		12,5	45	0,797	40	0,461	45	0,701
		10	25	0,375	25	0,310	25	0,291
		7,5	15	0,344	15	0,280	15	0,260
		5	10	0,344	15	0,230	10	0,250
32	3	28,5	220	2,891	240	1,502	225	2,634
		26,7	220	2,64	230	2,513	215	2,594
		25	220	2,422	220	1,813	205	2,293
		24,3	190	2,421	205	3,215	190	2,724
		23,3	170	2,219	185	1,792	185	2,274
		21,7	175	2,812	170	1,763	160	2,143
		20	155	1,734	145	1,482	150	1,962
		18,3	130	1,36	130	1,042	135	1,623
		16,7	110	1,093	105	0,831	105	1,151
		15,3	95	0,938	105	0,671	105	0,801
		13,3	60	0,703	70	0,551	60	0,611
		11,7	40	0,625	50	0,470	40	0,480
		10	25	0,5	40	0,400	25	0,381
		8,3	20	0,485	30	0,381	20	0,430
		6,7	10	0,453	10	0,340	10	0,321
		5	10	0,453	10	0,331	10	0,340
32	4	21,2	190	1,641	205	1,222	200	2,063
		20	180	1,718	180	1,412	180	2,304
		18,8	165	1,625	165	1,633	170	1,712
		18,2	150	1,5	160	1,161	160	0,811
		17,5	150	1,453	155	1,412	155	1,082
		16,2	125	1,219	120	0,762	115	0,951
		15	110	1,015	105	0,981	100	0,631
		13,8	90	1,469	85	0,661	90	1,182
		12,5	55	0,75	70	0,791	55	0,551
		11,5	50	0,875	55	0,521	50	0,641

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}
		10	25	0,625	40	0,471	25	0,430
		8,8	25	0,609	30	0,451	25	0,411
		7,5	15	0,594	25	0,370	15	0,401
		6,2	10	0,578	15	0,381	10	0,380
		5	10	0,578	15	0,390	10	0,401
21	2	22,5	260	1,438	270	0,831	260	1,002
		20	230	1,407	230	0,732	230	0,901
		19	230	1,406	230	0,701	230	0,911
		17,5	205	1,219	230	0,961	205	1,162
		16	200	0,969	200	0,651	200	0,561
		15	200	0,875	200	0,571	200	0,631
		13,5	160	0,797	180	0,631	190	0,931
		12,5	160	0,782	140	0,631	140	0,591
		11,5	125	0,625	125	0,551	140	0,360
		10	120	0,547	120	0,430	105	0,491
		7,5	85	0,437	85	0,391	85	0,341
21	3	15	200	0,89	200	0,550	200	0,722
		13,3	180	0,844	180	0,491	200	0,520
		12,7	200	0,766	200	0,671	180	0,491
		11,7	145	0,75	145	0,471	145	0,551
		10,7	140	0,641	120	0,501	140	0,370
		10	120	0,64	120	0,470	120	0,521
		9	120	0,625	100	0,391	120	0,481
		8,3	105	0,594	90	0,401	105	0,470
		7,7	105	0,579	90	0,390	105	0,451
		6,7	70	0,531	70	0,361	70	0,381
		5	55	0,469	55	0,350	55	0,310
21	4	11,2	160	0,828	160	0,400	160	0,511
		10	120	0,812	120	0,421	120	0,640
		9,5	120	0,765	120	0,411	120	0,621
		8,8	120	0,844	120	0,420	120	0,561
		8	105	0,734	105	0,371	105	0,531
		7,5	105	0,656	95	0,420	105	0,461
		6,8	70	0,672	70	0,420	70	0,460
		6,2	70	0,703	70	0,421	70	0,401
		5,8	60	0,594	70	0,390	60	0,390
		5	55	0,594	60	0,381	55	0,391
		3,8	10	0,671	10	0,371	10	0,340

à suivre...

TAB5.6 : *suite...*

			Euc		**Mah**		**Pro**	
n	m	L	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}	G_{tot}	T_{cpu}

Annexe3

Le tableau 5.7 présente les résultats numériques obtenus par l'approche tabou à voisinage étendu avec différentes solutions initiales générées par la méthodes à deux phases avec les trois métriques différentes. Cet algorithme a été testé sur une machine (*Pentium 4, 512 Mo*). Il a été comparé aux résultats obtenus par les méthodes proposées par (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005; Archetti *et al.*, 2007b). Les notations suivantes sont considérées :

- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigé par la solution
- L : la longueur maximale d'une tournée
- **TEuc** : les résultats des solutions obtenues avec l'approche tabou avec solution initiale à distance Euclidienne
- **TPro** : les résultats des solutions obtenues avec l'approche tabou avec solution initiale à distance de profilage
- **TMah** : les résultats des solutions obtenues avec l'approche tabou avec solution initiale à distance de Mahalanobis
- **Best** : les résultats des meilleurs solutions obtenues avec les algorithmes proposés par (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005), le meilleur compromis proposé par Archetti *et al.* (2007b) et par notre approche.
- G_{tot} : le gain total obtenu par la solution de l'instance correspondante
- T_{cpu} : le temps mis, en secondes, pour résoudre l'instance correspondante.
- *Ratio* : ratio par rapport à la meilleure solution entre CGW (Chao *et al.*, 1996b), TMH (Tang et Miller-Hooks, 2005), AHS (le meilleur compromis de Archetti *et al.* (2007b)).

TABLE 5.7 – *Résultats numériques de l'approche tabou à voisinage étendu avec trois solution initiales différentes*

			TEuc			**TPro**			**TMah**			**Best**
n	m	L	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*	G_{tot}
100	4	60	1229	273,3	0,959	1227	401,1	0,957	1281	386,5	0,999	1282
		57,5	1229	343,6	0,976	1237	374,1	0,983	1223	699,7	0,971	1259
		55	1191	269,2	0,987	1172	287,2	0,971	1166	215,5	0,966	1207
		52,5	1155	470,5	0,996	1121	378,6	0,966	1155	372,5	0,996	1160

à suivre...

TAB5.7 : *suite...*

			TEuc			**TPro**			**TMah**			**Best**
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$	G_{tot}
		50	1064	289,7	0,950	1092	464,2	0,975	1118	601,0	0,998	1120
		47,5	1018	312,4	0,969	1026	468,0	0,976	1030	553,9	0,980	1051
		45	956	479,4	0,979	961	608,5	0,984	968	214,5	0,991	977
		42,5	912	448,4	0,996	910	334,4	0,993	909	284,0	0,992	916
		40	856	320,4	0,974	865	544,9	0,984	875	471,8	0,995	879
		37,5	814	193,8	0,991	815	242,9	0,993	819	179,3	0,998	821
		35	728	240,6	0,995	726	237,6	0,992	732	174,2	1	732
		32,5	657	334,4	1	656	176,0	0,998	657	86,3	1	657
		30	571	108,8	1	565	77,8	0,989	564	58,4	0,988	571
		27,5	461	65,3	1	461	39,0	1	461	51,5	1	461
		25	324	30,2	1	324	30,5	1	324	23,0	1	324
		22,5	183	9,6	1	183	9,8	1	183	9,5	1	183
		20	38	1,2	1	38	1,2	1	38	0,9	1	38
	3	80	1254	489,3	0,966	1275	287,3	0,982	1278	349,5	0,985	1298
		76,7	1275	224,2	0,989	1264	382,8	0,981	1259	323,0	0,977	1289
		73,3	1256	395,7	0,987	1225	293,9	0,963	1241	523,8	0,976	1272
		70	1229	381,4	0,983	1235	381,9	0,988	1226	482,7	0,981	1250
		66,7	1207	463,8	0,991	1180	520,5	0,969	1185	217,4	0,973	1218
		63,3	1164	481,5	0,993	1136	576,6	0,969	1145	462,7	0,977	1172
		60	1075	440,8	0,961	1099	311,3	0,982	1085	329,4	0,970	1119
		56,7	1048	399,4	0,992	1012	494,1	0,958	1032	261,2	0,977	1056
		53,3	961	500,6	0,986	971	505,2	0,996	966	382,4	0,991	975
		50	901	492,2	0,980	906	339,9	0,986	912	388,5	0,992	919
		46,7	856	334,4	0,994	855	300,1	0,993	850	305,3	0,987	861
		43,3	799	133,0	0,991	799	134,5	0,991	792	185,9	0,983	806
		40	724	237,3	1	729*	209,9	1,007	729*	163,9	1,007	724
		36,7	646	86,0	0,989	652	101,6	0,998	647	126,5	0,991	653
		33,3	579	84,1	1	578	82,5	0,998	578	68,9	0,998	579
		30	468	79,3	1	467	76,7	0,998	467	50,9	0,998	468
		26,7	334	29,7	0,997	333	26,5	0,994	333	28,6	0,994	335
		23,3	193	8,0	1	193	7,8	1	193	7,8	1	193
		20	38	1,2	1	38	1,0	1	38	1,0	1	38
	2	120	1299	162,8	0,995	1304	191,4	0,998	1300	369,3	0,995	1306
		115	1291	253,5	0,993	1291	473,0	0,993	1280	237,3	0,985	1300
		110	1252	286,2	0,976	1274	416,2	0,993	1276	423,3	0,995	1283
		105	1222	785,7	0,966	1248	425,7	0,987	1259	314,9	0,995	1265
		100	1167	241,9	0,942	1228	442,2	0,991	1214	413,5	0,980	1239
		95	1184	458,8	0,974	1187	250,7	0,976	1200	374,1	0,987	1216
		90	1138	592,5	0,982	1148	268,7	0,991	1146	369,5	0,989	1159
		85	1096	535,4	0,968	1096	536,8	0,968	1095	336,8	0,967	1132
		80	1042	295,4	0,971	1071	548,3	0,998	1042	264,3	0,971	1073
		75	1016	304,7	0,997	1002	579,7	0,983	986	436,7	0,968	1019
		70	924	368,9	0,960	924	315,0	0,960	912	195,8	0,948	962
		65	910	231,1	0,993	862	182,8	0,941	862	185,5	0,941	916
		60	815	204,5	0,985	821	253,1	0,993	805	416,2	0,973	827
		55	753*	182,3	1,004	750	185,8	1	737	159,5	0,983	750
		50	686*	230,9	1,003	679	141,0	0,993	687*	110,3	1,004	684
		45	602	188,1	0,974	602	93,8	0,974	601	218,6	0,972	618
		40	529	77,3	0,996	521	61,4	0,981	524	68,3	0,987	531
		35	448	39,0	0,991	452	45,2	1	452	56,8	1	452
		30	341	24,1	1	341	22,2	1	341	24,0	1	341
		25	206	9,1	1	206	8,4	1	206	9,1	1	206
102	4	100	1077	453,0	1	1066	467,0	0,990	1075	177,0	0,998	1077
		95	1022*	270,9	1,001	1022*	604,4	1,001	1022*	154,9	1,001	1021

à suivre...

TAB5.7 : *suite...*

			TEuc			**TPro**			**TMah**			**Best**
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$	G_{tot}
		90	950	204,0	0,979	970	202,0	1	960	269,3	0,990	970
		85	901	131,8	0,996	900	348,7	0,994	900	122,3	0,994	905
		80	825	115,8	0,975	818	447,2	0,967	838	116,6	0,991	846
		75	778*	98,8	1,003	772	255,5	0,995	775	99,2	0,999	776
		70	702	93,1	0,967	713	203,1	0,982	730*	169,5	1,006	726
		65	639	136,2	0,989	640	141,7	0,991	638	78,3	0,988	646
		60	578	60,6	0,983	583	94,6	0,991	577	104,3	0,981	588
		55	516	101,8	0,996	514	58,6	0,992	514	61,4	0,992	518
		50	462	69,3	1	462	56,3	1	462	55,0	1	462
		45	357	23,8	0,975	364	38,0	0,995	366	61,9	1	366
		40	283	22,7	0,993	285	17,4	1	283	24,7	0,993	285
		35	217	10,7	1	217	16,6	1	217	18,1	1	217
		30	164	6,9	1	164	7,2	1	164	6,5	1	164
		25	123	4,2	1	123	4,6	1	123	4,0	1	123
		20	79	1,3	1	79	1,3	1	79	0,9	1	79
		15	46	1,3	1	46	1,4	1	46	0,9	1	46
		10	30	1,3	1	30	1,3	1	30	0,8	1	30
	3	133,3	1113*	210,8	1,001	1109	392,3	0,997	1110	214,1	0,998	1112
		126,7	1065	274,3	0,987	1070	489,6	0,992	1079	543,1	1	1079
		120	1019	182,8	0,995	1018	309,1	0,994	1019	187,5	0,995	1024
		113,3	971	159,3	0,993	971	176,1	0,993	965	161,0	0,987	978
		106,7	920	146,5	0,994	878	255,0	0,948	919	149,8	0,992	926
		100	873	196,0	0,999	842	330,6	0,963	874	133,4	1	874
		93,3	806	178,5	0,990	802	287,1	0,985	792	107,9	0,973	814
		86,7	745	100,8	0,985	746	106,3	0,987	755	200,2	0,999	756
		80	681	89,0	1	669	101,7	0,982	681	83,3	1	681
		73,3	623	116,2	0,986	618	100,4	0,978	628	73,1	0,994	632
		66,7	553	76,8	0,982	549	107,3	0,975	557	59,9	0,989	563
		60	487	54,9	1	478	57,4	0,982	485	44,2	0,996	487
		53,3	424	39,5	0,998	422	31,4	0,993	425	53,8	1	425
		46,7	344	19,5	1	344	22,9	1	344	17,5	1	344
		40	247	14,5	1	247	14,5	1	244	13,8	0,988	247
		33,3	175	7,4	1	175	6,8	1	175	6,6	1	175
		26,7	117	3,5	1	117	3,9	1	117	3,1	1	117
		20	79	1,2	1	79	1,2	1	79	0,8	1	79
		13,3	46	1,1	1	46	1,2	1	46	0,8	1	46
	2	200	1157	258,0	0,986	1154	862,8	0,984	1159	259,0	0,988	1173
		190	1109	459,8	0,976	1107	357,5	0,974	1121	234,9	0,987	1136
		180	1039	406,3	0,950	1079	407,2	0,986	1080	218,6	0,987	1094
		170	1012	398,4	0,975	1008	558,9	0,971	1015	208,8	0,978	1038
		160	962	521,8	0,960	941	188,3	0,939	976	191,7	0,974	1002
		150	935	478,4	0,989	914	231,2	0,967	933	408,3	0,987	945
		140	878	342,5	0,994	872	199,5	0,988	863	151,0	0,977	883
		130	827*	129,8	1,004	827*	137,5	1,004	827*	129,5	1,004	824
		120	761	115,3	0,992	761	117,0	0,992	758	114,0	0,988	767
		110	705*	93,1	1,001	705*	94,3	1,001	705*	96,0	1,001	704
		100	646*	79,1	1,005	635	117,2	0,988	642	74,1	0,998	643
		90	579*	60,4	1,002	579*	62,4	1,002	579*	61,9	1,002	578
		80	517	46,8	0,992	517	46,5	0,992	517	45,7	0,992	521
		70	453	32,5	0,987	459	32,7	1	459	33,3	1	459
		60	387	23,6	1	387	22,3	1	387	22,8	1	387
		50	290	13,0	1	290	12,4	1	290	12,9	1	290
		40	190	6,6	1	190	6,2	1	190	6,3	1	190
		30	101	2,0	1	101	2,0	1	101	1,8	1	101

à suivre...

TAB5.7 : *suite...*

			TEuc			**TPro**			**TMah**			**Best**
n	m	L	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*	G_{tot}
		20	64	1,0	1	64	1,0	1	64	0,7	1	64
		10	30	1,0	1	30	1,0	1	30	0,7	1	30
66	4	32,5	1620	49,9	1	1620	31,9	1	1560	32,1	0,963	1620
		31,2	1495	30,6	0,984	1510	36,3	0,993	1520	29,4	1	1520
		30	1390	39,5	0,959	1445	43,0	0,997	1445	37,8	0,997	1450
		28,8	1325	32,0	0,953	1380	35,6	0,993	1370	32,0	0,986	1390
		27,5	1320	41,3	1	1320	34,7	1	1320	32,1	1	1320
		26,2	1300	33,1	1	1300	30,5	1	1300	30,5	1	1300
		25	1115	36,0	0,961	1160	60,8	1	1160	33,0	1	1160
		23,8	1000	28,5	0,971	1025	41,1	0,995	1020	28,8	0,990	1030
		22,5	960	26,1	1	960	29,6	1	960	29,8	1	960
		21,2	860	23,3	1	860	38,3	1	860	38,5	1	860
		20	765	33,5	1	765	30,5	1	760	21,6	0,993	765
		18,8	690	20,1	1	690	19,4	1	690	31,4	1	690
		17,5	620	16,1	1	620	14,8	1	620	13,8	1	620
		16,2	550	16,7	0,991	550	14,5	0,991	555	19,3	1	555
		15	430	13,0	1	430	11,8	1	430	10,5	1	430
		13,8	340	10,2	1	340	9,4	1	340	9,4	1	340
		12,5	340	7,5	1	340	7,4	1	340	7,6	1	340
		11,2	240	5,1	1	240	5,2	1	240	5,4	1	240
		10	140	4,6	1	140	4,0	1	140	3,9	1	140
		8,8	140	2,8	1	140	2,8	1	140	3,5	1	140
		7,5	80	2,2	1	80	2,1	1	80	2,2	1	80
		6,2	20	0,8	1	20	0,7	1	20	0,6	1	20
		5	20	0,7	1	20	0,7	1	20	0,6	1	20
		3,8	20	0,7	1	20	0,7	1	20	0,6	1	20
	3	43,3	1620	30,3	0,991	1620	27,0	0,991	1635	26,2	1	1635
		41,7	1590	39,8	0,997	1575	38,9	0,987	1570	37,4	0,984	1595
		40	1535	45,2	0,987	1530	60,9	0,984	1515	37,6	0,974	1555
		38,3	1475	67,2	0,993	1465	37,9	0,987	1475	38,5	0,993	1485
		36,7	1425	38,9	1	1425	41,7	1	1425	56,2	1	1425
		35	1345	59,1	1	1340	50,1	0,996	1345	37,9	1	1345
		33,3	1260	88,8	1	1260	40,9	1	1250	36,4	0,992	1260
		31,7	1190	39,7	1	1190	35,6	1	1190	49,4	1	1190
		30	1120	32,7	0,996	1120	30,2	0,996	1125	33,9	1	1125
		28,3	1065	28,7	0,995	1070	30,4	1	1070	28,5	1	1070
		26,7	990	30,5	1	990	28,3	1	990	26,3	1	990
		25	870	24,5	1	870	33,9	1	870	25,5	1	870
		23,3	755	22,6	1	755	20,9	1	755	22,6	1	755
		21,7	650	19,5	1	650	16,5	1	650	16,6	1	650
		20	595	15,4	1	595	21,0	1	595	24,8	1	595
		18,3	495	11,5	1	495	12,4	1	495	16,9	1	495
		16,7	470	9,2	1	470	10,1	1	470	9,6	1	470
		15	335	7,5	1	335	7,4	1	335	8,3	1	335
		13,3	260	5,9	1	260	5,6	1	260	5,0	1	260
		11,7	185	4,4	1	185	4,4	1	185	4,4	1	185
		10	110	3,4	1	110	3,1	1	110	2,9	1	110
		8,3	95	2,3	0,864	95	2,3	0,864	95	2,5	0,864	110
		6,7	60	1,3	1	60	1,3	1	60	1,3	1	60
		5	20	0,6	1	20	0,6	1	20	0,5	1	20
		3,3	15	0,7	1	15	0,6	1	15	0,5	1	15
	2	65	1660	25,6	0,988	1660	21,0	0,988	1680	21,0	1	1680
		62,5	1645*	29,5	1,006	1635	26,3	1	1640*	27,4	1,003	1635
		60	1590	36,3	0,988	1590	34,1	0,988	1610	30,7	1	1610

à suivre...

Tab5.7 : *suite...*

			TEuc			**TPro**			**TMah**			**Best**
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$	G_{tot}
		57,5	1550	74,5	0,994	1560	58,9	1	1565*	34,4	1,003	1560
		55	1485	48,9	0,990	1485	61,5	0,990	1500	37,6	1	1500
		52,5	1425	44,9	0,976	1450	40,6	0,993	1460	39,0	1	1460
		50	1385	53,1	0,989	1400	45,0	1	1380	40,8	0,986	1400
		47,5	1325	64,5	0,989	1315	61,3	0,981	1325	42,8	0,989	1340
		45	1260	64,0	1	1260	49,7	1	1255	47,2	0,996	1260
		42,5	1195	38,1	1	1195	43,4	1	1195	37,4	1	1195
		40	1150	42,5	1	1150	40,3	1	1150	51,9	1	1150
		37,5	1020	45,5	1	990	42,6	0,971	1010	32,2	0,990	1020
		35	925	37,9	1	925	28,4	1	925	27,0	1	925
		32,5	860	23,4	1	840	23,4	0,977	860	29,7	1	860
		30	800	21,5	1	800	22,9	1	800	30,2	1	800
		27,5	670	19,9	1	670	19,9	1	660	18,0	0,985	670
		25	580	13,0	1	580	17,2	1	580	16,0	1	580
		22,5	480	10,3	1	480	11,5	1	480	11,4	1	480
		20	395	7,7	0,963	410	9,5	1	410	9,4	1	410
		17,5	315	5,2	0,984	320	6,0	1	315	5,2	0,984	320
		15	240	4,4	1	240	4,8	1	240	3,9	1	240
		12,5	175	2,7	0,972	180	3,2	1	180	2,3	1	180
		10	80	2,0	1	80	2,0	1	80	1,5	1	80
		7,5	50	1,0	1	50	1,0	1	50	0,9	1	50
		5	20	0,5	1	20	0,5	1	20	0,4	1	20
64	4	20	1068	32,5	1	1068	31,7	1	1068	36,8	1	1068
		18,8	912	34,9	1	912	34,8	1	912	30,5	1	912
		17,5	696	26,0	1	696	23,0	1	696	26,1	1	696
		16,2	528	15,9	0,967	528	17,0	0,967	528	20,1	0,967	546
		15	366	10,2	1	366	10,0	1	366	10,8	1	366
	3	26,7	1158	32,7	0,990	1158	32,4	0,990	1164	64,0	0,995	1170
		25	1080	31,4	1	1080	31,2	1	1080	39,6	1	1080
		23,3	1002	32,7	1	1002	32,4	1	1002	36,0	1	1002
		21,7	894	27,2	1	888	30,4	0,993	894	28,5	1	894
		20	828	23,5	1	828	25,0	1	828	26,9	1	828
		18,3	642	21,9	1	642	20,7	1	642	23,7	1	642
		16,7	444	12,0	1	444	16,9	1	444	16,9	1	444
		15	282	8,3	1	282	8,2	1	282	8,2	1	282
	2	40	1224	38,7	0,971	1224	38,4	0,971	1224	42,3	0,971	1260
		37,5	1152	35,9	0,970	1152	35,6	0,970	1170	41,9	0,985	1188
		35	1110	42,2	0,995	1110	42,0	0,995	1110	53,2	0,995	1116
		32,5	1032	34,5	1	1032	34,3	1	1032	36,1	1	1032
		30	948	33,9	1	948	33,5	1	948	54,7	1	948
		27,5	888	28,6	1	888	28,5	1	888	30,5	1	888
		25	780	23,3	1	780	23,3	1	780	24,9	1	780
		22,5	660	19,4	1	660	21,3	1	660	19,5	1	660
		20	588	13,9	1	588	16,2	1	534	14,5	0,908	588
		17,5	360	11,7	1	360	12,3	1	360	12,8	1	360
33	4	27,5	670	2,9	1	670	3,2	1	670	2,5	1	670
		26,2	670	3,0	1	670	3,3	1	610	2,4	0,910	670
		25	600	3,1	1	570	3,9	0,950	600	2,9	1	600
		23,8	560	3,0	1	560	3,1	1	560	2,9	1	560
		22,5	560	3,3	1	560	2,8	1	560	3,1	1	560
		21,2	500	2,9	1	500	3,4	1	500	3,3	1	500
		20	440	3,0	1	440	3,0	1	440	2,2	1	440
		18,8	390	2,5	1	390	3,3	1	390	2,1	1	390
		17,5	380	2,6	1	380	3,0	1	380	2,3	1	380

à suivre...

TAB5.7 : *suite...*

			TEuc			**TPro**			**TMah**			**Best**
n	m	L	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*	G_{tot}
		16,2	350	2,3	1	350	2,6	1	350	1,8	1	350
		15	310	2,2	1	310	2,3	1	310	1,8	1	310
		13,8	270	2,1	1	270	2,0	1	270	1,6	1	270
		12,5	240	1,9	1	240	1,4	1	240	1,5	1	240
		11,2	220	1,7	1	220	1,4	1	220	1,6	1	220
		10	190	1,4	1	190	1,2	1	190	1,2	1	190
		8,8	140	1,3	1	140	1,1	1	140	0,9	1	140
		7,5	100	0,8	1	100	0,7	1	100	0,7	1	100
		6,2	90	0,7	1	90	0,6	1	90	0,8	1	90
		5	30	0,6	1	30	0,5	1	30	0,6	1	30
		3,5	20	0,6	1	20	0,5	1	20	0,5	1	20
	3	36,7	750	3,5	0,987	750	3,8	0,987	760	3,8	1	760
		35	710	3,1	0,986	710	3,5	0,986	710	2,9	0,986	720
		33,3	710	3,0	1	710	3,4	1	710	3,9	1	710
		31,7	680	4,2	1	680	5,2	1	630	3,6	0,926	680
		30	610	3,8	0,953	600	3,6	0,938	640	3,7	1	640
		28,3	590	3,2	1	590	3,6	1	580	3,8	0,983	590
		26,7	560	3,8	0,982	570	4,2	1	570	4,5	1	570
		25	510	4,5	0,981	510	4,0	0,981	520	3,7	1	520
		23,3	480	4,3	1	480	4,3	1	480	3,4	1	480
		21,7	440	3,7	1	440	3,3	1	430	4,6	0,977	440
		20	380	2,8	1	380	3,0	1	380	2,6	1	380
		18,3	330	2,4	1	330	2,6	1	330	2,5	1	330
		16,7	300	2,1	1	300	2,5	1	300	2,5	1	300
		15	270	2,0	1	270	1,6	1	270	1,9	1	270
		13,3	230	1,5	1	230	1,5	1	230	1,6	1	230
		11,7	200	1,3	1	200	1,2	1	200	1,4	1	200
		10	170	1,1	1	170	1,0	1	170	1,0	1	170
		8,3	120	0,9	1	120	0,7	1	120	0,8	1	120
		6,7	90	0,7	1	90	0,7	1	90	0,6	1	90
		5	30	0,5	1	30	0,4	1	30	0,4	1	30
	2	55	800	4,2	1	800	4,1	1	800	3,3	1	800
		52,5	800	4,4	1	800	4,3	1	800	3,4	1	800
		50	750	3,9	0,949	750	3,8	0,949	790	4,8	1	790
		47,5	750	4,2	0,987	750	4,0	0,987	760	5,2	1	760
		45	720	4,6	1	720	4,4	1	720	3,6	1	720
		42,5	690	4,5	1	690	4,4	1	670	4,7	0,971	690
		40	660	5,0	1	660	4,8	1	660	6,0	1	660
		37,5	620	4,9	1	620	4,7	1	610	4,4	0,984	620
		35	590	4,8	1	570	4,8	0,966	570	4,0	0,966	590
		32,5	550	5,0	1	550	4,6	1	550	4,8	1	550
		30	490	4,2	0,961	490	3,9	0,961	510	4,6	1	510
		27,5	460	3,9	1	460	3,7	1	450	4,3	0,978	460
		25	400	3,1	0,976	410	4,2	1	410	3,6	1	410
		22,5	360	2,8	1	360	2,9	1	360	2,9	1	360
		20	300	2,3	1	300	2,2	1	300	2,7	1	300
		17,5	260	2,3	1	260	2,4	1	260	2,1	1	260
		15	220	1,7	1	220	1,8	1	220	1,5	1	220
		12,5	180	1,4	1	180	1,0	1	180	1,0	1	180
		10	150	0,9	1	150	0,7	1	150	0,7	1	150
		7,5	90	0,5	1	90	0,4	1	90	0,5	1	90
32	4	21,2	210	3,2	1	210	3,5	1	205	2,3	0,976	210
		20	190	2,7	1	190	3,3	1	190	2,3	1	190
		18,8	175	2,5	1	175	2,6	1	175	2,6	1	175

à suivre...

Tab5.7 : *suite...*

			TEuc			**TPro**			**TMah**			**Best**
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$	G_{tot}
		18,2	165	2,3	1	165	2,0	1	165	2,3	1	165
		17,5	155	2,1	1	155	2,0	1	155	2,3	1	155
		16,2	130	1,9	1	130	1,7	1	130	1,5	1	130
		15	120	1,5	1	120	1,3	1	120	1,5	1	120
		13,8	100	1,8	1	100	1,6	1	100	1,1	1	100
		12,5	75	1,0	1	75	0,9	1	75	1,1	1	75
		11,5	60	1,1	1	60	1,2	1	60	0,8	1	60
		10	45	0,8	1	45	0,9	1	45	0,7	1	45
		8,8	35	0,7	1	35	0,6	1	35	0,6	1	35
		7,5	25	0,6	1	25	0,5	1	25	0,5	1	25
		6,2	15	0,5	1	15	0,5	1	15	0,4	1	15
		5	15	0,5	1	15	0,5	1	15	0,4	1	15
	3	28,5	255*	4,1	1,020	255*	3,9	1,020	255*	2,6	1,020	250
		26,7	230	3,7	1	230	3,9	1	230	3,4	1	230
		25	220	3,5	1	210	3,2	0,955	220	2,8	1	220
		24,3	205	3,5	0,953	205	3,6	0,953	205	4,1	0,953	215
		23,3	190	3,6	1	190	3,5	1	190	3,0	1	190
		21,7	175	3,8	1	170	3,1	0,971	175	2,6	1	175
		20	155	2,5	1	155	2,9	1	155	2,2	1	155
		18,3	135	2,0	1	135	2,3	1	135	2,0	1	135
		16,7	115	1,6	1	115	1,8	1	115	1,5	1	115
		15,3	100	1,3	0,952	105	1,3	1	105	1,1	1	105
		13,3	70	0,9	0,933	70	0,9	0,933	70	0,8	0,933	75
		11,7	50	0,8	1	50	0,7	1	50	0,6	1	50
		10	40	0,6	1	40	0,6	1	40	0,5	1	40
		8,3	30	0,5	1	30	0,5	1	30	0,5	1	30
		6,7	15	0,4	1	15	0,4	1	15	0,4	1	15
		5	15	0,4	1	15	0,4	1	15	0,4	1	15
	2	42,5	275	4,5	0,982	275	4,5	0,982	270	4,7	0,964	280
		40	265	3,7	1	265	4,3	1	260	4,4	0,981	265
		37,5	250	4,5	1	245	5,2	0,980	250	4,9	1	250
		36,5	240	4,1	1	240	5,3	1	240	3,9	1	240
		35	235	4,2	1	235	4,7	1	230	4,5	0,979	235
		32,5	215	4,1	1	215	4,7	1	205	3,6	0,953	215
		30	190	3,5	0,974	195	4,9	1	190	3,7	0,974	195
		27,5	175	3,4	1	175	3,8	1	175	3,4	1	175
		25	155	3,1	1	155	3,0	1	155	3,2	1	155
		23	135	2,6	1	130	3,1	0,963	130	2,4	0,963	135
		20	110	1,9	1	110	2,1	1	110	1,8	1	110
		17,5	90	1,4	1	90	1,6	1	90	1,3	1	90
		15	80	1,5	1	80	1,5	1	80	1,0	1	80
		12,5	45	0,8	1	45	0,9	1	45	0,6	1	45
		10	30	0,4	1	30	0,5	1	30	0,4	1	30
		7,5	20	0,3	1	20	0,4	1	20	0,3	1	20
		5	15	0,3	1	15	0,3	1	15	0,3	1	15
21	4	11,2	180	0,9	1	180	0,7	1	180	0,6	1	180
		10	120	0,8	1	120	0,8	1	120	0,5	1	120
		9,5	120	0,8	1	120	0,8	1	120	0,5	1	120
		8,8	120	0,7	1	120	0,7	1	120	0,5	1	120
		8	105	0,7	1	105	0,7	1	105	0,5	1	105
		7,5	105	0,6	1	105	0,6	1	105	0,5	1	105
		6,8	70	0,6	1	70	0,6	1	70	0,5	1	70
		6,2	70	0,5	1	70	0,5	1	70	0,5	1	70
		5,8	70	0,5	1	70	0,5	1	70	0,5	1	70

à suivre...

TAB5.7 : *suite...*

			TEuc			**TPro**			**TMah**			**Best**
n	m	L	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*	G_{tot}
		5	70	0,5	1	70	0,5	1	70	0,5	1	70
		3,8	10	0,5	1	10	0,4	1	10	0,4	1	10
	3	15	200	1,0	1	200	1,0	1	200	0,7	1	200
		13,3	200	0,9	1	200	0,8	1	200	0,7	1	200
		12,7	200	0,8	1	200	0,8	1	200	0,9	1	200
		11,7	165	0,8	1	165	0,8	1	165	0,7	1	165
		10,7	145	0,9	1	145	0,5	1	145	0,6	1	145
		10	120	0,8	1	120	0,7	1	120	0,6	1	120
		9	120	0,8	1	120	0,6	1	120	0,5	1	120
		8,3	105	0,7	1	105	0,6	1	105	0,5	1	105
		7,7	105	0,7	1	105	0,7	1	105	0,5	1	105
		6,7	70	0,6	1	70	0,6	1	70	0,4	1	70
		5	70	0,5	1	70	0,4	1	70	0,4	1	70
	2	22,5	275	1,7	1	275	1,5	1	275	1,0	1	275
		20	260	1,6	1	260	1,1	1	260	0,9	1	260
		19	230	1,6	1	230	1,1	1	230	0,9	1	230
		17,5	230	1,4	1	230	1,3	1	230	1,1	1	230
		16	200	1,2	1	200	0,8	1	200	0,8	1	200
		15	200	1,0	1	200	0,8	1	200	0,8	1	200
		13,5	180	0,9	0,947	190	1,1	1	190	0,7	1	190
		12,5	160	0,8	1	160	0,7	1	160	0,8	1	160
		11,5	140	0,7	1	140	0,5	1	140	0,6	1	140
		10	120	0,6	1	120	0,6	1	120	0,5	1	120
		7,5	90	0,5	1	90	0,4	1	90	0,4	1	90

Annexe4

Le tableau 5.8 présente les résultats numériques obtenus par l'approche tabou avec la structure de voisinage réduite avec la solution initiale générée par la méthode à deux phases avec la distance de Mahalanobis. Cet algorithme a été testé sur une machine (*Pentium 4, 512 Mo*). Il a été comparé aux résultats obtenus par les méthodes de (Chao *et al.*, 1996b; Tang et Miller-Hooks, 2005; Archetti *et al.*, 2007b) et la méthode de recherche tabou avec voisinage étendu décrite dans le chapitre 2. Les notations suivantes sont considérées :

- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigé par la solution
- L : la longueur maximale d'une tournée
- **CGW** : les résultats obtenus avec l'approche de (Chao *et al.*, 1996b)
- **TMH** : les résultats obtenus avec l'approche de (Tang et Miller-Hooks, 2005)
- **AHS** : les résultats obtenus avec l'approche de (Archetti *et al.*, 2007b)
- **TE** : les résultats obtenus avec l'approche tabou à voisinage étendu
- **TR** : les résultats obtenus avec l'approche tabou à voisinage réduit
- G_{tot} : le gain total obtenu par la solution de l'instance correspondante
- T_{cpu} : le temps mis, en secondes, pour résoudre l'instance correspondante.
- *Ratio* : ratio par rapport à la meilleure solution entre CGW (Chao *et al.*, 1996b), TMH (Tang et Miller-Hooks, 2005), AHS (le meilleur compromis de Archetti *et al.* (2007b)).

TABLE 5.8 – *Résultats numériques de l'approche tabou avec voisinage réduit*

			CGW	**TMH**	**AHS**	**TE**			**TR**		
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*
100	4	60	1253	1255	1282	1281	386,5	0,999	1281	267,2	0,999
		57,5	1230	1243	1259	1223	699,7	0,971	1253	127,5	0,995
		55	1155	1165	1207	1166	215,5	0,966	1193	263,0	0,988
		52,5	1084	1124	1160	1155	372,5	0,996	1153	143,1	0,994
		50	996	1056	1120	1118	601,0	0,998	1115	139,0	0,996
		47,5	995	1014	1051	1030	553,9	0,980	1020	109,4	0,971
		45	932	977	968	968	214,5	0,991	971	261,3	0,994
		42,5	895	910	916	909	284,0	0,992	916	153,0	1
		40	847	875	879	875	471,8	0,995	876	68,4	0,997

à suivre...

TAB5.8 : *suite...*

			CGW	**TMH**	**AHS**		**TE**			**TR**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		37,5	770	819	821	819	179,3	0,998	815	56,7	0,993
		35	697	732	732	732	174,2	1	732	43,8	1
		32,5	641	627	657	657	86,3	1	654	32,5	0,995
		30	545	554	571	564	58,4	0,988	571	25,7	1
		27,5	460	453	461	461	51,5	1	461	20,4	1
		25	304	315	324	324	23,0	1	324	15,9	1
		22,5	182	182	183	183	9,5	1	183	2,9	1
		20	38	38	38	38	0,9	1	38	0,7	1
	3	80	1285	1288	1298	1278	349,5	0,985	1294	139,0	0,997
		76,7	1239	1282	1289	1259	323,0	0,977	1264	116,6	0,981
		73,3	1225	1265	1272	1241	523,8	0,976	1252	209,3	0,984
		70	1222	1249	1250	1226	482,7	0,981	1237	102,0	0,990
		66,7	1115	1218	1208	1185	217,4	0,973	1192	114,9	0,979
		63,3	1078	1151	1172	1145	462,7	0,977	1148	198,8	0,980
		60	1018	1119	1111	1085	329,4	0,970	1096	138,8	0,979
		56,7	956	1005	1056	1032	261,2	0,977	1022	126,3	0,968
		53,3	946	951	975	966	382,4	0,991	970	143,3	0,995
		50	889	906	919	912	388,5	0,992	918	168,1	0,999
		46,7	829	860	861	850	305,3	0,987	859	127,2	0,998
		43,3	798	785	806	792	185,9	0,983	804	98,1	0,998
		40	717	709	724	729	163,9	1,007	723	74,6	0,999
		36,7	623	646	653	647	126,5	0,991	653	103,7	1
		33,3	552	579	579	578	68,9	0,998	576	35,9	0,995
		30	432	465	468	467	50,9	0,998	468	47,1	1
		26,7	333	333	335	333	28,6	0,994	334	25,0	0,997
		23,3	191	192	193	193	7,8	1	182	5,1	0,943
		20	38	38	38	38	1,0	1	38	0,7	1
	2	120	1299	1306	1306	1300	369,3	0,995	1305	237,9	0,999
		115	1286	1294	1300	1280	237,3	0,985	1299	313,5	0,999
		110	1242	1277	1283	1276	423,3	0,995	1283	195,6	1
		105	1199	1255	1265	1259	314,9	0,995	1257	219,3	0,994
		100	1199	1208	1239	1214	413,5	0,980	1231	296,5	0,994
		95	1147	1175	1216	1200	374,1	0,987	1195	282,2	0,983
		90	1112	1150	1159	1146	369,5	0,989	1173*	302,1	1,012
		85	1039	1089	1132	1095	336,8	0,967	1092	268,5	0,965
		80	1003	1022	1073	1042	264,3	0,971	1062	379,3	0,990
		75	932	963	1019	986	436,7	0,968	1012	320,4	0,993
		70	899	915	962	912	195,8	0,948	918	180,8	0,954
		65	858	914	916	862	185,5	0,941	899	303,1	0,981
		60	807	827	827	805	416,2	0,973	819	259,4	0,990
		55	737	749	750	737	159,5	0,983	753*	284,6	1,004
		50	669	666	684	687	110,3	1,004	687*	188,6	1,004
		45	580	593	618	601	218,6	0,972	608	116,5	0,984
		40	531	517	531	524	68,3	0,987	531	126,9	1
		35	440	438	452	452	56,8	1	452	60,0	1
		30	341	341	341	341	24,0	1	341	27,5	1
		25	194	202	206	206	9,1	1	195	11,7	0,947
102	4	100	1066	1067	1077	1075	177,0	0,998	1062	46,7	0,986
		95	990	1019	1021	1022	154,9	1,001	1022*	49,5	1,001
		90	886	966	970	960	269,3	0,990	956	44,5	0,986
		85	882	905	899	900	122,3	0,994	901	54,0	0,996
		80	801	832	846	838	116,6	0,991	832	35,0	0,983
		75	728	776	770	775	99,2	0,999	776	36,0	1
		70	683	726	715	730	169,5	1,006	724	37,4	0,997

à suivre...

Tab5.8 : *suite...*

			CGW	**TMH**	**AHS**		**TE**			**TR**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		65	610	643	646	638	78,3	0,988	646	40,9	1
		60	562	576	588	577	104,3	0,981	576	22,7	0,980
		55	516	503	518	514	61,4	0,992	518	23,2	1
		50	462	462	462	462	55,0	1	457	24,2	0,989
		45	338	359	366	366	61,9	1	364	17,7	0,995
		40	283	285	285	283	24,7	0,993	283	7,0	0,993
		35	209	217	217	217	18,1	1	209	3,7	0,963
		30	156	164	164	164	6,5	1	153	2,5	0,933
		25	123	123	123	123	4,0	1	109	1,7	0,886
		20	79	79	79	79	0,9	1	79	1,3	1
		15	46	46	46	46	0,9	1	30	0,6	0,652
		10	30	30	30	30	0,8	1	30	0,6	1
	3	133,3	1095	1098	1112	1110	214,1	0,998	1118*	172,3	1,005
		126,7	1064	1061	1079	1079	543,1	1	1073	87,6	0,994
		120	1008	1011	1024	1019	187,5	0,995	1020	133,8	0,996
		113,3	943	966	978	965	161,0	0,987	976	64,4	0,998
		106,7	919	922	926	919	149,8	0,992	918	62,8	0,991
		100	848	874	871	874	133,4	1	859	56,8	0,983
		93,3	813	789	814	792	107,9	0,973	805	83,0	0,989
		86,7	754	756	745	755	200,2	0,999	737	85,3	0,975
		80	676	681	681	681	83,3	1	677	43,3	0,994
		73,3	602	632	632	628	73,1	0,994	632	66,6	1
		66,7	539	563	562	557	59,9	0,989	557	33,5	0,989
		60	466	481	487	485	44,2	0,996	481	26,9	0,988
		53,3	419	416	425	425	53,8	1	425	42,6	1
		46,7	338	344	344	344	17,5	1	344	22,6	1
		40	235	247	247	244	13,8	0,988	247	16,5	1
		33,3	163	175	175	175	6,6	1	175	5,1	1
		26,7	117	117	117	117	3,1	1	105	2,3	0,897
		20	79	79	79	79	0,8	1	48	1,1	0,608
		13,3	46	46	46	46	0,8	1	32	0,5	0,696
	2	200	1173	1165	1168	1159	259,0	0,988	1166	344,8	0,994
		190	1127	1116	1136	1121	234,9	0,987	1124	330,8	0,989
		180	1082	1067	1094	1080	218,6	0,987	1080	143,0	0,987
		170	1031	1017	1038	1015	208,8	0,978	1041*	220,6	1,003
		160	987	987	1002	976	191,7	0,974	976	168,1	0,974
		150	934	914	945	933	408,3	0,987	930	201,8	0,984
		140	864	864	883	863	151,0	0,977	875	221,8	0,991
		130	811	817	824	827	129,5	1,004	822	139,7	0,998
		120	758	767	759	758	114,0	0,988	767	202,7	1
		110	693	702	704	705	96,0	1,001	705*	88,2	1,001
		100	633	638	643	642	74,1	0,998	644*	75,2	1,002
		90	576	578	575	579	61,9	1,002	579*	89,5	1,002
		80	517	521	521	517	45,7	0,992	517	76,5	0,992
		70	453	459	459	459	33,3	1	457	56,8	0,996
		60	379	382	387	387	22,8	1	387	39,2	1
		50	275	290	289	290	12,9	1	290	21,4	1
		40	190	190	190	190	6,3	1	190	8,6	1
		30	101	101	101	101	1,8	1	101	2,9	1
		20	64	64	64	64	0,7	1	64	1,2	1
		10	30	30	30	30	0,7	1	30	0,5	1
66	4	32,5	1545	1575	1620	1560	32,1	0,963	1585	14,5	0,978
		31,2	1490	1520	1520	1520	29,4	1	1520	10,5	1
		30	1420	1410	1450	1445	37,8	0,997	1440	10,4	0,993

à suivre...

TAB5.8 : *suite...*

			CGW	**TMH**	**AHS**		**TE**			**TR**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*
		28,8	1380	1380	1390	1370	32,0	0,986	1380	12,2	0,993
		27,5	1310	1310	1320	1320	32,1	1	1320	10,9	1
		26,2	1260	1275	1300	1300	30,5	1	1300	10,0	1
		25	1160	1100	1160	1160	33,0	1	1160	11,8	1
		23,8	1020	1000	1030	1020	28,8	0,990	1010	9,8	0,981
		22,5	950	960	960	960	29,8	1	960	11,6	1
		21,2	860	860	860	860	38,5	1	860	15,1	1
		20	750	760	765	760	21,6	0,993	735	8,6	0,961
		18,8	675	680	690	690	31,4	1	670	9,2	0,971
		17,5	620	620	620	620	13,8	1	620	7,4	1
		16,2	495	555	555	555	19,3	1	515	5,7	0,928
		15	430	430	430	430	10,5	1	415	8,2	0,965
		13,8	340	340	340	340	9,4	1	340	3,6	1
		12,5	340	340	340	340	7,6	1	340	3,3	1
		11,2	240	240	240	240	5,4	1	235	2,3	0,979
		10	140	140	140	140	3,9	1	140	1,9	1
		8,8	140	140	140	140	3,5	1	140	2,3	1
		7,5	80	80	80	80	2,2	1	80	1,3	1
		6,2	20	20	20	20	0,6	1	20	0,5	1
		5	20	20	20	20	0,6	1	20	0,5	1
		3,8	20	20	20	20	0,6	1	20	0,5	1
	3	43,3	1635	1635	1635	1635	26,2	1	1635	13,2	1
		41,7	1580	1580	1595	1570	37,4	0,984	1590	16,3	0,997
		40	1530	1530	1555	1515	37,6	0,974	1550	17,5	0,997
		38,3	1450	1465	1485	1475	38,5	0,993	1475	15,5	0,993
		36,7	1400	1410	1425	1425	56,2	1	1415	18,6	0,993
		35	1330	1330	1345	1345	37,9	1	1345	15,0	1
		33,3	1250	1240	1260	1250	36,4	0,992	1260	22,3	1
		31,7	1175	1175	1190	1190	49,4	1	1190	14,1	1
		30	1105	1115	1125	1125	33,9	1	1125	20,4	1
		28,3	1060	1065	1070	1070	28,5	1	1070	13,0	1
		26,7	990	990	990	990	26,3	1	990	11,4	1
		25	870	835	870	870	25,5	1	870	11,6	1
		23,3	755	755	755	755	22,6	1	755	10,9	1
		21,7	650	650	650	650	16,6	1	650	9,7	1
		20	595	575	595	595	24,8	1	585	16,2	0,983
		18,3	480	495	495	495	16,9	1	495	15,2	1
		16,7	470	470	470	470	9,6	1	470	6,9	1
		15	335	335	335	335	8,3	1	335	5,5	1
		13,3	255	260	260	260	5,0	1	260	3,0	1
		11,7	185	185	185	185	4,4	1	185	3,0	1
		10	110	110	110	110	2,9	1	110	1,8	1
		8,3	110	95	95	95	2,5	0,864	95	1,6	0,864
		6,7	60	60	60	60	1,3	1	60	1,3	1
		5	20	20	20	20	0,5	1	20	0,6	1
		3,3	15	15	15	15	0,5	1	15	0,4	1
	2	65	1680	1665	1670	1680	21,0	1	1680	17,4	1
		62,5	1635	1630	1635	1640	27,4	1,003	1640*	16,5	1,003
		60	1600	1610	1590	1610	30,7	1	1610	17,1	1
		57,5	1545	1560	1560	1565	34,4	1,003	1565*	18,2	1,003
		55	1490	1500	1500	1500	37,6	1	1500	19,3	1
		52,5	1450	1445	1460	1460	39,0	1	1460	18,8	1
		50	1380	1380	1400	1380	40,8	0,986	1380	19,1	0,986
		47,5	1310	1310	1340	1325	42,8	0,989	1330	26,1	0,993

à suivre...

Tab5.8 : *suite...*

			CGW	**TMH**	**AHS**		**TE**			**TR**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		45	1250	1260	1260	1255	47,2	0,996	1260	25,1	1
		42,5	1195	1185	1195	1195	37,4	1	1190	16,5	0,996
		40	1150	1090	1150	1150	51,9	1	1150	24,2	1
		37,5	1010	975	1020	1010	32,2	0,990	1010	15,5	0,990
		35	920	920	925	925	27,0	1	920	14,6	0,995
		32,5	855	860	860	860	29,7	1	845	15,6	0,983
		30	790	770	800	800	30,2	1	745	15,5	0,931
		27,5	670	670	670	660	18,0	0,985	670	16,9	1
		25	580	560	580	580	16,0	1	580	14,6	1
		22,5	480	480	480	480	11,4	1	480	12,4	1
		20	395	410	410	410	9,4	1	400	14,2	0,976
		17,5	315	320	320	315	5,2	0,984	315	9,6	0,984
		15	240	240	240	240	3,9	1	240	5,8	1
		12,5	175	180	180	180	2,3	1	180	3,4	1
		10	80	80	80	80	1,5	1	80	1,4	1
		7,5	50	50	50	50	0,9	1	50	1,0	1
		5	20	20	20	20	0,4	1	20	0,7	1
64	4	20	1068	1068	1068	1068	36,8	1	1062	14,3	0,994
		18,8	912	912	912	912	30,5	1	912	16,3	1
		17,5	690	696	696	696	26,1	1	696	19,6	1
		16,2	546	522	528	528	20,1	0,967	528	10,0	0,967
		15	366	366	366	366	10,8	1	366	4,9	1
	3	26,7	1158	1152	1170	1164	64,0	0,995	1170	18,0	1
		25	1080	1080	1080	1080	39,6	1	1080	20,5	1
		23,3	972	990	1002	1002	36,0	1	1002	15,9	1
		21,7	894	876	894	894	28,5	1	894	18,0	1
		20	828	828	828	828	26,9	1	828	12,8	1
		18,3	642	612	642	642	23,7	1	642	20,9	1
		16,7	444	444	444	444	16,9	1	444	9,5	1
		15	282	282	282	282	8,2	1	282	4,6	1
	2	40	1242	1260	1260	1224	42,3	0,971	1254	32,6	0,995
		37,5	1176	1188	1188	1170	41,9	0,985	1182	25,8	0,995
		35	1104	1116	1116	1110	53,2	0,995	1116	28,6	1
		32,5	1032	1032	1032	1032	36,1	1	1032	16,1	1
		30	942	936	948	948	54,7	1	942	14,7	0,994
		27,5	888	888	888	888	30,5	1	888	13,4	1
		25	780	780	780	780	24,9	1	780	13,8	1
		22,5	660	660	660	660	19,5	1	660	16,1	1
		20	588	588	588	534	14,5	0,908	534	16,8	0,908
		17,5	360	360	360	360	12,8	1	360	9,7	1
		15	192	192	192	192	4,8	1	192	4,2	1
33	4	27,5	670	670	670	670	2,5	1	640	2,6	0,955
		26,2	670	670	670	610	2,4	0,910	620	2,9	0,925
		25	600	600	600	600	2,9	1	570	3,4	0,950
		23,8	560	560	560	560	2,9	1	560	2,7	1
		22,5	530	560	560	560	3,1	1	560	3,0	1
		21,2	490	490	500	500	3,3	1	500	2,2	1
		20	440	440	440	440	2,2	1	440	2,4	1
		18,8	380	380	390	390	2,1	1	390	2,0	1
		17,5	380	380	380	380	2,3	1	380	2,3	1
		16,2	350	350	350	350	1,8	1	340	1,6	0,971
		15	300	310	310	310	1,8	1	300	1,4	0,968
		13,8	260	260	270	270	1,6	1	260	1,3	0,963
		12,5	240	240	240	240	1,5	1	240	1,1	1

à suivre...

TAB5.8 : *suite...*

			CGW	**TMH**	**AHS**		**TE**			**TR**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*
		11,2	220	220	220	220	1,6	1	220	1,6	1
		10	180	190	190	190	1,2	1	170	1,1	0,895
		8,8	140	140	140	140	0,9	1	130	0,9	0,929
		7,5	100	100	100	100	0,7	1	90	0,6	0,900
		6,2	90	90	90	90	0,8	1	90	1,2	1
		5	30	30	30	30	0,6	1	30	0,7	1
		3,5	20	20	20	20	0,5	1	20	0,6	1
	3	36,7	720	750	760	760	3,8	1	750	3,5	0,987
		35	710	710	720	710	2,9	0,986	710	3,1	0,986
		33,3	710	710	710	710	3,9	1	710	3,1	1
		31,7	630	680	680	630	3,6	0,926	680	5,1	1
		30	620	640	640	640	3,7	1	640	3,4	1
		28,3	580	590	590	580	3,8	0,983	580	3,3	0,983
		26,7	550	550	570	570	4,5	1	570	3,6	1
		25	520	510	520	520	3,7	1	520	2,9	1
		23,3	470	470	480	480	3,4	1	480	3,1	1
		21,7	430	430	440	430	4,6	0,977	430	3,2	0,977
		20	380	380	380	380	2,6	1	380	2,8	1
		18,3	330	330	330	330	2,5	1	330	2,4	1
		16,7	300	300	300	300	2,5	1	300	1,7	1
		15	270	270	270	270	1,9	1	270	1,3	1
		13,3	230	230	230	230	1,6	1	220	1,3	0,957
		11,7	200	200	200	200	1,4	1	200	1,0	1
		10	170	170	170	170	1,0	1	150	0,8	0,882
		8,3	120	120	120	120	0,8	1	110	0,7	0,917
		6,7	90	90	90	90	0,6	1	80	0,8	0,889
		5	30	30	30	30	0,4	1	30	0,6	1
	2	55	800	800	800	800	3,3	1	800	3,3	1
		52,5	800	800	800	800	3,4	1	800	3,1	1
		50	780	780	790	790	4,8	1	780	4,5	0,987
		47,5	750	760	760	760	5,2	1	760	3,6	1
		45	710	710	720	720	3,6	1	710	2,9	0,986
		42,5	680	690	690	670	4,7	0,971	690	3,7	1
		40	660	660	660	660	6,0	1	650	3,7	0,985
		37,5	620	620	620	610	4,4	0,984	610	3,3	0,984
		35	590	590	590	570	4,0	0,966	590	4,2	1
		32,5	540	540	550	550	4,8	1	550	3,8	1
		30	470	490	510	510	4,6	1	510	4,5	1
		27,5	440	460	460	450	4,3	0,978	460	3,3	1
		25	390	410	410	410	3,6	1	390	2,5	0,951
		22,5	350	350	360	360	2,9	1	360	2,7	1
		20	300	290	300	300	2,7	1	300	2,8	1
		17,5	260	250	260	260	2,1	1	260	1,8	1
		15	220	220	220	220	1,5	1	220	1,5	1
		12,5	170	180	180	180	1,0	1	180	1,2	1
		10	150	150	150	150	0,7	1	150	1,0	1
		7,5	90	90	90	90	0,5	1	90	0,5	1
32	4	21,2	210	210	210	205	2,3	0,976	205	2,9	0,976
		20	190	190	190	190	2,3	1	185	2,2	0,974
		18,8	160	175	175	175	2,6	1	170	2,4	0,971
		18,2	160	165	165	165	2,3	1	165	2,3	1
		17,5	155	155	155	155	2,3	1	155	2,4	1
		16,2	130	130	130	130	1,5	1	120	1,5	0,923
		15	120	120	120	120	1,5	1	115	1,6	0,958

à suivre...

Tab5.8 : *suite...*

			CGW	**TMH**	**AHS**		**TE**			**TR**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*
		13,8	100	100	100	100	1,1	1	85	1,4	0,850
		12,5	70	75	75	75	1,1	1	75	1,3	1
		11,5	60	60	60	60	0,8	1	55	1,0	0,917
		10	45	45	45	45	0,7	1	40	1,1	0,889
		8,8	35	35	35	35	0,6	1	35	0,6	1
		7,5	25	25	25	25	0,5	1	25	0,4	1
		6,2	15	15	15	15	0,4	1	15	0,4	1
		5	15	15	15	15	0,4	1	15	0,4	1
	3	28,5	250	250	250	255	2,6	1,020	255*	2,2	1,020
		26,7	230	230	230	230	3,4	1	230	2,3	1
		25	215	220	220	220	2,8	1	220	2,2	1
		24,3	215	205	205	205	4,1	0,953	205	3,2	0,953
		23,3	190	190	190	190	3,0	1	185	2,8	0,974
		21,7	175	170	175	175	2,6	1	170	2,4	0,971
		20	155	155	155	155	2,2	1	145	1,8	0,935
		18,3	135	135	135	135	2,0	1	130	1,5	0,963
		16,7	115	115	115	115	1,5	1	105	1,4	0,913
		15,3	105	105	105	105	1,1	1	105	1,3	1
		13,3	75	70	70	70	0,8	0,933	70	1,2	0,933
		11,7	50	50	50	50	0,6	1	50	0,7	1
		10	40	40	40	40	0,5	1	40	1,0	1
		8,3	30	30	30	30	0,5	1	30	0,6	1
		6,7	15	15	15	15	0,4	1	10	0,3	0,667
		5	15	15	15	15	0,4	1	10	0,3	0,667
	2	42,5	280	280	280	270	4,7	0,964	280	4,0	1
		40	265	265	265	260	4,4	0,981	265	3,8	1
		37,5	250	250	250	250	4,9	1	250	3,5	1
		36,5	240	240	240	240	3,9	1	240	2,8	1
		35	235	235	235	230	4,5	0,979	230	3,6	0,979
		32,5	215	215	215	205	3,6	0,953	215	3,1	1
		30	190	190	195	190	3,7	0,974	195	2,9	1
		27,5	175	175	175	175	3,4	1	175	2,5	1
		25	155	155	155	155	3,2	1	155	2,1	1
		23	130	135	135	130	2,4	0,963	130	2,3	0,963
		20	110	110	110	110	1,8	1	110	1,8	1
		17,5	90	90	90	90	1,3	1	90	1,4	1
		15	80	80	80	80	1,0	1	80	1,2	1
		12,5	45	45	45	45	0,6	1	40	0,8	0,889
		10	30	30	30	30	0,4	1	25	0,7	0,833
		7,5	20	20	20	20	0,3	1	15	0,2	0,750
		5	15	15	15	15	0,3	1	15	0,2	1
21	4	11,2	180	180	180	180	0,6	1	160	1,0	0,889
		10	120	120	120	120	0,5	1	120	0,8	1
		9,5	120	120	120	120	0,5	1	120	0,8	1
		8,8	120	120	120	120	0,5	1	120	0,9	1
		8	105	105	105	105	0,5	1	105	1,0	1
		7,5	105	105	105	105	0,5	1	105	1,0	1
		6,8	70	70	70	70	0,5	1	70	0,7	1
		6,2	70	70	70	70	0,5	1	70	1,0	1
		5,8	70	70	70	70	0,5	1	70	1,0	1
		5	70	70	70	70	0,5	1	60	1,0	0,857
		3,8	10	10	10	10	0,4	1	10	0,4	1
	3	15	200	200	200	200	0,7	1	230*	0,9	1,150
		13,3	200	200	200	200	0,7	1	200	1,2	1

à suivre...

TAB5.8 : *suite...*

			CGW	**TMH**	**AHS**		**TE**			**TR**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		12,7	200	200	200	200	0,9	1	200	1,3	1
		11,7	165	165	165	165	0,7	1	165	1,3	1
		10,7	140	140	145	145	0,6	1	120	0,6	0,828
		10	120	120	120	120	0,6	1	120	0,7	1
		9	120	120	120	120	0,5	1	100	0,6	0,833
		8,3	105	105	105	105	0,5	1	90	0,7	0,857
		7,7	105	105	105	105	0,5	1	90	0,7	0,857
		6,7	70	70	70	70	0,4	1	70	0,7	1
		5	70	70	70	70	0,4	1	55	0,7	0,786
	2	22,5	270	270	275	275	1,0	1	275	1,3	1
		20	260	260	260	260	0,9	1	260	1,2	1
		19	230	230	230	230	0,9	1	230	1,4	1
		17,5	230	230	230	230	1,1	1	230	1,5	1
		16	200	200	200	200	0,8	1	200	1,1	1
		15	200	200	200	200	0,8	1	200	1,1	1
		13,5	190	190	190	190	0,7	1	190	1,1	1
		12,5	160	160	160	160	0,8	1	160	1,1	1
		11,5	140	140	140	140	0,6	1	140	1,5	1
		10	120	120	120	120	0,5	1	120	0,7	1
		7,5	90	90	90	90	0,4	1	90	1,0	1

Annexe5

Le tableau 5.9 présente les résultats numériques obtenus par les deux approches tabou avec pénalité avec et sans normalisation. Ces deux approches ont été testées sur une machine (*Pentium 4, 256 Mo*). Les notations suivantes sont considérées :

- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigé par la solution
- L : la longueur maximale d'une tournée
- TP : les résultats des meilleurs solutions obtenues avec l'approche tabou avec pénalité à fonction objectif non normalisée
- $\widehat{TP}$: les résultats des meilleurs solutions obtenues avec l'approche tabou avec pénalité à fonction objectif normalisée
- G_{tot} : le gain total obtenu par la solution de l'instance correspondante
- T_{cpu} : le temps mis, en secondes, pour résoudre l'instance correspondante.
- *Ratio* : Ratio de la solution fournie par la méthode correspondante par rapport à la meilleure solution fourni par les méthodes CGW, TMH, AHS (voir tableau 5.5 de l'Annexe 1).

TABLE 5.9 – *Résultats numériques des approches tabou avec pénalité*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	*Ratio*	G_{tot}	T_{cpu}	*Ratio*
100	4	60	1257	488,1	0,978	1281	305,1	0,997
		57,5	1252	330,1	0,994	1253	165,1	0,994
		55	1204	679,9	0,994	1199	364,5	0,990
		52,5	1136	221,8	0,978	1160	299,9	0,999
		50	1118	613,3	0,998	1115	214,8	0,996
		47,5	1055	272,7	0,994	1057	417,5	0,996
		45	971	252,0	0,994	963	270,2	0,986
		42,5	908	216,3	0,989	907	162,0	0,988
		40	877	263,3	0,997	876	138,6	0,995

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		37,5	815	218,0	0,993	815	108,9	0,993
		35	730	83,1	0,997	729	125,7	0,996
		32,5	656	81,8	0,998	652	52,3	0,992
		30	571	68,5	1	563	40,5	0,986
		27,5	460	26,4	0,998	460	23,1	0,998
		25	315	18,9	0,972	324	21,9	1
		22,5	172	3,0	0,940	172	2,8	0,940
		20	38	1,2	1	38	1,2	1
	3	80	1293	469,0	0,992	1281	228,2	0,982
		76,7	1275	303,6	0,985	1278	413,0	0,987
		73,3	1261	809,1	0,991	1237	244,8	0,972
		70	1245	668,6	0,996	1236	186,4	0,989
		66,7	1196	519,9	0,979	1194	195,5	0,977
		63,3	1145	360,8	0,977	1145	331,1	0,977
		60	1095	460,1	0,977	1089	238,2	0,971
		56,7	1034	678,9	0,974	1034	228,7	0,974
		53,3	971	380,9	0,992	964	228,3	0,985
		50	899	286,0	0,978	899	239,0	0,978
		46,7	852	227,8	0,990	858	280,8	0,997
		43,3	804	241,6	0,994	804	202,5	0,994
		40	727	342,3	0,997	729	307,3	1
		36,7	653	131,7	1	653	112,1	1
		33,3	576	73,0	0,995	576	72,3	0,995
		30	454	50,2	0,970	467	70,8	0,998
		26,7	332	30,9	0,991	332	30,3	0,991
		23,3	177	5,4	0,917	182	5,7	0,943
		20	38	1,2	1	38	1,3	1
	2	120	1303	339,1	0,998	1304	388,2	0,998
		115	1286	496,2	0,988	1290	522,4	0,992
		110	1283	318,2	0,998	1281	294,6	0,996
		105	1262	403,8	0,998	1262	401,8	0,998
		100	1237	539,7	0,997	1237	454,6	0,997
		95	1204	652,1	0,989	1204	838,0	0,989
		90	1150	550,3	0,980	1156	606,6	0,985
		85	1102	527,7	0,973	1101	541,9	0,973
		80	1045	441,6	0,973	1044	395,8	0,972

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		75	974	350,3	0,953	983	633,6	0,962
		70	918	317,7	0,954	911	292,3	0,947
		65	860	252,5	0,937	860	324,1	0,937
		60	806	286,2	0,965	808	355,9	0,968
		55	742	830,5	0,980	742	511,0	0,980
		50	682	396,3	0,993	687	392,5	1
		45	609	208,5	0,985	602	367,2	0,974
		40	531	215,6	1	531	202,4	1
		35	452	116,9	1	452	114,9	1
		30	341	75,3	1	341	50,8	1
		25	161	21,8	0,782	161	16,8	0,782
102	4	100	1062	125,8	0,986	1073	187,0	0,996
		95	1022	96,0	1	1022	87,7	1
		90	956	100,0	0,986	956	83,5	0,986
		85	900	84,7	0,990	900	75,5	0,990
		80	832	64,2	0,983	832	64,9	0,983
		75	776	70,4	0,994	776	67,8	0,994
		70	724	64,2	0,997	724	65,5	0,997
		65	641	76,8	0,992	641	70,7	0,992
		60	576	42,1	0,976	576	39,6	0,976
		55	518	43,8	0,996	514	39,8	0,988
		50	454	30,9	0,983	457	51,4	0,989
		45	359	27,4	0,981	352	17,4	0,962
		40	283	11,5	0,993	283	11,0	0,993
		35	209	6,1	0,963	209	5,9	0,963
		30	153	3,4	0,933	153	3,5	0,933
		25	109	2,3	0,886	109	2,2	0,886
		20	79	1,4	1	79	1,4	1
		15	30	1,1	0,652	30	1,1	0,652
		10	30	1,0	1	30	1,0	1
	3	133,3	1110	147,5	0,994	1111	141,1	0,995
		126,7	1074	217,9	0,994	1065	166,5	0,985
		120	1019	144,4	0,993	1019	142,5	0,993
		113,3	978	175,6	0,991	976	123,4	0,989
		106,7	918	129,5	0,990	918	123,0	0,990
		100	859	123,2	0,983	859	118,4	0,983

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		93,3	792	109,3	0,973	792	107,3	0,973
		86,7	750	139,4	0,984	742	113,8	0,974
		80	677	83,8	0,991	677	78,3	0,991
		73,3	628	118,4	0,992	628	93,7	0,992
		66,7	557	68,1	0,988	557	64,4	0,988
		60	481	51,4	0,988	481	49,5	0,988
		53,3	401	38,0	0,944	401	36,8	0,944
		46,7	307	24,0	0,892	307	22,9	0,892
		40	227	11,5	0,919	227	11,2	0,919
		33,3	170	5,1	0,971	170	5,1	0,971
		26,7	105	2,9	0,897	105	2,9	0,897
		20	48	1,3	0,608	48	1,3	0,608
		13,3	32	1,0	0,696	32	1,1	0,696
	2	200	1163	391,8	0,986	1163	239,1	0,986
		190	1121	202,9	0,987	1121	233,2	0,987
		180	1080	393,7	0,987	1080	345,9	0,987
		170	1019	501,5	0,977	1019	418,1	0,977
		160	976	205,6	0,974	976	219,2	0,974
		150	927	212,5	0,981	927	202,8	0,981
		140	863	204,2	0,972	863	191,9	0,972
		130	818	171,7	0,989	825	188,8	0,998
		120	758	163,5	0,988	758	152,1	0,988
		110	705	135,5	1	705	130,8	1
		100	644	140,4	1	644	137,5	1
		90	579	139,3	1	579	129,4	1
		80	517	129,8	0,992	517	120,8	0,992
		70	457	156,8	0,996	457	100,8	0,996
		60	387	80,9	1	387	69,1	1
		50	290	41,2	1	290	38,5	1
		40	190	15,5	1	190	15,6	1
		30	101	5,0	1	101	4,7	1
		20	64	1,2	1	64	1,2	1
		10	30	1,0	1	30	0,9	1
66	4	32,5	1570	33,3	0,969	1570	24,4	0,969
		31,2	1520	18,7	1	1520	17,5	1
		30	1440	18,4	0,993	1450	31,5	1

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		28,8	1385	24,8	0,996	1380	18,6	0,993
		27,5	1320	18,7	1	1320	18,4	1
		26,2	1300	17,8	1	1300	17,1	1
		25	1160	20,9	1	1160	20,1	1
		23,8	1010	16,9	0,981	1020	15,7	0,990
		22,5	960	26,2	1	960	18,4	1
		21,2	835	18,2	0,971	855	19,3	0,994
		20	735	14,7	0,961	735	14,2	0,961
		18,8	670	11,4	0,971	670	11,5	0,971
		17,5	620	12,9	1	620	11,6	1
		16,2	515	8,7	0,928	515	8,8	0,928
		15	400	6,9	0,930	400	6,7	0,930
		13,8	340	4,3	1	340	4,5	1
		12,5	340	4,2	1	340	4,5	1
		11,2	235	2,7	0,979	230	3,0	0,958
		10	140	1,5	1	140	1,6	1
		8,8	140	1,6	1	140	1,8	1
		7,5	80	1,0	1	80	1,1	1
		6,2	20	0,6	1	20	0,7	1
		5	20	0,6	1	20	0,7	1
		3,8	20	0,6	1	20	0,7	1
	3	43,3	1635	21,6	1	1635	21,7	1
		41,7	1590	27,9	0,997	1590	26,7	0,997
		40	1535	38,6	0,987	1530	44,6	0,984
		38,3	1475	27,9	0,993	1475	29,3	0,993
		36,7	1425	33,0	1	1410	33,5	0,989
		35	1345	27,9	1	1345	37,0	1
		33,3	1255	26,7	0,996	1260	32,7	1
		31,7	1190	30,8	1	1190	28,7	1
		30	1120	23,6	0,996	1125	29,3	1
		28,3	1065	24,0	0,995	1070	25,1	1
		26,7	990	19,6	1	990	18,6	1
		25	870	21,1	1	870	19,4	1
		23,3	755	29,2	1	755	17,3	1
		21,7	650	18,5	1	650	15,3	1
		20	575	17,6	0,966	575	16,0	0,966

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		18,3	485	14,1	0,980	485	13,0	0,980
		16,7	470	12,3	1	470	11,3	1
		15	335	9,4	1	335	8,3	1
		13,3	260	4,3	1	260	4,2	1
		11,7	185	4,3	1	185	4,1	1
		10	110	1,7	1	110	1,5	1
		8,3	95	1,3	0,864	95	1,0	0,864
		6,7	60	1,0	1	60	1,0	1
		5	20	0,5	1	20	0,4	1
		3,3	15	0,5	1	15	0,4	1
	2	65	1680	25,0	1	1680	24,9	1
		62,5	1645	32,9	1,006	1645	34,7	1,006
		60	1610	31,8	1	1610	27,3	1
		57,5	1565	32,6	1	1565	28,3	1
		55	1500	29,3	0,997	1500	27,6	0,997
		52,5	1460	29,4	1	1460	28,8	1
		50	1380	30,5	0,986	1380	29,8	0,986
		47,5	1340	43,9	1	1330	36,5	0,993
		45	1255	44,0	0,996	1260	42,2	1
		42,5	1190	29,1	0,996	1190	27,8	0,996
		40	1150	45,5	1	1150	43,4	1
		37,5	1010	29,1	0,990	1010	27,0	0,990
		35	925	27,7	1	925	22,1	1
		32,5	825	22,5	0,959	825	21,3	0,959
		30	740	21,5	0,925	740	20,6	0,925
		27,5	660	27,3	0,985	660	29,6	0,985
		25	570	22,2	0,983	570	20,9	0,983
		22,5	480	22,3	1	480	20,7	1
		20	400	25,0	0,976	400	20,4	0,976
		17,5	315	17,8	0,984	315	16,7	0,984
		15	240	11,0	1	240	9,4	1
		12,5	180	6,0	1	180	5,2	1
		10	80	2,0	1	80	1,6	1
		7,5	50	1,3	1	50	1,1	1
		5	20	0,5	1	20	0,4	1
64	4	20	1062	27,1	0,994	1062	25,4	0,994

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		18,8	912	29,3	1	912	32,5	1
		17,5	696	30,7	1	696	31,2	1
		16,2	528	17,2	0,967	528	15,2	0,967
		15	366	7,0	0,938	366	6,7	0,938
	3	26,7	1170	34,0	1	1170	39,7	1
		25	1080	39,5	1	1068	27,8	0,989
		23,3	1002	28,7	1	1002	27,2	1
		21,7	894	30,1	1	894	26,9	1
		20	828	24,7	1	828	23,1	1
		18,3	642	27,6	1	636	34,6	0,991
		16,7	444	17,4	1	444	14,3	1
		15	282	7,6	1	282	7,1	1
	2	40	1224	49,9	0,971	1224	47,1	0,971
		37,5	1170	47,3	0,985	1170	44,0	0,985
		35	1104	34,3	0,989	1104	33,3	0,989
		32,5	1032	27,1	1	1032	26,1	1
		30	942	25,1	0,994	942	24,3	0,994
		27,5	888	23,6	1	888	22,8	1
		25	780	22,4	1	780	21,6	1
		22,5	660	21,6	1	660	20,9	1
		20	534	33,6	0,908	534	32,9	0,908
		17,5	360	18,4	1	360	16,8	1
		15	192	6,3	1	192	5,9	1
33	4	27,5	670	2,1	1	670	1,7	1
		26,2	610	1,6	0,910	670	1,9	1
		25	540	1,8	0,900	540	1,5	0,900
		23,8	560	2,1	1	560	1,8	1
		22,5	500	2,3	0,893	500	2,0	0,893
		21,2	500	1,9	1	500	1,6	1
		20	440	1,5	1	440	1,2	1
		18,8	390	1,5	1	390	1,2	1
		17,5	380	1,7	1	370	1,3	0,974
		16,2	340	1,0	0,919	340	0,7	0,919
		15	300	0,9	0,968	300	0,6	0,968
		13,8	260	1,0	0,963	260	0,7	0,963
		12,5	240	0,9	1	240	0,5	1

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		11,2	220	1,0	1	220	0,6	1
		10	170	0,7	0,895	170	0,4	0,895
		8,8	130	0,5	0,929	130	0,3	0,929
		7,5	90	0,4	0,900	90	0,2	0,900
		6,2	90	0,5	1	90	0,3	1
		5	30	0,3	1	30	0,1	1
		3,5	20	0,3	1	20	0,1	1
	3	36,7	740	3,1	0,974	760	3,8	1
		35	720	3,4	1,014	720	3,0	1,014
		33,3	710	3,2	0,986	710	2,7	0,986
		31,7	680	3,7	1	680	3,3	1
		30	640	3,1	1	640	2,8	1
		28,3	590	4,0	1	590	3,5	1
		26,7	560	3,8	0,982	550	3,1	0,965
		25	520	3,2	1	520	2,8	1
		23,3	460	2,4	0,958	460	2,0	0,958
		21,7	440	4,2	1	440	3,9	1
		20	370	2,2	0,974	380	1,9	1
		18,3	330	2,1	1	330	1,8	1
		16,7	300	1,9	1	300	1,5	1
		15	270	1,3	1	270	1,0	1
		13,3	220	1,3	0,957	220	1,0	0,957
		11,7	200	0,9	1	200	0,7	1
		10	150	0,8	0,882	150	0,5	0,882
		8,3	110	0,6	0,917	110	0,3	0,917
		6,7	80	0,4	0,889	80	0,3	0,889
		5	30	0,3	1	30	0,2	1
	2	55	800	3,7	1	800	3,3	1
		52,5	800	3,8	1	800	3,4	1
		50	770	4,0	0,975	770	3,7	0,975
		47,5	760	4,3	1	760	4,3	1
		45	710	4,0	0,986	710	3,1	0,986
		42,5	690	4,7	1	690	4,4	1
		40	650	4,8	0,985	640	4,4	0,970
		37,5	610	3,8	0,984	600	3,4	0,968
		35	580	3,7	0,983	580	3,4	0,983

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		32,5	550	5,3	1	550	5,0	1
		30	500	4,6	0,980	510	5,0	1
		27,5	460	4,6	1	450	3,7	0,978
		25	390	3,9	0,951	390	3,6	0,951
		22,5	360	3,3	1	360	3,1	1
		20	290	2,8	0,967	290	2,5	0,967
		17,5	260	2,1	1	260	1,8	1
		15	220	1,4	1	220	1,1	1
		12,5	180	1,1	1	180	0,9	1
		10	150	0,7	1	150	0,5	1
		7,5	90	0,4	1	90	0,3	1
32	4	21,2	205	1,5	0,976	205	1,2	0,976
		20	185	1,6	0,974	185	1,3	0,974
		18,8	175	2,2	1	175	1,7	1
		18,2	165	1,3	1	165	1,0	1
		17,5	155	1,6	1	155	1,2	1
		16,2	120	1,0	0,923	120	0,7	0,923
		15	115	0,9	0,958	115	0,6	0,958
		13,8	85	0,7	0,850	85	0,5	0,850
		12,5	75	0,8	1	75	0,5	1
		11,5	55	0,5	0,917	55	0,3	0,917
		10	40	0,4	0,889	40	0,2	0,889
		8,8	35	0,4	1	35	0,2	1
		7,5	25	0,3	1	25	0,1	1
		6,2	15	0,3	1	15	0,1	1
		5	15	0,3	1	15	0,1	1
	3	28,5	255	2,0	1,020	255	1,7	1,020
		26,7	230	2,9	1	230	2,5	1
		25	220	2,2	1	220	1,9	1
		24,3	205	3,5	0,953	205	3,1	0,953
		23,3	185	2,3	0,974	185	1,9	0,974
		21,7	170	2,2	0,971	170	1,8	0,971
		20	145	1,9	0,935	145	1,6	0,935
		18,3	130	1,4	0,963	130	1,1	0,963
		16,7	105	1,3	0,913	105	0,9	0,913
		15,3	105	1,0	1	105	0,8	1

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		13,3	70	0,7	0,933	70	0,5	0,933
		11,7	50	0,5	1	50	0,3	1
		10	40	0,3	1	40	0,2	1
		8,3	30	0,4	1	30	0,2	1
		6,7	10	0,3	0,667	10	0,1	0,667
		5	10	0,3	0,667	10	0,1	0,667
	2	42,5	275	4,1	0,982	275	3,4	0,982
		40	255	4,4	0,962	255	4,0	0,962
		37,5	250	4,7	1	245	4,2	0,980
		36,5	240	3,5	1	240	3,4	1
		35	215	4,0	0,915	215	3,7	0,915
		32,5	205	3,6	0,953	205	3,2	0,953
		30	190	3,6	0,974	190	3,3	0,974
		27,5	175	3,5	1	175	3,0	1
		25	155	3,0	1	155	2,7	1
		23	130	2,5	0,963	130	2,2	0,963
		20	110	2,2	1	110	1,9	1
		17,5	90	1,5	1	90	1,3	1
		15	80	1,1	1	80	0,9	1
		12,5	40	0,7	0,889	40	0,5	0,889
		10	25	0,4	0,833	25	0,2	0,833
		7,5	15	0,3	0,750	15	0,2	0,750
		5	15	0,2	1	15	0,1	1
21	4	11,2	160	0,3	0,889	160	0,1	0,889
		10	120	0,3	1	120	0,1	1
		9,5	120	0,3	1	120	0,1	1
		8,8	120	0,3	1	120	0,1	1
		8	105	0,3	1	105	0,1	1
		7,5	105	0,3	1	105	0,1	1
		6,8	70	0,3	1	70	0,1	1
		6,2	70	0,3	1	70	0,1	1
		5,8	70	0,3	1	70	0,1	1
		5	60	0,3	0,857	60	0,1	0,857
		3,8	10	0,2	1	10	0,1	1
	3	15	200	0,5	1	200	0,3	1
		13,3	200	0,5	1	200	0,3	1

à suivre...

TAB5.9 : *suite...*

			TP			$\widehat{TP}$		
n	m	L	G_{tot}	T_{cpu}	$Ratio$	G_{tot}	T_{cpu}	$Ratio$
		12,7	200	0,5	1	200	0,4	1
		11,7	165	0,5	0,971	165	0,3	0,971
		10,7	120	0,4	0,828	120	0,3	0,828
		10	120	0,4	1	120	0,3	1
		9	100	0,3	0,833	100	0,2	0,833
		8,3	90	0,3	0,857	90	0,2	0,857
		7,7	90	0,3	0,857	90	0,2	0,857
		6,7	70	0,3	1	70	0,2	1
		5	55	0,3	0,786	55	0,2	0,786
	2	22,5	275	1,0	1	275	0,8	1
		20	260	0,9	1	260	0,7	1
		19	230	0,8	1	230	0,6	1
		17,5	230	1,0	1	230	0,8	1
		16	200	0,8	1	200	0,6	1
		15	200	0,7	1	200	0,5	1
		13,5	190	0,7	1	190	0,5	1
		12,5	160	0,7	1	160	0,5	1
		11,5	125	0,6	0,893	125	0,4	0,893
		10	120	0,5	1	120	0,3	1
		7,5	90	0,4	1	90	0,3	1

Annexe6

Le tableau 5.10 présente les résultats numériques obtenus par les approches notre approche basée sur le principe de la mémoire adaptative comparée au résultats obtenus par les méthodes développées par (Archetti *et al.*, 2007b) considérées comme les meilleurs compromis pour le PmTS. Les notations suivantes sont considérées :

- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigé par la solution
- L : la longueur maximale d'une tournée
- **FVNS** : les résultats des solutions obtenues avec l'approche rapide avec la recherche à voisinage variable de Archetti *et al.* (2007b),
- **SVNS** : les résultats des solutions obtenues avec l'approche lente avec la recherche à voisinage variable de Archetti *et al.* (2007b),
- **MA** : les résultats obtenus par notre approche basée sur le principe de la mémoire adaptative,
- G_{tot} : le gain total obtenu par la solution de l'instance correspondante,
- T_{tot} : le temps de calcul CPU nécessaire pour résoudre l'instance correspondante

TABLE 5.10 – *Résultats numériques de l'approche basée sur le principe de mémoire adaptative*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
100	4	60	1282	1285	1282	201,4
		57,5	1259	1260	1260	173,3
		55	1207	1207	1207	153,5
		52,5	1160	1161	1160	149,8
		50	1120	1120	1120	147,0
		47,5	1051	1061	1061	143,8
		45	968	976	976	117,8

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		42,5	916	918	916	129,0
		40	879	880	880	96,6
		37,5	821	821	821	85,2
		35	732	732	732	66,5
		32,5	657	657	657	50,1
		30	571	571	571	33,9
		27,5	461	461	461	25,3
		25	324	324	324	13,6
		22,5	183	183	183	6,9
		20	38	38	38	4,7
	3	80	1298	1304	1295	218,5
		76,7	1289	1295	1295	185,0
		73,3	1272	1273	1273	219,0
		70	1250	1245	1250	199,3
		66,7	1208	1222	1222	210,1
		63,3	1172	1172	1172	205,8
		60	1111	1121	1114	178,2
		56,7	1056	1062	1062	159,1
		53,3	975	979	979	148,5
		50	919	919	919	129,3
		46,7	861	861	861	124,8
		43,3	806	809	809	101,7
		40	724	729	727	78,7
		36,7	653	653	646	67,2
		33,3	579	579	579	54,4
		30	468	468	468	31,8
		26,7	335	335	335	27,9
		23,3	193	193	193	16,1
		20	38	38	38	4,0
	2	120	1306	1306	1306	329,6
		115	1300	1301	1300	232,3
		110	1283	1285	1275	217,8
		105	1265	1263	1263	255,6
		100	1239	1241	1241	225,4
		95	1216	1218	1218	248,2
		90	1159	1174	1159	218,3

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		85	1132	1132	1132	203,8
		80	1073	1074	1074	235,4
		75	1019	1022	1022	208,0
		70	962	962	962	161,9
		65	916	918	916	171,3
		60	827	835	835	136,9
		55	750	757	757	132,7
		50	684	687	687	118,8
		45	618	618	618	105,6
		40	531	531	528	61,8
		35	452	452	452	45,2
		30	341	341	341	34,5
		25	206	206	206	18,7
102	4	100	1077	1077	1077	101,3
		95	1021	1022	1021	95,1
		90	970	970	970	97,2
		85	899	909	901	91,4
		80	846	846	846	66,7
		75	770	781	777	60,6
		70	715	726	730	59,0
		65	646	646	646	66,9
		60	588	590	588	42,8
		55	518	520	520	39,8
		50	462	462	462	28,0
		45	366	366	366	26,3
		40	285	285	285	12,0
		35	217	217	217	8,4
		30	164	164	164	6,9
		25	123	123	123	3,8
		20	79	79	79	5,2
		15	46	46	46	3,3
		10	30	30	30	3,3
	3	133,3	1112	1117	1114	124,1
		126,7	1079	1081	1081	116,5
		120	1024	1026	1026	115,2
		113,3	978	987	987	98,1

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		106,7	926	927	927	93,9
		100	871	874	874	87,7
		93,3	814	813	814	83,6
		86,7	745	762	749	71,8
		80	681	683	681	68,1
		73,3	632	633	632	58,0
		66,7	562	564	563	47,1
		60	487	487	487	46,8
		53,3	425	425	422	54,6
		46,7	344	344	344	23,3
		40	247	247	247	15,6
		33,3	175	175	175	14,5
		26,7	117	117	117	7,2
		20	79	79	79	5,3
		13,3	46	46	46	2,9
	2	200	1168	1179	1179	197,1
		190	1136	1135	1136	180,2
		180	1094	1094	1094	170,8
		170	1038	1043	1038	161,1
		160	1002	1002	1002	150,5
		150	945	945	945	143,1
		140	883	888	888	135,3
		130	824	827	827	172,1
		120	759	767	767	154,0
		110	704	705	705	103,3
		100	643	644	644	116,3
		90	575	579	579	110,4
		80	521	521	521	76,0
		70	459	459	459	71,1
		60	387	387	387	49,2
		50	289	290	290	33,4
		40	190	190	190	18,4
		30	101	101	101	8,9
		20	64	64	64	5,0
		10	30	30	30	3,3
66	4	32,5	1620	1620	1620	49,1

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		31,2	1520	1520	1520	43,5
		30	1450	1450	1450	43,7
		28,8	1390	1390	1385	49,4
		27,5	1320	1320	1320	53,8
		26,2	1300	1300	1300	74,1
		25	1160	1160	1160	39,2
		23,8	1030	1030	1030	37,5
		22,5	960	960	960	32,3
		21,2	860	860	860	34,6
		20	765	765	765	29,3
		18,8	690	690	690	29,7
		17,5	620	620	620	40,3
		16,2	555	555	555	20,8
		15	430	430	430	14,6
		13,8	340	340	340	22,3
		12,5	340	340	340	12,4
		11,2	240	240	240	10,4
		10	140	140	140	12,3
		8,8	140	140	140	13,0
		7,5	80	80	80	5,8
		6,2	20	20	20	2,8
		5	20	20	20	2,8
		3,8	20	20	20	2,7
	3	43,3	1635	1635	1635	47,4
		41,7	1595	1595	1595	45,7
		40	1555	1555	1555	50,8
		38,3	1485	1485	1485	48,4
		36,7	1425	1425	1425	47,9
		35	1345	1345	1345	48,0
		33,3	1260	1260	1260	46,8
		31,7	1190	1190	1190	54,9
		30	1125	1125	1125	45,7
		28,3	1070	1070	1070	41,6
		26,7	990	990	990	40,2
		25	870	870	870	34,9
		23,3	755	755	755	36,3

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		21,7	650	650	650	42,4
		20	595	595	595	40,6
		18,3	495	495	495	27,4
		16,7	470	470	470	30,0
		15	335	335	335	20,0
		13,3	260	260	260	17,2
		11,7	185	185	185	14,7
		10	110	110	110	9,6
		8,3	95	95	95	10,0
		6,7	60	60	60	5,7
		5	20	20	20	3,1
		3,3	15	15	15	2,4
	2	65	1670	1670	1680	70,9
		62,5	1635	1635	1640	81,9
		60	1590	1610	1610	62,2
		57,5	1560	1565	1565	91,0
		55	1500	1505	1500	83,4
		52,5	1460	1460	1460	75,3
		50	1400	1400	1400	68,6
		47,5	1340	1340	1340	64,2
		45	1260	1260	1260	75,8
		42,5	1195	1195	1190	61,8
		40	1150	1150	1150	59,4
		37,5	1020	1020	1020	65,6
		35	925	925	925	50,8
		32,5	860	860	860	46,6
		30	800	800	800	60,5
		27,5	670	670	670	55,7
		25	580	580	580	31,8
		22,5	480	480	480	38,4
		20	410	410	410	28,0
		17,5	320	320	320	28,4
		15	240	240	240	23,3
		12,5	180	180	180	13,5
		10	80	80	80	9,5
		7,5	50	50	50	3,9

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		5	20	20	20	4,9
64	4	20	1068	1068	1068	37,1
		18,8	912	912	912	45,7
		17,5	696	696	696	30,5
		16,2	528	528	528	13,7
		15	366	390	366	7,0
	3	26,7	1170	1170	1170	51,7
		25	1080	1080	1080	45,8
		23,3	1002	1002	1002	43,9
		21,7	894	894	894	61,8
		20	828	828	828	32,8
		18,3	642	642	642	28,8
		16,7	444	444	444	16,2
		15	282	282	282	6,0
	2	40	1260	1260	1260	97,1
		37,5	1188	1188	1188	91,8
		35	1116	1116	1116	88,3
		32,5	1032	1032	1032	76,0
		30	948	948	948	67,4
		27,5	888	888	888	62,0
		25	780	780	780	63,5
		22,5	660	660	660	55,5
		20	588	588	588	56,4
		17,5	360	360	360	31,0
		15	192	192	192	12,3
33	4	27,5	670	670	670	6,9
		26,2	670	670	670	6,1
		25	600	600	600	5,4
		23,8	560	560	560	9,1
		22,5	560	560	560	5,3
		21,2	500	500	500	7,1
		20	440	440	440	6,9
		18,8	390	390	390	5,0
		17,5	380	380	380	6,5
		16,2	350	370	350	4,4
		15	310	310	310	4,7

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		13,8	270	270	270	3,3
		12,5	240	240	240	3,9
		11,2	220	220	220	5,8
		10	190	190	190	2,1
		8,8	140	140	140	1,9
		7,5	100	100	100	1,6
		6,2	90	90	90	1,7
		5	30	30	30	1,3
		3,5	20	20	20	1,2
	3	36,7	760	760	760	10,1
		35	720	720	720	10,1
		33,3	710	710	710	15,5
		31,7	680	680	680	9,8
		30	640	640	640	7,5
		28,3	590	590	590	12,5
		26,7	570	570	570	9,6
		25	520	520	520	11,2
		23,3	480	480	480	7,7
		21,7	440	440	440	9,4
		20	380	380	380	8,0
		18,3	330	330	330	10,0
		16,7	300	300	300	10,0
		15	270	270	270	7,7
		13,3	230	230	230	6,4
		11,7	200	200	200	4,8
		10	170	170	170	3,1
		8,3	120	120	120	2,1
		6,7	90	90	90	1,4
		5	30	30	30	1,2
	2	55	800	800	800	23,5
		52,5	800	800	800	16,4
		50	790	790	790	11,8
		47,5	760	760	760	20,2
		45	720	720	720	13,1
		42,5	690	690	690	20,7
		40	660	660	660	12,7

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		37,5	620	620	620	17,5
		35	590	590	590	18,6
		32,5	550	550	550	17,2
		30	510	510	510	15,5
		27,5	460	460	460	10,2
		25	410	410	410	11,7
		22,5	360	360	360	10,2
		20	300	300	300	9,0
		17,5	260	260	260	7,0
		15	220	220	220	6,6
		12,5	180	180	180	3,8
		10	150	150	150	2,6
		7,5	90	90	90	1,9
32	4	21,2	210	210	210	6,5
		20	190	190	190	7,5
		18,8	175	175	175	4,6
		18,2	165	165	165	9,4
		17,5	155	155	155	7,3
		16,2	130	130	130	3,8
		15	120	120	120	5,5
		13,8	100	100	100	3,8
		12,5	75	75	75	3,8
		11,5	60	60	60	2,9
		10	45	45	45	1,9
		8,8	35	35	35	1,4
		7,5	25	25	25	1,2
		6,2	15	15	15	1,0
		5	15	15	15	0,9
	3	28,5	250	250	255	10,3
		26,7	230	230	230	9,6
		25	220	220	220	9,6
		24,3	205	205	205	14,6
		23,3	190	190	190	9,8
		21,7	175	175	175	9,6
		20	155	155	155	10,0
		18,3	135	135	135	7,4

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
		16,7	115	115	115	8,9
		15,3	105	105	105	5,3
		13,3	70	70	70	3,7
		11,7	50	50	50	2,1
		10	40	40	40	2,0
		8,3	30	30	30	1,4
		6,7	15	15	15	0,9
		5	15	15	15	0,8
	2	42,5	280	280	280	14,7
		40	265	265	265	14,9
		37,5	250	250	250	15,9
		36,5	240	240	240	24,1
		35	235	235	235	16,2
		32,5	215	215	215	18,3
		30	195	195	195	14,3
		27,5	175	175	175	13,5
		25	155	155	155	11,0
		23	135	135	135	9,7
		20	110	110	110	8,2
		17,5	90	90	90	5,5
		15	80	80	80	4,4
		12,5	45	45	45	2,5
		10	30	30	30	1,6
		7,5	20	20	20	1,0
		5	15	15	15	0,8
21	4	11,2	180	180	180	1,6
		10	120	120	120	2,9
		9,5	120	120	120	2,6
		8,8	120	120	120	2,3
		8	105	105	105	1,9
		7,5	105	105	105	2,2
		6,8	70	70	70	1,8
		6,2	70	70	70	1,8
		5,8	70	70	70	1,6
		5	70	70	70	1,8
		3,8	10	10	10	0,8

à suivre...

TAB5.10 : *suite...*

			FVNS	**SVNS**	**MA**	
n	m	L	G_{tot}	G_{tot}	G_{tot}	T_{tot}
	3	15	200	200	200	5,3
		13,3	200	200	200	2,9
		12,7	200	200	200	3,0
		11,7	165	165	165	3,0
		10,7	145	145	145	2,7
		10	120	120	120	3,5
		9	120	120	120	2,8
		8,3	105	105	105	1,4
		7,7	105	105	105	1,8
		6,7	70	70	70	1,8
		5	70	70	70	1,6
	2	22,5	275	275	275	5,6
		20	260	260	260	5,7
		19	230	230	230	4,9
		17,5	230	230	230	4,5
		16	200	200	200	4,1
		15	200	200	200	4,3
		13,5	190	190	190	4,3
		12,5	160	160	160	3,4
		11,5	140	140	140	2,7
		10	120	120	120	2,5
		7,5	90	90	90	1,9

Annexe7

Le tableau 5.11 présente les résultats numériques obtenus par l'approche basée sur le principe de la mémoire adaptative comparée sur 139 nouvelles instances de tailles 121 et 151. Les notations suivantes sont considérées :

- n : le nombre des sommets y compris les dépôts
- m : le nombre des tournées exigé par la solution
- L : la longueur maximale d'une tournée
- **MA** : les résultats obtenus par notre approche basée sur le principe de la mémoire adaptative,
- G_{tot} : le gain total obtenu par la solution de l'instance correspondante,
- T_{tot} : le temps de calcul CPU nécessaire pour résoudre l'instance correspondante

TABLE 5.11 – *Résultats numériques de l'approche basée sur le principe de mémoire adaptative sur les nouvelles instances*

			MA	
n	m	L	G_{tot}	T_{tot}
151	4	210	2225	500,0
		200	2221	459,8
		190	2194	450,0
		180	2165	430,7
		170	2124	428,1
		160	2060	421,7
		150	1986	419,6
		140	1903	408,4
		130	1808	406,5
		120	1716	391,3
		110	1598	333,4
		100	1436	279,0

à suivre...

TAB5.11 : *suite...*

			MA	
n	m	L	G_{tot}	T_{tot}
		90	1266	198,3
		80	1143	170,9
		70	967	133,8
		60	765	102,9
		50	558	44,2
		40	376	28,6
		30	250	15,7
	3	250	2215	591,4
		240	2200	592,0
		230	2164	530,0
		220	2120	530,0
		210	2102	664,3
		200	2061	656,6
		190	1996	562,8
		180	1893	542,2
		170	1837	560,8
		160	1773	592,0
		150	1687	479,5
		140	1572	403,2
		130	1467	364,0
		120	1371	295,3
		110	1295	287,6
		100	1192	222,3
		90	1054	227,7
		80	957	181,7
		70	815	117,6
		60	645	85,9
		50	516	47,2
		40	376	25,5
		30	250	10,8
	2	280	2026	782,0
		270	1980	745,5
		260	1939	805,1
		250	1898	692,3
		240	1854	657,4
		230	1780	678,7

à suivre...

TAB5.11 : *suite...*

			MA	
n	m	L	G_{tot}	T_{tot}
		220	1745	749,4
		210	1687	672,4
		200	1619	625,4
		190	1541	490,8
		180	1466	579,5
		170	1395	523,0
		160	1315	406,4
		150	1245	386,6
		140	1166	311,5
		130	1102	385,4
		120	1030	246,9
		110	949	246,4
		100	863	213,3
		90	762	150,7
		80	682	156,1
		70	592	121,2
		60	487	103,6
		50	376	76,8
		40	291	38,0
		30	208	27,0
121	4	200	1152	368,5
		190	1051	219,0
		180	1015	234,7
		170	881	140,5
		160	826	119,4
		150	795	140,8
		140	757	114,2
		130	713	138,0
		120	614	93,1
		110	533	174,1
		100	432	85,1
		90	302	50,7
		80	273	51,0
		70	242	37,7
		60	211	34,0
		50	271	33,6

à suivre...

TAB5.11 : *suite...*

			MA	
n	m	L	G_{tot}	T_{tot}
		40	132	16,0
		30	83	10,0
		20	32	5,2
	3	290	1362	303,0
		280	1360	311,6
		270	1345	317,0
		260	1324	336,5
		250	1344	405,7
		240	1317	417,5
		230	1185	383,3
		220	1122	400,9
		210	1093	438,3
		200	1032	499,4
		190	953	465,9
		180	920	338,3
		170	864	332,0
		160	795	140,2
		150	795	129,6
		140	756	154,5
		130	713	134,6
		120	614	106,3
		110	524	172,8
		100	414	81,5
		90	302	51,0
		80	274	47,0
		70	242	47,4
		60	211	31,3
		50	175	27,3
		40	132	18,2
		30	83	11,6
		20	32	7,3
	2	250	1055	634,3
		240	970	621,2
		230	957	613,4
		220	913	499,0
		210	880	483,1

à suivre...

TAB5.11 : *suite...*

			MA	
n	m	L	G_{tot}	T_{tot}
		200	841	456,6
		190	795	446,1
		180	795	420,0
		170	768	378,9
		160	726	373,1
		150	690	251,1
		140	627	214,9
		130	567	212,4
		120	539	198,4
		110	492	190,2
		100	412	127,6
		90	302	47,9
		80	269	43,3
		70	242	36,6
		60	211	30,7
		50	175	25,4
		40	132	17,7
		30	83	11,5
		20	32	6,0