Heuristique basée sur la mémoire adaptative pour le problème de tournées de véhicules sélectives

Conference Paper · November 2007

CITATIONS
2

READS
125

3 authors:

Mahdi Khemakhem
Prince Sattam bin Abdulaziz University
47 PUBLICATIONS **144** CITATIONS
SEE PROFILE

Frédéric Semet
École Centrale de Lille
117 PUBLICATIONS **4,455** CITATIONS
SEE PROFILE

Habib Chabchoub
University of Sfax
179 PUBLICATIONS **775** CITATIONS
SEE PROFILE

Heuristique basée sur la mémoire adaptative pour le problème de tournées de véhicules sélectives

Mahdi Khemakhem
LOGIQ
Institut Supérieur de Gestion Industrielle
Sfax, Tunisie.
mahdi.khemakhem@gmail.com

Frédéric Semet
LAMIH-ROI
ISTV Université de Valenciennes
Valenciennes, France
Frederic.semet@univ-valenciennes.fr

Habib Chabchoub
LOGIQ
Institut Supérieur de Gestion Industrielle
Sfax, Tunisie
habib.chabchoub@fsegs.rnu.tn

Résumé - *Nous présentons dans cet article une méthode approchée, basée sur la mémoire adaptative, pour la résolution du Problème de Tournées de Véhicules Sélectives (SVRP). Nous commençons par la définition du problème. Ensuite nous précisons le principe général de notre approche ainsi que ses principales caractéristiques. Notre méthode a montré ses capacités pour résoudre le SVRP et peut être considérée comparable aux heuristiques précédemment proposées dans la littérature.*

Mots clés : Transport, VRP, SVRP, STSP, Heuristique, recherche taboue, mémoire adaptative.

1 Introduction

Dans cet article, nous présentons une heuristique pour la résolution approchée du *Problème de Tournées de Véhicules Sélectives* (PTVS) ou *Selective Vehicle Routing Problem* (SVRP). Le SVRP est une variante du *Problème de Tournées de Véhicules avec Gains* (PTVG). Le SVRP peut être définit comme suit:

Soit $G=(V, E)$ un graphe complet, où $V= \{v_1, v_2,\ldots, v_n\}$ est l'ensemble de n sommets, v_1 et v_n représentent des dépôts. E est l'ensemble des arrêtes de G. Soit $g_i \geq 0$ le gain de chaque sommet v_i de G avec $g_1 = g_n = 0$. Soit d_{ij} les distances reliant chaque deux sommets v_i et v_j de V.

Le SVRP consiste à déterminer un ensemble $\Pi = \{\pi_1,\ldots,\pi_m\}$ de m tournées élémentaires associes a m véhicules contenant chacune les sommets v_1 et v_n comme sommet de départ et sommet d'arrivée. Les m tournées sont identifiées de telle sorte que la distance de chacune ne dépasse pas une longueur limite L et le gain total récolté par l'ensemble des m tournées soit maximisé. Une des caractéristiques du SVRP est donc que tous les sommets peuvent ne pas être visités.

Dans la littérature, le SVRP a été traité de façon approchée par Chao *et al.* en 1996 [1], Hao *et al.* en 2005 [5] et Archetti *et al.* en 2007 [4] sous le nom du "Team Orienteering Problem" (TOP) lorsque deux dépôts différents sont considérés. Un cas particulier du SVRP, où les deux dépôts sont confondus, est traité par Chabchoub *et al.* en 2005 [12,13]. Les premières tentatives de résolution exacte du TOP ont été effectuées par Boussier *et al.* en 2005 [9].

Dans la suite de cet article, nous décrirons le processus général de notre heuristique à mémoire adaptative à la section 2. A la section 3 nous détaillons les différents éléments de l'heuristique proposée, puis nous présentons les résultats numériques obtenus à la section 4. Finalement nous conclurons à la section 5.

2 Description générale de l'heuristique

Notre heuristique est basée sur le concept de la mémoire adaptative. Ce concept a été utilisé fréquemment pour résoudre plusieurs problèmes d'optimisation dont le plus proche du SVRP est le Problème de Tournée de Véhicules avec contraintes de Capacités (CVRP).

2.1 Principe

Notre heuristique suit le principe de la méthode appelée *BoneRoute* proposée par Tarantilis et Kiranoudis pour résoudre le CVRP [11]. Ce principe consiste à stocker, initialement, un ensemble de solutions dans une mémoire, puis à générer à chaque itération une solution en manipulant les solutions stockées dans la mémoire. Cette mémoire est mise à jour, d'une façon dynamique, en introduisant les nouvelles solutions générées. La différence majeure entre la méthode utilisée par Rochat et Taillard [10] et celle utilisée par Tarantilis et Kiranoudis [11], c'est que la première utilise uniquement une mémoire de

solutions de bonnes qualités et la deuxième accepte en plus les solutions de moyennes et de mauvaises qualités. Une grande importance est attribuée à la façon avec la quelle les nouvelles solutions seront générées.

La méthode de génération des nouvelles solutions, utilisée dans *BoneRoute*, est contenue dans son appellation qui est à l'origine du mot *Os* (*Bone* selon la terminologie anglaise) représentant une séquence de sommets (chaîne). En effet, son principe est d'extraire les *Bones* en commun entre les solutions de la mémoire, ensuite, de construire une nouvelle solution en partant de l'ensemble des *Bones* extraits. Ces *Bones* doivent satisfaire deux critères fixés par l'utilisateur a savoir la taille des *Bones* à extraire *TB* (le nombre des sommets de la chaîne) et la fréquence *FB* minimale requise de l'apparition d'une *Bone* dans les solutions de la mémoire.

2.2 Processus général

L'heuristique, proposée dans cet article, suit un processus global décrit dans ce qui suit. Elle commence à générer un ensemble de solutions initiales en utilisant la méthode à deux phases proposée par Chabchoub *et al* pour la résolution du SVRP [12,2]. Ces solutions sont de moyennes qualités, l'heuristique actuelle applique une procédure de recherche taboue sur chaque solution afin d'améliorer d'avantage sa qualité. Les solutions améliorées par la procédure taboue sont triées selon leurs qualités et stockées dans la mémoire des solutions. Les phases déjà mentionnées sont appliquées une seule fois, les prochaines phases seront répétées tant que le critère d'arrêt n'est pas satisfait.

Étape 1: Générer un ensemble de solutions pour le SVRP par la méthode à deux phases proposée par Chabchoub *et al.* [12,2].
Étape 2: Améliorer les solutions générées par une méthode de recherche taboue.
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 *Bones* en communs entre les solutions de la mémoire en respectant les paramètres *TB* et *FB* de taille et de fréquence minimale des *Bones*.
Étape 5: Utiliser la méthode à deux phases pour générer une nouvelle solution en considérant les *Bones* extraites.
Étape 6: Utiliser une méthode de recherche taboue pour améliorer la solution extraite.
Étape 7: Éliminer de la mémoire la solution de mauvaise qualité.
Étape 8: Insérer dans la mémoire la solution générée par l'Étape 6.

Figure 1. Schéma général de l'heuristique à mémoire adaptative

En respectant les paramètres *TB* et *FB*, notre heuristique extrait les *Bones* de taille *TB* qui figurent au moins *FB* fois dans l'ensemble des solutions de la mémoire. En considérant les *Bones* comme étant des sommets du graphe, la méthode à deux phases [12,2] est appliquée pour générer une nouvelle solution au SVRP. Cette solution est améliorée par la procédure de recherche taboue utilisée lors de la génération des solutions initiales. La solution de plus mauvaise qualité, stockée dans la mémoire, sera remplacée par la nouvelle solution même si cette dernière est de mauvaise qualité et à condition qu'elle ne soit pas déjà présente dans la mémoire. La mémoire des solutions sera triée et une nouvelle extraction des *Bones* aura lieu. Le processus de notre heuristique s'arrête dés qu'un nombre d'itérations sans amélioration est atteint. Le processus général de notre heuristique est décrit par la figure 1.

3 Détails de l'heuristique

Dans cette section nous présentons en détail les procédures et les éléments, utilisés dans notre heuristique, déjà mentionnés dans son schéma général.

3.1 Génération des solutions initiales

Comme déjà mentionné, l'heuristique commence par la génération d'un ensemble de *TM* solutions du SVRP pour les stocker dans la mémoire, avec *TM* la taille de la mémoire. Cette phase est assurée par la méthode à deux phases pour la résolution du SVRP proposée par Chabchoub *et al.* [12,2]. 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*-means) étudiée dans un cadre formel par Diday en 1971 [8]. Pour des raisons de performance la méthode de classification utilisée par Chabchoub *et al.* utilise la distance de Mahalanobis comme métrique d'évaluation. Une des caractéristiques de la méthode *k*-means 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.* [3] pour résoudre le Problème du Voyageur de Commerce Sélectif (STSP). En résolvant le STSP sur chaque classe, nous obtenons une solution globale pour le SVRP 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*-means.

3.2 Procédure d'amélioration avec la recherche taboue

Les solutions du SVRP fournies par la méthode à deux phases sont de qualités moyennes. En effet, l'efficacité de la méthode à deux phases a été montrée uniquement pour les problèmes du SVRP à un seul dépôt ou à deux dépôts très proches. Afin de remédier à ce problème, nous avons proposé une procédure basée

sur la recherche taboue pour améliorer les solutions fournies par la méthode à deux phases.

La méthode d'amélioration proposée suit les principes de bases de la recherche taboue. En effet, l'algorithme commence la recherche par la solution initiale Π à l'aide de la méthode à deux phases. Ensuite il initialise les paramètres tels que le nombre d'itérations maximal *iterTMax*, 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 taboue tant que le nombre d'itérations *iter* n'atteint pas *iterTMax*. La procédure taboue 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 considérée comme la meilleure solution rencontrée et le compteur des itérations est 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 par 1 et Π sera remplacée par sa meilleure solution voisine Ω^* même si cette dernière n'améliore pas la meilleure solution rencontrée Π^*. Le schéma général de la procédure d'amélioration est décrit par la figure 2.

Étape 1: Soit la solution Π de gain total $G(\Pi)$ générée par la méthode la méthode à deux phases.
Étape 2: Initialiser

- la meilleure solution rencontrée $\Pi^* = \Pi$,
- le nombre d'itération maximal *iterTMax*,
- fréquences d'aspiration *fa*, de diversification *fd*, d'intensification *fi*,
- la taille *Tl* de la liste tabou *liste*.

Répéter les étapes de 3 à 9 jusqu'à *iter=iterTMax*

Étape 3: Sélectionner le meilleur voisin Ω^* de Π du 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 $\Pi^* := \Omega^*$ et *iter*=-1,
Étape 6: Si *iter* est divisible par *fa* alors appliquer aspiration,
Étape 7: Si *iter* est divisible par *fi* alors appliquer intensification,
Étape 8: Si *iter* est divisible par *fd* alors appliquer diversification,
Étape 9: $\Pi := \Omega^*$, *iter*++.

Figure 2. Schéma général de la procédure d'amélioration avec la recherche taboue

Le passage d'une solution Π à une solution voisine Ω suit une structure de voisinage bien précise. Au début de l'algorithme tabou, une procédure est appliquée une seule fois pour déterminer pour chaque sommet $v_i \in V - \{v_1, v_n\}$ l'ensemble $\widetilde{V}_i$ contenant ses *pp* plus proches sommets. Pour une solution courante $\Pi = \{\pi_1,...,\pi_k,...,\pi_m\}$ un voisin Ω est obtenu en supprimant un sommet v_p de sa tournée π_k puis en libérant un autre sommet v_q de l'ensemble $\widetilde{V}_p$ s'il appartient à une tournée π_h de la solution courante. Après la suppression, une tentative d'amélioration avec échange sera appliquée dans le but d'augmenter les gains des tournées. L'échange est effectué entre les sommets non visités par Π y compris les sommets libérés. Si la permutation aboutit à une augmentation du gain de la tournée, elle est validé et la procédure passe à un autre sommet; sinon, elle n'est pas considérée et la procédure passe à un autre sommet de la tournée courante ou d'une autre tournée si nécessaire. La procédure d'échange est suivie par une autre procédure d'amélioration avec insertion. Elle consiste à essayer d'insérer l'un des sommets restants (non visités) 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 pas la contrainte de longueur et le gain de la tournée actuelle augmente (naturellement dans le cas d'insertion). La libération d'un sommet ainsi que les procédures d'amélioration seront 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 la solution voisine Ω obtenu sera examinée, dans le cas où elle fournit la meilleure solution du voisinage elle sera sauvegardé dans Ω^* comme étant le meilleur voisin de la solution Π et ses deux couples d'attributs $(v_t = v_p, \pi_t = \pi_k)$ et $(v_{t'} = v_q, \pi_{t'} = \pi_h)$ comme étant les sommets v_t et $v_{t'}$ sont libérés (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*. Nous avons développé la liste taboue *liste* sous forme d'une matrice de taille $(n \times m)$. Lors de la déclaration d'un attribut (v_t, π_t) comme un élément tabou $liste(v_t, \pi_t)$ est mis à *iter+Tl*. Tant que $liste(v_t, \pi_t) > iter$ le mouvement défini par l'attribut (v_t, π_t) est considéré tabou. De cette façon la vérification de l'interdiction d'un mouvement est plus rapide que l'utilisation d'une liste FIFO.

A 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 d'échange 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. A chaque *fi* itérations sans amélioration, une procédure d'intensification est appliqué dans le but d'améliorer la solution courante. Cette phase consiste à appliquer sur chaque tournée de la solution courante l'algorithme Genius proposée par Gendreau *et al.* [6] pour résoudre le Problème de Voyageur de Commerce (TSP). L'algorithme Genius permet de corriger, dans la mesure du possible, chaque tournée afin de réduire sa longueur. La réduction des longueurs des tournées peut engendrer l'insertion d'autre sommets qui ne figurent pas dans la solution et d'induire une augmentation du gain total. Après *fd* itérations sans amélioration et lors de l'échec de la procédure d'intensification, une procédure de diversification est appelée. Cette procédure consiste à choisir les deux tournées π_1 et π_2 de gains les plus faibles dans la solution actuelle. Ensuite, il s'agit de couper chaque tournée à son milieu et de reconnecter les chaines générées en appliquant le principe de la recherche locale *2-opt* proposée par Lin [7]. Après cette transformation, si l'une des deux tournées modifiées π_k dépasse la longueur maximale *L*, une procédure de correction de route 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

$$i = \arg\max_j \left\{ \frac{d_{(j-1)j} + d_{j(j+1)} - d_{(j-1)(j+1)}}{g_j}, \forall v_j \in \pi_k \right\}$$

3.3 Extraction des *Bones*

L'extraction de l'ensemble des *Bones* en commun entre les solutions de la mémoire ce 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 *TB* 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 TB sommets existe au moins FB fois dans la mémoire de solution, elle est sélectionnée parmi les *Bones* en commun et la procédure passe au *TB* 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 existe déjà dans la mémoire contenant les solutions et afin d'éviter le problème de cyclage, la procédure d'extraction des *Bones* commence par la deuxième solution de meilleure qualité et non plus par la première.

3.4 Construction d'une nouvelle solution

Après l'extraction des *Bones*, la construction d'une nouvelle solution est assurée par la méthode à deux phases en considérant les *Bones* 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 *Bones* déjà extraits. De ce fait, il faut conserver à la mesure du possible la structure de ces *Bones* dans la nouvelle solution générée. Pour cela et avant d'appliquer la méthode à deux phases, notre heuristique considère les *Bones*, privées de leurs extrémités *Ext1* et *Ext2*, comme étant des sommets notés *vb*. Un nouveau graphe est donc généré (voir figure 3).

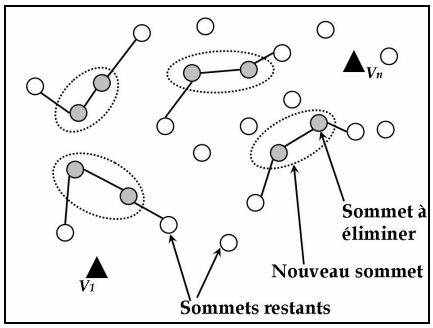

Figure 3. Transformation du graphe

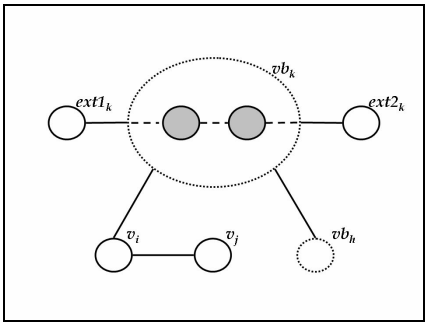

Figure 4. Caractéristiques du graphe transformé

Ce nouveau graphe possède de nouvelles caractéristiques tels que les distances et les gains associés aux anciens et aux nouveaux sommets. Puisque chaque *Bone* 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 *Bones* dans la nouvelle solution, il faut faire de sorte que si l'une des extrémités d'un *Bone* est visité par une tournée, tous les autres sommets d'un *Bone* soient visités. Pour ce faire nous avons défini les nouvelles caractéristiques du nouveau graphe comme suit (voir la figure 4):

- Les distances $d_{v_i v_j}$ entre les sommets non éliminés reste inchangées.
- Les distances $d_{vb_k v_i}$ 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 ($d_{vb_k vi} = d_{vb_k vb_h} = \infty$).

- 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 *Bone* correspondante.
- Les gains des extrémités d'un nouveau sommet vb_k sont annulés ($g_{ext1_k} = g_{ext2_h} = 0$).

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 du à l'introduction de la distance infini et du gain nul. De cette façon, nous conservons la forme des *Bones* 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 : $n' = n + (NB \times (3 - TB))$ avec NB est le nombre des *Bones* extraites. Donc, dans le cas où *TB* est supérieur à 3 la taille du graphe est réduit.

4 Résultats expérimentaux

Après un certains nombre de tests numériques, nous avons fixé les paramètres de notre heuristique comme suit : *TM* :=5, *TB* :=$\lceil n/(5 \times m) \rceil$, *FB* :=2, *Tl* :=$\sqrt{(n \times m)/2}$, *iterTMax* :=5 lors de la génération des solutions initiales et *iterTMax* :=2 lors de l'amélioration de la nouvelle solution générée, *fa :=1*, *fd := iterTMax/2*, *fi :=iterTMax/4, l*e nombre d'itération maximale *iterMax* sans amélioration de l'heuristique est fixé à 10.

Nous avons testé notre heuristique 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.* [1]. Ces instances sont 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-ensemble d'instances est obtenu en variant la longueur maximale *L* d'une tournée. Archetti *et al.* [4] ont proposé quatre heuristiques et ont montré que celles basées sur la recherche à voisinage variable comme étaient les meilleures heuristique. 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.

		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

Tableau 1. Résumé des résultats numériques

Le tableau 1 présente un résumé des résultats numériques fournis par notre méthode. Il présente, pour chaque valeur de *n* fixé, les totaux des gains obtenus par l'approche pour des différentes valeurs de *m* et *L*. Uniquement les valeurs maximales obtenues par "FastVNS" et "SlowVNS" sont considérées. Le tableau 2 présente pour chaque taille de problème le temps d'exécution moyen est maximum correspondant à chaque méthodes. Nous considérons les notations suivantes :

- **SVNS** : Les résultats correspondant à "SolowVNS".
- **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 correspondant
- **T_{max}** : Le temps maximum pour la résolution d'une instance de taille correspondant.
- **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.

FVNS				
n	T_{moy}	Ratio	T_{max}	Ratio
100	22,5	--	121,0	--
102	10,3	--	90,0	--
66	34,2	--	30,0	--
64	8,7	--	20,0	--
33	0,2	--	1,0	--
32	0,1	--	1,0	--
21	0,0	--	0,0	--
SVNS				
n	T_{moy}	Ratio	T_{max}	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	T_{max}	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	--

Tableau 2. Résumé des temps d'exécution CPU

En observant le tableau 1, nous remarquons que pour la majorité des classes d'instances notre approche est meilleure que "FastVNS" et moins bonne que "SlowVNS". En moyenne, la qualité des solutions obtenues par **MA** est à 99.93% des qualités des solutions obtenus 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**, sauf pour quelques exceptions pour les problèmes de petites tailles. Vue la non-conformité des machines où les tests on été effectuée, nous avons utilisée la relation, pour calculer le ratio entre les performances des machines, utilisée par Prins pour évaluer différents algorithmes de résolution du VRPC [14]. Selon le logiciel d'évaluation de performance "*SiSoftware Sandra Standard*", notre machine fonctionne avec une performance P_a=3705 Mflops et celle utilisée pour tester "FastVNS" et "SlowVNS" une performance P_b=3492 Mflops. Nous avons pris "FastVNS" comme méthode de référence et nous avons calculé le ratio pour chaque méthode selon la formule suivante :

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

avec T_a le temps d'exécution de la méthode que nous voulons évaluer et T_b est le temps d'exécution de la méthode de référence "FastVNS".
En tenant compte de ces ratios et malgré la différence entre les machine, **MA** est considérée plus rapide que **SVNS** puisque ces ratios sont nettement très inférieurs à ceux de SVNS. Au pire des cas, notre approche nécessite quatre minutes pour résoudre une instance face à 2 minutes pour **FVNS** et 10 minutes pour **SVNS**.

En fin, nous pouvons dire que notre approche présente un nouveau compromis entre la qualité de solution et le temps d'exécution pour SVRP.

5 Conclusion

Dans cet article nous avons présenté une approche heuristique qui combine une méthode à deux phases et une procédure de recherche taboue au sein d'un processus d'une recherche à mémoire adaptative pour la résolution du problème de tournées de véhicules sélectives.

Expérimentalement, nous avons pu observer que notre approche 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 SVRP.

Références

[1] I.-M. Chao, B.L. Golden, E.A. Wasil: "The Team Orienteering Problem". *European Journal of Operational Research*, vol 8, pp 464-474, 1996.

[2] H. Chabchoub, M. Khemakhem, F. Semet. "Le Problème de Tournées de Véhicules Sélectives: extensions de l'approche basée sur l'algorithme k-means", *Metaheuristiquse*, Hammamet, Tunisie, novembre, 2006.

[3] M. Gendreau, G. Laporte, F. Semet : "A Tabu Search Heuristic for the Undirected Selective Travelling Salesman Problem", *European Journal of Operational Research* vol 106, pp 539-545, 1998.

[4] C. Archetti, A. Hertz, M.G. Speranza : "Metaheuristics for the Team Orienteering Problem", *Journal of Heuristics,* vol 13, pp 49-76*,* 2007.

[5] H. Tang, E. Miller-Hooks : "A Tabu Search Heuristic for the Team Orienteering Problem", *Computers and Operation Research*, vol 32, pp 1379-1407, 2005.

[6] M. Gendreau, A. Hertz, G. Laporte : "New insertion and postoptimization procedures for the travelling salesman problem". *Operations Research*, vol 40, pp 1086-1094, 1992.

[7] S. Lin : "Computer solutions of the travelling salesman problem", *Bell System Computer Journal*, vol 44, pp 2245-2269, 1965.

[8] E. Diday: "La méthode des nuées dynamiques", *Revue de Statistique Appliquée*, vol.19, pp 19-34, 1971.

[9] S. Boussier, D Feillet, M. Gendreau: "An exact algorithm for Team Orienteering Problems" *Rapport technique LIA-2005,* Laboratoire d'Informatique d'Avignon.

[10] Y. Rochat, E. Taillard: "Probabilistic diversification and intensification in local search for vehicle routing", *Journal of Heuristics,* vol 1, pp 147-167, 1995.

[11] C.D. Tarantilis, C.T. Kiranoudis: " BoneRoute: An Adaptive Memory-Based Method for Effective Fleet Management ", *Annals of Operations Research*, vol 115, pp 227-241, 2002.

[12] H. Chabchoub, M. Khemakhem, F. Semet, M. Tmar: "Le problème d'élaboration de *m*-tournées sélectives: une approche basée sur la méthodes des centres mobiles" *Logistique & Transport*, Hammamet, Tunisie, mai, 2006.

[13] H. Chabchoub, M. Khemakhem, F. Semet : "A Hybrid Heuristic for the Selective Vehicle Routing Problem", *6th Metaheuristics International Conference*, *MIC'05*, August 22-26,Vienna, 2005

[14] C. Prins: "A simple and effective evolutionary algorithm for the vehicle routing problem", *Computers & Operations Research, vol* 31, pp 1985-2002, 2004.