Université de Montréal

Recherche tabou pour un problème de tournées de véhicules avec une flotte privée et un transporteur externe

par
Marc-André Naud

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

Décembre 2008

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé :

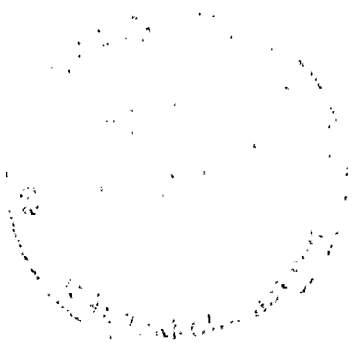
Recherche tabou pour un problème de tournées de véhicules avec une flotte privée et un transporteur externe

présenté par :

Marc-André Naud

a été évalué par un jury composé des personnes suivantes :

Jacques Ferland, président-rapporteur
Jean-Yves Potvin, directeur de recherche
Bernard Gendron, membre du jury



Sommaire

Dans ce mémoire, nous nous intéressons à un problème où le propriétaire d'une flotte privée de véhicules ne peut satisfaire toute la demande de transport à laquelle il doit faire face. Il doit donc recourir aux services d'un transporteur externe pour satisfaire la demande excédentaire, moyennant un certain coût.

Le travail rapporté dans ce mémoire consiste à améliorer un algorithme de recherche tabou générant des tournées pour la flotte interne afin de minimiser le coût de ces tournées, plus le coût du recours au transporteur externe pour les clients non servis. Ceci est réalisé en modifiant la procédure de création de la solution initiale et en implémentant une structure de voisinage basée sur des chaînes d'éjection. Dans les problèmes considérés, la flotte de véhicules peut être homogène ou hétérogène, ce qui rend le problème encore plus difficile.

Les résultats obtenus par la recherche tabou sont comparés à d'autres approches rapportées dans la littérature sur des problèmes tests standards.

Mots clés : Tournées de véhicules, flotte privée, transporteur externe, recherche tabou, chaînes d'éjection.

Abstract

In the following, we consider the problem faced by the owner of a private fleet of vehicles who cannot satisfy all the demand that needs to be fulfilled. Thus, he must use an external common carrier to satisfy the excess demand at an additional cost.

The work reported here consists in improving a previously developed tabu search heuristic that minimizes the cost incurred by the use of the internal fleet plus the cost of the external common carrier for unserved customers. This improvement is realized by modifiying the procedure for creating the initial solution and by developing a neighborhood structure based on ejection chains. In the problems considered, the internal fleet is either homogeneous or heterogeneous, which makes the problem harder to solve.

The results obtained by the tabu search heuristic are compared with other approaches reported in the litterature on standard test problems.

Keywords : Vehicle routing, private fleet, common carrier, tabu search, ejection chains.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Les problèmes de tournées de véhicules constituent une classe très vaste de problèmes présentant des caractéristiques diverses. Le problème de base est le problème de tournées avec contraintes de capacité, ou capacitated Vehicle Routing Problem (*VRP*) en anglais. Dans ce problème, une flotte de véhicules homogène de capacité finie doit desservir à coût minimal un ensemble de clients ayant chacun une demande finie. Pour ce faire, chaque véhicle réalise une tournée unique qui commence et se termine à un dépôt central. Le coût à minimiser est habituellement la distance totale parcourue par les véhicules. Le problème du voyageur de commerce, ou Traveling Salesman Problem (*TSP*) en anglais, peut être vu comme un cas particulier du *VRP* où l'on dispose d'un seul véhicule de capacité infinie, qui est en mesure de desservir tous les clients. Comme le *TSP* est un problème *NP-complet*, il en est de même pour le VRP, ce qui justifie l'abondance et la variété d'heuristiques ou métaheuristiques qui s'attaquent à ce problème.

Depuis l'introduction du *VRP* par Dantzig [11], plusieurs méthodes heuristiques ont été développées afin de résoudre ce problème (e.g., l'heuristique classique des savings de Clarke et Wright [8]). Une heuristique est une méthode d'approximation qui permet de trouver une bonne solution à un problème dans des temps de calcul raisonnables (ce qui n'est pas le cas pour les

méthodes exactes, surtout lorsque le problème est de taille conséquente). Par contre, une heuristique ne donne aucune garantie quant à l'obtention d'une solution optimale ou même d'une solution de qualité donnée.

Par la suite sont apparues les métaheuristiques, qui ont permis d'améliorer grandement les résultats obtenus. Une métaheuristique correspond à une stratégie de recherche qui guide différentes heuristiques plus simples dans l'espace des solutions. Les métaheuristiques possèdent généralement un mécanisme leur permettant de s'échapper des optima locaux. Ceci leur permet d'explorer une plus grande portion de l'espace des solutions qu'une méthode de descente pure, par exemple. D'un autre côté, cette recherche plus étendue, où la transition d'une solution à une autre n'est pas nécessairement améliorante, mène au danger de cycler dans l'espace des solutions et il faut donc prévoir des dispositifs permettant d'empêcher (autant que possible) un tel comportement. Enfin, un critère d'arrêt doit être défini, par exemple un nombre fixe d'itérations ou un nombre maximal d'itérations consécutives sans amélioration de la meilleure solution rencontrée, car la méthode ne s'arrête pas au premier optimum local atteint. Une faiblesse relative des métaheuristiques est leur vitesse de calcul plutôt lente. Heureusement, avec la puissance des ordinateurs qui augmente, ce défaut est de moins en moins apparent.

Dans Gendreau, Laporte et Potvin [19], on mentionne six différentes métaheuristiques pour résoudre le *VRP*, soit (1) le recuit simulé, (2) le recuit déterministe, (3) les algorithmes génétiques, (4) les algorithmes de colonies de fourmis, (5) les réseaux de neurones et (6) la recherche tabou, cette dernière étant la métaheuristique utilisée dans ce travail.

La mécanique de base d'une recherche tabou est assez semblable à celle du recuit simulé et repose sur la notion de voisinage de la solution courante. Essentiellement, un voisinage de solutions est obtenu à l'aide d'une classe de transformations locales qui sont appliquées à la solution courante. Par contre, au lieu de choisir la prochaine solution aléatoirement dans le voisinage comme le fait le recuit simulé, la recherche tabou choisit la meilleure

solution dans le voisinage qui n'est pas tabou. Ici, une solution est tabou si les attributs de la transformation locale menant à cette solution ont été appliqués récemment au cours de la recherche. Par exemple, dans le cas du VRP, une transformation ou un mouvement peut correspondre à déplacer un client d'une route dans une autre. Dans un tel cas, l'attribut associé au mouvement est un couple où le premier élément est le client et le deuxième la route d'origine du client. Il sera donc interdit, ou tabou, pendant un nombre θ d'itérations (durée tabou) de replacer ce client dans sa route d'origine, ceci afin d'éviter de générer à nouveau la même solution et cycler. La durée tabou peut être fixe ou varier aléatoirement dans une plage de valeurs entières.

Une exception à la règle définissant le statut tabou d'une solution est le critère d'aspiration. Si le critère d'aspiration est satisfait, le statut tabou est ignoré et le mouvement correspondant peut être effectué. Dans presque tous les cas, le critère d'aspiration consiste à accepter toute solution qui est meilleure que la meilleure solution rencontrée jusqu'à date, même si cette nouvelle solution est tabou. En effet, dans un tel cas, il n'y aucun risque de cycler !

Voici maintenant une description d'une recherche tabou simple en pseudo-code pour un problème de minimisation :

1. Générer une solution initiale s. Initialiser $s^* \leftarrow s$ et $k \leftarrow 0$;
2. $k \leftarrow k + 1$;
3. Générer un sous-ensemble $M(s)$ du voisinage $N(s)$ tel que $s' \in M(s)$ si et seulement si (s' n'est pas tabou) ou (s' satisfait les conditions d'aspiration) ;
4. Soit $s^k \in M(s)$ tel que $f(s^k)$ est minimal dans $M(s)$;
5. $s \leftarrow s^k$;
6. Si $f(s^k) < f(s^*)$, alors $s^* \leftarrow s^k$;
7. Mettre à jour les statuts tabou ainsi que les conditions d'aspiration ;
8. Si le critère d'arrêt n'est pas satisfait, retourner à l'étape 2 ; sinon, arrêter et retourner s^*.

Dans cette description, s correspond à la solution courante, s^* est la meilleure solution rencontrée et f est la fonction objectif. Comme on peut le voir, il n'est pas nécessaire de considérer le voisinage complet d'une solution (en particulier si celui-ci est de taille considérable), mais on peut se limiter à un sous-voisinage choisi soit aléatoirement, soit à l'aide de règles particulières.

Dans ce mémoire, nous appliquons la recherche tabou à une variante du VRP où l'on peut faire appel à un transporteur externe pour desservir des clients si la flotte interne ne peut les accomoder, mais à un coût généralement plus élevé. Ce problème est appelé Vehicle Routing Problem with Private fleet and Common carrier (*VRPPC*) dans la littérature. Nous considérons aussi la variante hétérogène du problème où les caractéristiques des véhicules ne sont pas toutes identiques. L'apport majeur de ce travail est l'ajout d'une structure de voisinage basée sur des chaînes d'éjection à un algorithme de recherche tabou précédemment rapporté dans Côté et Potvin [10], afin d'obtenir des résultats compétitifs à la fois pour les problèmes homogènes et hétérogènes.

Dans la suite du travail, le chapitre 2 propose une revue de la littérature sur les applications de la recherche tabou au *VRP* et au *VRPPC*. Au chapitre 3, une modélisation de notre problème est proposée. Au chapitre 4, nous présentons une description de l'algorithme de recherche tabou original [10]. Les modifications apportées à cet algorithme, qui représentent le coeur de ce travail, sont l'objet du chapitre 5. Des résultats expérimentaux sur des problèmes tests standards sont rapportés au chapitre 6. Enfin, une conclusion suit.

Chapitre 2

Revue de la littérature

Dans ce chapitre, nous survolons d'abord rapidement les applications de la recherche tabou permettant de résoudre le problème classique de tournées de véhicules avec contraintes de capacité. Ce survol de la littérature s'inspire de celui de Gendreau, Laporte et Povin [19]. Nous nous penchons ensuite sur les problèmes pour lesquels un transporteur externe est disponible.

2.1 Recherche tabou pour le problème de tournées de véhicules

La recherche tabou a été utilisée abondamment pour résoudre le problème de tournées de véhicules. Les premières mises en oeuvre de la recherche tabou dans un contexte de tournées de véhicules se retrouvent dans Willard [47], où l'on définit le voisinage comme étant les solutions obtenues en effectuant des échanges d'arêtes de type 2-opt et 3-opt [29, 30], ainsi que dans Pureza et Franca [32], où le voisinage correspond à des réinsertions et des échanges de sommets entre les routes. Comme il s'agit des premières applications de la recherche tabou pour le *VRP*, ces deux algorithmes n'ont pas donné des résultats significativement supérieurs à ceux obtenus avec les meilleures heuristiques de l'époque.

Ensuite, Osman [31] développe les mouvements de type λ-*interchange* qui généralisent les réinsertions d'un sommet et échanges de deux sommets en déplaçant de 0 à λ sommets dans chacune des routes impliquées.

C'est dans le travail de Gendreau, Hertz et Laporte [17] que l'algorithme *Taburoute* est introduit. Cet algorithme a permis de rendre opérationnels certains des concepts introduits par Glover [21, 22], en plus d'en introduire de nouveaux afin d'améliorer les performances de la recherche tabou. Parmi ces concepts, mentionnons : (1) l'ajout de termes de pénalité pour non respect des contraintes lors de l'évaluation du coût de la solution, permettant ainsi à la recherche de considérer des solutions non réalisables ; de plus, ces termes de pénalité sont ajustés automatiquement par l'algorithme en fonction des résultats de la recherche ; (2) la génération de plusieurs solutions initiales différentes à partir desquelles on effectue une recherche tabou préliminaire ; la recherche tabou complète est alors appliquée à la meilleure solution obtenue ; (3) l'introduction d'un mécanisme de diversification continue où la fonction objectif est biaisée afin de favoriser les solutions contenant des éléments qui ont été peu impliqués dans les mouvements précédents ; (4) l'introduction d'une durée tabou θ choisie aléatoirement dans une plage de valeurs, au lieu d'une valeur fixe. *Taburoute* a produit plusieurs des meilleures solutions connues à l'époque pour les problèmes tests de Christofides, Mingozzi et Toth [6].

L'algorithme de Taillard [41] contient plusieurs similarités avec *Taburoute*, sauf que la recherche se restreint à l'ensemble des solutions réalisables. La principale innovation consiste à partitionner le problème en sous-problèmes de taille plus petite, sur lesquels une recherche tabou individuelle est appliquée. La recherche tabou développée par Xu et Kelly [48] se démarque en proposant une structure de voisinage plus complexe qui se formule sous la forme de problèmes de flot dans des réseaux. Dans Rego et Roucairol [36], une structure de voisinage de type chaîne d'éjection est proposée (plus de détails sont fournis sur les chaînes d'éjection à la section 5.3). L'am-

pleur du voisinage n'est toutefois pas aussi considérable que dans le présent mémoire.

Plusieurs travaux ont aussi fait appel à l'intégration d'une mémoire adaptative au sein de la recherche tabou comme ceux de Taillard *et al.* et Tarantilis [43, 44]. Cette approche, appelée parfois Adaptive Memory Programming (*AMP*), a été proposée pour la première fois par Rochat et Taillard [38] et consiste à garder en mémoire les meilleures solutions rencontrées au cours de la recherche. De nouvelles solutions initiales sont alors générées pour la recherche tabou en juxtaposant des routes provenant de différentes solutions dans la mémoire. Évidemment, il faut s'assurer de produire une solution admissible lors du processus de juxtaposition. Rochat et Taillard [38] ont également ajouté des éléments d'oscillation stratégique à leur recherche tabou.

La recherche granulaire, ou Granular Tabu Search (*GTS*) en anglais, est un concept très intéressant qui a été introduit par Toth et Vigo [45]. L'idée principale provient de l'observation que les arcs avec un grand coût ont peu de chance d'appartenir à une solution de bonne qualité. Les arcs ayant un coût supérieur à une certaine valeur sont donc éliminés, ce qui réduit considérablement la taille du graphe et permet de diminuer les temps de calcul.

2.2 Problèmes de tournées avec flotte hétérogène

Puisque la recherche tabou décrite dans ce texte résout la variante du *VRPPC* avec flotte hétérogène, nous allons présenter ici quelques travaux qui se sont attaqués à des problèmes de tournées avec flotte hétérogène.

Taillard [42] propose d'abord une approche de type *AMP*. Pour chaque type de véhicules, l'algorithme résout d'abord le sous-problème de tournées avec flotte homogène. Des solutions au problème avec flotte hétérogène sont alors générées en combinant les routes obtenues pour les problèmes homogènes. Dans Gendreau *et al.* [18], le problème est résolu en utilisant un al-

gorithme de recherche tabou assez complexe, faisant également appel à une mémoire adaptative. Parmi les stratégies utilisées, on retrouve la création de solutions initiales à l'aide d'un algorithme de balayage, la possibilité d'explorer le domaine non réalisable et la diversification continue qui pénalise les solutions contenant des éléments fréquemment impliqués dans les mouvements précédents. Li, Golden et Wasil [28] utilisent une variante déterministe du recuit simulé appelée *Record-to-record travel*. La méthode consiste à se déplacer dans le voisinage de la solution courante tant qu'il y a amélioration de façon à obtenir une nouvelle solution record de nature locale et, par la suite, à perturber cette nouvelle solution. Si la solution perturbée est suffisamment proche du record global, elle devient alors le nouveau record global. Cette approche à deux niveaux permet à l'algorithme d'échapper plus facilement aux nombreux minima locaux de l'espace des solutions, minima locaux qui sont nombreux à cause de la complexité du problème avec flotte hétérogène.

2.3 *VRPPC*

Le *VRPPC* est une extension du problème classique de tournées de véhicules. Le travail rapporté dans Ball *et al.* [1] fut le premier à proposer une approche de résolution pour un problème avec transporteur externe. Cependant, il s'agissait d'un problème de type cueillette et livraison, et la taille de la flotte privée constituait une variable du problème.

Volgenant et Jonker [46] ont étudié un problème de cueillette seulement avec transporteur externe, mais pour une flotte privée contenant un seul véhicule. Par la suite, il a été montré par Diabi et Ramesh [12] que ce problème pouvait être résolu exactement pour des instances contenant jusqu'à 200 clients, à l'aide d'un algorithme de séparation et évaluation progressive (branch-and-bound). Klincewicz, Luss et Pilcher [25] ont ensuite étudié le problème avec plusieurs véhicules et ont introduit une heuristique qui consiste à classer les clients à desservir en différents secteurs et, par la suite, à optimiser chaque secteur indépendemment les uns des autres, ce qui réduit

les temps de calcul.

Ce n'est pas avant le travail de Chu [7] qu'une formulation mathématique du *VRPPC* est apparue. Cette formulation n'a pas été vraiment exploitée, car le problème est ensuite résolu à l'aide d'une heuristique basée sur la méthode des gains (savings) de Clarke et Wright [8]. De plus, des échanges intra- et inter-routes sont proposés. Des résultats supérieurs à ceux rapportés dans Chu [7] ont par la suite été obtenus par Bolduc, Boctor et Renaud [2] avec une méthode très rapide appelée *SRI*, pour *Selection, Routing and Improvement.* Cette heuristique exploite notamment deux solutions initiales différentes afin de diversifier les résultats. Les mêmes auteurs ont ensuite apporté des améliorations à cette heuristique en l'intégrant dans un cadre métaheuristique [3]. Cette approche, appelée *RIP* pour *Randomized construction - Improvement - Perturbation*, intègre des mécanismes de perturbation dans la phase de construction de la solution initiale ainsi qu'à chaque itération de la phase d'amélioration. Cette métaheuristique comprend les heuristiques suivantes : (1) une variante de l'heuristique de Clarke et Wright [8] qui perturbe aléatoirement les valeurs de la matrice de coût en suivant le modèle de Li, Golden et Wasil [27] ; (2) une procédure d'amélioration de type *4-opt** [37] ; (3) une variante des λ-*interchanges* [31] avec $\lambda = 2$, variante qui n'évalue qu'une partie des échanges possibles afin d'augmenter la vitesse d'exécution tout en conservant pratiquement la même qualité de solution ; (4) une procédure qui permet les échanges entre les véhicules de la flotte privée et le transporteur externe ; (5) un mécanisme de perturbation qui consiste à modifier les types de véhicules associés à des routes choisies au hasard.

Finalement, Côté et Potvin [10] présentent des résultats supérieurs à ceux rapportés par Bolduc *et al.* [3], mais pour des problèmes avec flotte homogène seulement, en faisant appel à une recherche tabou (que nous appellerons RT dans la suite). Cet algorithme est décrit de façon précise au chapitre 4. Il faut noter que le travail présenté dans ce mémoire consiste à apporter des améliorations à RT afin d'obtenir des résultats compétitifs autant pour les

problèmes avec flotte homogène qu'avec flotte hétérogène. Les améliorations se situent principalement au niveau de la procédure de construction de la solution initiale, ainsi qu'au niveau de la structure de voisinage utilisée au sein de la recherche tabou. Dans ce dernier cas, un voisinage basé sur les chaînes d'éjection est proposé.

Chapitre 3

Formulation mathématique du problème

Soit $G = (V^0, A)$ un graphe orienté avec V^0 l'ensemble des sommets, incluant le dépôt v_0, et A l'ensemble des arcs reliant chaque paire de sommets distincts entre eux. Ainsi, $V = V^0/\{v_0\} = \{v_1, v_2, \ldots, v_n\}$ est l'ensemble des n clients. On définit les coûts de déplacement c_{ij} du sommet v_i au sommet v_j comme étant la distance entre ces sommets, pour $v_i, v_j \in V^0$, $i \neq j$. Chaque sommet $v_i \in V$ a une demande q_i et un coût e_i si le client est desservi par le transporteur externe. Une flotte privée de m véhicules est disponible où chaque véhicule $d_k \in \{d_1, d_2, \ldots, d_m\}$ a une capacité Q_k et un coût fixe f_k lorsque ce véhicule est utilisé pour desservir au moins un client. Le but du *VRPPC* est de trouver une solution qui satisfait les quatres conditions suivantes :

- chaque véhicule de la flotte privée dessert au plus une route qui débute et finit au dépôt ;
- chaque client est visité exactement une fois par un véhicule de la flotte privée ou par le transporteur externe ;
- la demande totale de chaque route de la flotte de véhicules privée ne peut dépasser la capacité maximale Q_k du véhicule d_k qui dessert cette

route ;

- la somme des coûts de déplacements, des coûts fixes des véhicules et des coûts encourus par l'utilisation du transporteur externe est minimisée.

Afin de présenter le modéle mathématique du *VRPPC*, il faut d'abord introduire quatre types de variables :

- x_{ij}^k est 1 si le véhicule d_k visite v_j immédiatement après v_i, 0 sinon ; $i, j = 0, \ldots, n$, $i \neq j$; $k = 1, \ldots, m$;
- y_i^k est 1 si le véhicule d_k visite v_i, 0 sinon ; $i = 0, 1, \ldots, n$; $k = 1, \ldots, m$;
- z_i est 1 si le client v_i est affecté au transporteur externe, 0 sinon ; $i = 1, \ldots, n$;
- $u_i^k \geq 0$ est une borne supérieure sur la charge du véhicule d_k après avoir servi le client v_i ; $i = 1, \ldots, n$; $k = 1, \ldots, m$.

On a donc :

$$\min \sum_{k=1}^{m} f_k y_0^k + \sum_{i=0}^{n} \sum_{\substack{j=0 \\ i \neq j}}^{n} \sum c_{ij}^k x_{ij}^k + \sum_{i=1}^{n} e_i z_i \tag{3.1}$$

$$\text{s.à.} \sum_{j=1}^{n} \sum_{k=1}^{m} x_{0j}^k = \sum_{i=1}^{n} \sum_{k=1}^{m} x_{i0}^k \leq m \tag{3.2}$$

$$\sum_{\substack{j=0 \\ j \neq h}}^{n} x_{hj}^k = \sum_{\substack{i=0 \\ i \neq h}}^{n} x_{ih}^k = y_h^k \qquad h = 0, 1, \ldots, n; k = 1, \ldots, m \tag{3.3}$$

$$\sum_{k=1}^{m} y_i^k + z_i = 1 \qquad i = 1, \ldots, n \tag{3.4}$$

$$\sum_{i=1}^{n} q_i y_i^k \leq Q_k \qquad k = 1, \ldots, m \tag{3.5}$$

$$u_i^k - u_j^k + Q_k x_{ij}^k \leq Q_k - q_i \qquad i, j = 1, \ldots, n, i \neq j; k = 1, \ldots, m \tag{3.6}$$

Dans ce modèle, l'équation (3.1) est la fonction objectif. La contrainte (3.2) assure qu'il n'y a pas plus de routes qu'il n'y a de véhicules dans la flotte interne. La contrainte (3.3) assure que le véhicule qui entre dans un sommet est nécessairement celui qui sort du sommet en plus d'établir un lien entre les variable x_{ij}^k et y_i^k. La contrainte (3.4) oblige chaque client à

être desservi soit par un véhicule de la flotte interne soit par le transporteur externe. La contrainte (3.5) voit à ce que la capacité des véhicules ne soit pas dépassée. Finalement, (3.6) permet d'éliminer les sous-tours qui ne sont pas connectés au dépôt. En effet, si le tour d'un certain véhicule d_k, disons $(v_{\pi_1}, v_{\pi_2}, ..., v_{\pi_{\ell-1}})$, ne contient pas le dépôt, alors pour chacun des clients dans ce tour, nous avons (en posant $v_{\pi_\ell} = v_{\pi_1}$) :

$$u^k_{\pi_i} - u^k_{\pi_{i+1}} + Q_k x^k_{\pi_i \pi_{i+1}} \leq Q_k - q_{\pi_i}$$

ou encore, puisque $x^k_{\pi_i \pi_{i+1}} = 1$, $i = 1, ..., \ell - 1$,

$$u^k_{\pi_i} - u^k_{\pi_{i+1}} \leq -q_{\pi_i}.$$

en sommant ces inégalités sur tout les i, on obtient :

$$0 \leq -(q_{\pi_1} + q_{\pi_2} + ... + q_{\pi_{l-1}})$$

une contradiction. Il faut noter que ces contraintes d'élimination de sous-tours pour le VRP ont été introduites pour la première fois par Kulkarni et Bhave [26].

Chapitre 4

Description de l'algorithme original

Ce chapitre décrit la recherche tabou originale RT, telle que rapportée par Côté et Potvin [10], qui n'a été appliquée qu'à des problèmes où la flotte de véhicules est homogène. Nous supposons donc ici que $Q_k = Q$, k =1, 2, ... m. Il faut aussi noter que le véhicule d_0 représente le transporteur externe.

4.1 Procédure de création de la solution initiale

L'un des points cruciaux du problème consiste à choisir adéquatement les clients qui seront desservis par le transporteur externe. La procédure de création de la solution initiale commence donc par allouer une partie des clients au transporteur externe afin de satisfaire la demande excédentaire (par rapport à la capacité totale de la flotte interne). Par la suite, on crée les routes pour les véhicules de la flotte interne avec les clients qui restent. Il est à noter que l'ordre des clients est défini par leur indice.

1. Si $\sum_{i=1}^{n} q_i > mQ$, alors :
   - Allouer les h premiers clients au transporteur externe, où h est tel

que

$$\sum_{i=1}^{h} q_i \geq \sum_{i=1}^{n} q_i - mQ \geq \sum_{i=1}^{h-1} q_i$$

2. Créer les routes à partir des clients restants en utilisant une heuristique d'insertion à coût minimal ;
3. Tant qu'il y a amélioration :
   3.1 Appliquer une méthode de descente locale basée sur la structure de voisinage présentée à la section 4.2.2 (et qui s'inspire de celle utilisée dans le Unified Tabu Search (UTS) de Cordeau, Laporte et Mercier [9]) ;
   3.2 Appliquer une méthode de descente locale de type *4-opt** [37] à chacune des routes dans la solution.

4.2 Recherche tabou

4.2.1 Attributs d'une solution

On définit d'abord l'ensemble des attributs d'une solution s, soit $B(s) = \{(i,k)|\ i = 1,\ldots,n; k = 0,\ldots,m$ et le client v_i est desservi par le véhicule $d_k\}$. Comme on le verra à la section suivante, la modification apportée par un mouvement consiste simplement à ajouter ou retrancher des attributs dans cet ensemble.

4.2.2 Structure de voisinage

La structure de voisinage qui est définie pour cette recherche tabou contient deux types de mouvements [9] :

- Déplacer un client d'une route à une autre (incluant le transporteur externe)
- Échanger deux clients dans deux routes différentes, en insérant les clients à la meilleure position possible dans leur nouvelle route.

4.2.3 L'espace des solutions admissibles

L'espace des solutions admet les solutions non réalisables. Afin de pouvoir comparer les solutions réalisables et les solutions non réalisables, on introduit une pénalité pour la violation des contraintes de capacité. Dans le cas d'une solution quelconque s, cette pénalité correspond à :

$$q(s) = \sum_{k=1}^{m} \left[\sum_{i=1}^{n} q_i y_i^k - Q_k \right]^+$$

où $[x]^+ = \max\{0, x\}$. L'évaluation de la solution correspond alors à $g(s) = f(s) + \alpha q(s)$, où $f(s)$ est la fonction objectif originale et $\alpha > 0$ est un paramètre de pondération.

La pénalité de non réalisabilité est ajustée au cours de l'algorithme via un autre paramètre $\delta > 0$. En effet, le paramètre α est modifié à chaque itération de la façon suivante :

-Si la solution courante n'est pas réalisable, alors $\alpha \leftarrow \alpha(1+\delta)$. Ainsi, on augmente la pénalité, afin de forcer un retour vers le domaine réalisable.

-Si la solution courante est réalisable, alors $\alpha \leftarrow \frac{\alpha}{1+\delta}$. À l'opposé, on réduit la pénalité lorsque le domaine réalisable est atteint.

4.2.4 Critère d'aspiration

Le critère d'aspiration, qui permet d'ignorer le statut tabou d'une solution dans le voisinage de la solution courante, est le suivant :

- La nouvelle solution est réalisable, et ;
- Le coût de cette nouvelle solution est inférieur à celui de la meilleure solution rencontrée contenant au moins un des attributs impliqués dans le mouvement courant.

Le coût de la meilleure solution rencontrée contenant l'attribut (i, k) s'appelle le *niveau d'aspiration* de l'attribut (i, k). On dénote ce niveau d'aspiration

par σ_i^k (avec σ_i^k fixé à une valeur arbitrairement grande au départ).

4.2.5 La durée tabou

Au cours de la recherche, chaque attribut est associé à une durée tabou qui varie lorsque cet attribut est retiré de la solution. Ainsi, si un attribut (i,k) est retiré de la solution, alors l'attribut (i,k) sera tabou pour les θ prochaines itérations ou θ est choisi de façon aléatoire dans une certaine plage de valeurs. Au niveau de la notation, la dernière itération pour laquelle l'attribut (i,k) est interdit est dénotée par τ_i^k.

4.2.6 Diversification

La recherche tabou diversifie la recherche en pénalisant les attributs qui sont souvent impliqués dans les mouvements effectués lors de la recherche. Soit ρ_i^k le nombre de fois que l'attribut (i,k) a été ajouté à une solution au cours de la recherche et soit ρ_{min} le minimum des ρ_i^k pour les attributs qui sont impliqués dans le mouvement. Alors, la pénalité est proportionnelle au produit de $f(s)$, ρ_{min} et $\sqrt{nm}$, le tout divisé par le numéro de l'itération courante λ. Le paramètre γ permet ensuite d'ajuster l'importance de cette pénalité. Il est à noter que, même si ρ_{min} tend à diminuer avec la taille du problème, le facteur $\sqrt{nm}$ compense pour cette diminution. Pendant la recherche tabou, on favorisera donc des solutions qui sont différentes des solutions précédemment rencontrées. Au total, l'évaluation d'une solution s correspond donc à $h(s)$ où :

$$h(s) = g(s) + \gamma\sqrt{nm}f(s)\frac{\rho_{min}}{\lambda}$$

4.2.7 Critère d'arrêt

La recherche est arrêtée après un nombre fixe d'itérations.

4.2.8 Redémarrages

Grâce à la présence d'éléments aléatoires dans cette recherche tabou, celle-ci est exécutée plusieurs fois afin d'obtenir des résultats différents. Le résultat final est la meilleure solution trouvée pour toutes les exécutions.

4.3 Valeurs des paramètres

Après une optimisation des différents paramètres de la recherche tabou, les paramètres ont été fixés aux valeurs suivantes par Côté et Potvin [10] :

- le nombre total d'itérations est de 10 000 ou de 25 000, dépendant des besoins qualité / temps ;
- la durée tabou est pigée aléatoirement dans la plage de nombres entiers $[7, 14]$;
- le paramètre de diversification prend la valeur $\gamma = 0.01$;
- le paramètre d'ajustement de la pénalité pour non réalisabilité est fixé à la valeur $\delta = 1.25$;
- le nombre de redémarrages est fixé à 10.

Chapitre 5

Modifications apportées à l'algorithme

Le coeur de notre travail se situe dans ce chapitre. Nous décrivons ci-dessous les améliorations apportées à l'algorithme de recherche tabou original *RT* [10]. Nous allons donc dénoter par *RTE* ce nouvel algorithme, pour *Recherche Tabou Étendue.*

5.1 Structure de données utilisée pour la représentation des routes

Plusieurs structures de données peuvent être utilisées afin de représenter chacune des routes du problème, comme les tableaux, les listes chaînées ou les arbres [15]. Cependant, comme chaque route dans notre problème contient relativement peu de clients (bien moins que la limite de 1000 suggérée par Fredman [15]), les tableaux ont été retenus. En somme, nous avons maintenu la structure de données de l'algorithme original *RT*.

5.2 Procédure de création de la solution initiale

Puisque la recherche tabou est autorisée à redémarrer plusieurs fois, nous avons favorisé une procédure de création de la solution initiale de nature stochastique plutôt que déterministe. En effet, la présence d'éléments stochastiques permet de générer des points de départ différents pour la recherche tabou et ainsi de mieux échantillonner l'espace des solutions.

Deux procédures de création sont examinées. La première est une simple procédure d'insertion à coût minimal, et la seconde s'inspire de la phase 1 de l'heuristique *I3* [37], où une enveloppe convexe est construite. Pour les deux approches, nous avons créé une version déterministe et une version stochastique en introduisant des paramètres de perturbation qui affectent certains choix faits en cours d'algorithme. Ces deux procédures de création de la solution initiale sont décrites dans la suite.

5.2.1 Le prétraitement

Bien que les deux méthodes de création de la solution initiale soient différentes, il y a toutefois un prétraitement qui leur est commun. Ce prétraitement, qui consiste à allouer une partie des clients correspondants à la demande excédentaire, est similaire à celui proposé dans Côté et Potvin [10], mais une modification importante y a été apportée. En effet, plutôt que de traiter les sommets en ordre croissant de leur indice, l'allocation est réalisée en triant les clients en ordre croissant du ratio du coût du tranporteur externe sur leur demande. Cette valeur est en fait le coût du transport externe par unité de demande. Bien sûr, on est intéressé à allouer au transporteur externe les clients dont le coût par unité de demande est le plus bas.

En pseudo-code, la procédure de prétraitement est la suivante :

- Si $\sum_{i=1}^{n} q_i > \sum_{k=1}^{m} Q_k$, alors :
  - Trier les clients en ordre croissant, par rapport au ratio $\frac{e_i}{q_i}$, $i = 1, \ldots, n$;
  - Allouer les h premiers clients au transporteur externe, où h est tel

que :

$$\sum_{i=1}^{h} q_i \geq \sum_{i=1}^{n} q_i - \sum_{k=1}^{m} Q_k \geq \sum_{i=1}^{h-1} q_i$$

5.2.2 La procédure d'insertion à coût minimal (IM)

Cette méthode d'insertion est celle utilisée dans l'algorithme RT original.

Pour chaque client qui n'est pas encore dans la solution, on identifie la position d'insertion à coût minimal parmi tous les véhicules de la flotte interne qui ont une capacité résiduelle assez grande pour pouvoir satisfaire ce client. Si aucun véhicule ne peut satisfaire ce client, alors ce client est ajouté à la liste du transporteur externe.

Nous avons modifié la méthode afin de la rendre stochastique. Dans ce cas, il s'agit de considérer les ϕ meilleures positions d'insertion et d'en choisir une au hasard (une telle approche s'inspire de la métaheuristique GRASP [13]). Bien sûr, avec $\phi = 1$, on revient à la version déterministe.

5.2.3 La procédure d'enveloppe convexe de $I3$ (EC)

Le coeur de cette méthode repose sur la génération d'une enveloppe convexe qui est le plus petit ensemble convexe contenant un certain ensemble de clients. Dans notre cas, une enveloppe convexe est créée pour chaque véhicule en se basant sur les clients affectés à ce véhicule. Ceci suppose donc une procédure préalable d'affectation des clients aux véhicules qui sera décrite plus loin. La figure 5.1 illustre une enveloppe convexe pour un ensemble de clients quelconques.

L'affectation des clients aux véhicules

Pour affecter un client à un véhicule, on calcule un indice de dispersion associé à chaque véhicule de la flotte interne, en supposant que le client est desservi par le véhicule examiné. On choisit ensuite le véhicule pour

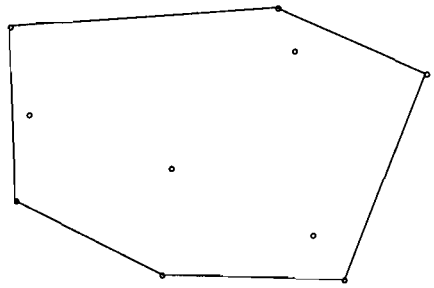


FIGURE 5.1 – Enveloppe convexe

lequel l'indice de dispersion est le moins élevé et pour lequel la contrainte de capacité est satisfaite, et on affecte le client à ce véhicule.

L'indice de dispersion s'inspire de celui utilisé par Gendreau *et al.* [20] qui utilise un tel indice pour un seul ensemble de clients ou pour deux ensembles de clients afin d'effectuer certains mouvements lors d'une recherche tabou. Soit d_k un véhicule, et soit c_{ij} la distance entre les clients v_i et v_j, où $i, j \in V_k$, avec V_k l'ensemble des clients affectés au véhicule d_k. Alors, l'indice de dispersion D_k est :

$$D_k = \begin{cases} \frac{\sum_{i,j \in V_k} c_{ij}}{|V_k|} & \text{si } |V_k| > 0 \\ 0 & \text{si } |V_k| = 0 \end{cases}$$

L'indice de dispersion de Gendreau *et al.* [20] pose $|V_k|(|V_k| - 1)$ au dénominateur. Mais en utilisant ce dénominateur, on a tendance à affecter le plus de clients possibles à un seul véhicule à la fois, puisque le dénominateur augmente très rapidement pour le véhicule choisi. Ainsi, les clients sont affectés à un premier véhicule jusqu'à ce que ce dernier soit à pleine capacité, et ainsi de suite.

Afin de rendre la procédure stochastique, on perturbe l'indice de dispersion à l'aide d'un nombre alétoire r tiré dans l'intervalle $[0, \beta]$, où β est un paramètre. L'indice de dispersion devient alors $D_k(1 + r)$. Évidemment, on revient au cas déterministe quand $\beta = 0$.

La création des routes.

Lorsque l'affectation des clients aux véhicules est complétée, il faut ensuite identifier l'enveloppe convexe des clients affectés à chacun des véhicules, puis créer une route à partir de cette enveloppe convexe. Ceci est fait à l'aide de la procédure utilisée en phase (1) de l'algorithme *I3*, procédure qui est appelée *CLOCK* [37]. Une fois que l'enveloppe convexe est identifiée, les clients qui n'en font pas partie sont ajoutés à la route par une méthode d'insertion à coût minimal, où le coût d'insertion classique ou détour $c_{ik} + c_{kj} - c_{ij}$ est remplacé par le ratio $\frac{c_{ik}+c_{kj}}{c_{ij}}$ lors de l'insertion du sommet k entre les sommets i et j.

Pseudo-code de EC

Dénotons par V^c l'ensemble des clients de la solution qui ne sont pas affectés à un véhicule et par D_k^i l'indice de dispersion pour l'ensemble des clients desservis par le véhicule d_k, additionné du client v_i. À chaque client v_i est associé une coordonnée cartésienne (x_i, y_i). On définit alors la fonction $v(x_i, y_i)$ qui associe un client v_i à sa position (x_i, y_i). Le paramètre β contrôle la perturbation stochastique.

1. Tant que $V^c \neq \emptyset$;

   1.1 $d_{min} \leftarrow \infty$;

   1.2 $\forall\ v_i \in V^c$ et $\forall\ k = 1, \ldots, m$, faire :

   - $r \leftarrow rand[0, \beta]$;
   - $p \leftarrow D_k^i(1 + r)$;
   - Si $p \leq d_{min}$, et les contraintes capacité du véhicule d_k restent satisfaites si l'on insère v_i dans la route du véhicule, alors :
     - $d_{min} \leftarrow p$;
     - $i^* \leftarrow i$;
     - $k^* \leftarrow k$;

   1.3 Si $d_{min} < \infty$, alors :

- Affecter le client v_{i^*} au véhicule d_{k^*} ;
- $V^c \leftarrow V^c \backslash \{v_{i^*}\}$;

1.4 Sinon, $\forall v_i \in V^c$;

- Affecter le client v_i au transporteur externe ;
- $V^c \leftarrow V^c \backslash \{v_i\}$;

2. Pour chaque véhicule d_k, $k = 1, \ldots, m$, créer l'enveloppe convexe du véhicule H_k :

2.1 Initialiser $\overline{x} \leftarrow \infty$ et $H_k \leftarrow \emptyset$;

2.2 *CLOCK* - Est - Nord / Est :

2.2.1 $R := \{v_i \in V_k \backslash H_k : x_i > \overline{x}\}$;

2.2.2 Si $R = \emptyset$, alors sauter à l'étape 2.3 ;

2.2.3 Sinon, $y \leftarrow \max_{v_i \in R} y_i$, et $x \leftarrow \min_{\substack{v_i \in R \\ y_i = y}} x_i$;

2.2.4 $\overline{x} \leftarrow x$, $\overline{y} \leftarrow y$, $R := R \backslash \{v(\overline{x}, \overline{y})\}$, $H_k := H_k \cup \{v(\overline{x}, \overline{y})\}$;

2.2.5 Répéter l'étape 2.2 ;

2.3 *CLOCK* - Sud - Est / Sud :

2.3.1 $R := \{v_i \in V_k \backslash H_k : y_i < \overline{y}\}$;

2.3.2 Si $R = \emptyset$, alors sauter à l'étape 2.4 ;

2.3.3 Sinon, $x \leftarrow \max_{v_i \in R} x_i$, et $y \leftarrow \max_{\substack{v_i \in R \\ x_i = x}} y_i$;

2.3.4 $\overline{x} \leftarrow x$, $\overline{y} \leftarrow y$, $R := R \backslash \{v(\overline{x}, \overline{y})\}$, $H_k := H_k \cup \{v(\overline{x}, \overline{y})\}$;

2.3.5 Répéter l'étape 2.3 ;

2.4 *CLOCK* - Ouest - Sud / Ouest :

2.4.1 $R := \{v_i \in V_k \backslash H_k : x_i < \overline{x}\}$;

2.4.2 Si $R = \emptyset$, alors sauter à l'étape 2.5 ;

2.4.3 Sinon, $y \leftarrow \min_{v_i \in R} y_i$, et $x \leftarrow \max_{\substack{v_i \in R \\ y_i = y}} x_i$;

2.4.4 $\overline{x} \leftarrow x$, $\overline{y} \leftarrow y$, $R := R \backslash \{v(\overline{x}, \overline{y})\}$, $H_k := H_k \cup \{v(\overline{x}, \overline{y})\}$;

2.4.5 Répéter l'étape 2.4 ;

2.5 *CLOCK* - Nord - Ouest / Nord :

2.5.1 $R := \{v_i \in V_k \backslash H_k : y_i > \overline{y}\}$;

2.5.2 Si $R = \emptyset$, alors sauter à l'étape 3 ;

2.5.3 Sinon, $x \leftarrow \min_{v_i \in R} y_i$, et $x \leftarrow \min_{v_i \in R, y_i = y} x_i$;

2.5.4 $\overline{x} \leftarrow x$, $\overline{y} \leftarrow y$, $R := R \backslash \{v(\overline{x}, \overline{y})\}$, $H_k := H_k \cup \{v(\overline{x}, \overline{y})\}$;

2.5.5 Répéter l'étape 2.5 ;

3. Pour chaque véhicule d_k, $k = 1, \ldots, m$, insérer les clients qui ne font pas partie de l'enveloppe convexe H_k :

   3.1 $R := V_k \backslash H_k$;

   3.2 Si $R \neq \emptyset$ alors $\forall v_i \in R$:

   - $(v_{opt}, v_{opt+1}) \leftarrow \arg\min_{\{v_t, v_{t+1}\} \in H_k} \frac{c_{i,t} + c_{i,t+1}}{c_{t,t+1}}$;
   - Insérer v_i entre les clients v_{opt} et v_{opt+1} ;

5.3 Chaînes d'éjection

Les chaînes d'éjection ont été introduites par Glover [23] pour le *TSP*. Plusieurs auteurs ont fait appel par la suite aux chaînes d'éjection dans leurs travaux pour résoudre le *TSP* [33], le *VRP* [34, 35, 36] et le *VRPTW* [4, 5, 16, 39, 40].

Une chaîne d'éjection est un mouvement complexe qui permet de générer un voisinage assez large autour de la solution courante. Cette structure de voisinage englobe, entres autres, les mouvements simples comme les réinsertions de clients et les échanges de clients entre deux routes. Elle consiste à déplacer un client d'une route k à une route $k+1$, provoquant l'éjection d'un client de la route $k+1$ qui provoquera à son tour l'éjection d'un client dans une route $k+2$, et ainsi de suite, créant une réaction en chaîne à plusieurs niveaux. L'idée des chaînes d'éjection provient du concept de chaînes alternées, où les arêtes alternent entre l'état utilisé et l'état libre.

Dénotons par $\underline{v}$ et $\overline{v}$ les sommets qui précèdent et succèdent au client v, respectivement. Partant d'un sommet v_0 qui est éjecté de sa route, une chaîne à ℓ niveaux se décrit comme (1) le remplacement des $\ell - 1$ triplets $(\underline{v}_k, v_k, \overline{v}_k)$,

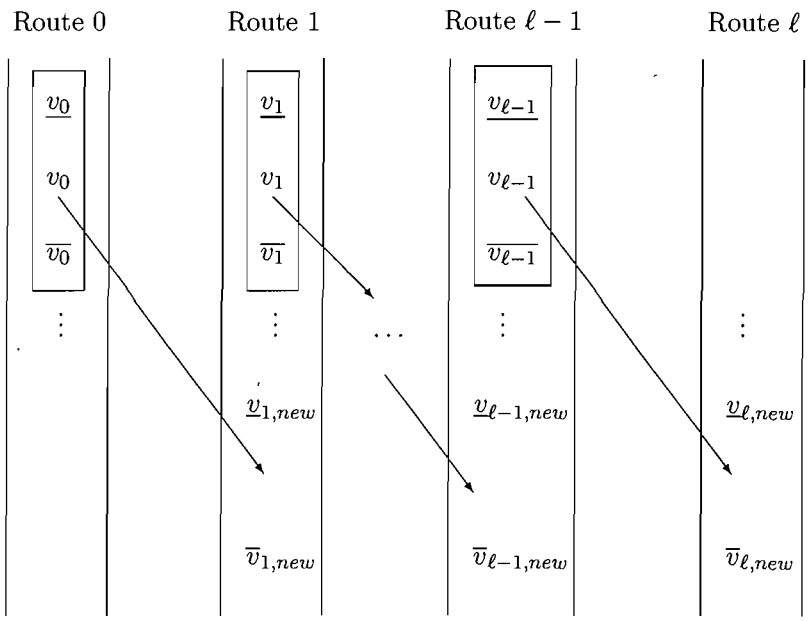


FIGURE 5.2 – Schéma représentant une chaîne d'éjection de longueur ℓ

$k = 1, \ldots, \ell-1$, par les triplets $(\underline{v}_{k,new}, v_{k-1}, \overline{v}_{k,new})$, $k = 1, \ldots, \ell-1$, respectivement, en plus de (2) l'insertion du dernier client $v_{\ell-1}$ soit dans la route qui a initié le processus pour obtenir $(\underline{v}_{0,new}, v_{\ell-1}, \overline{v}_{0,new})$ et ainsi compléter un cycle d'éjection ou bien dans une autre route pour créer une chaîne ouverte. Il faut noter que les clients $\underline{v}_{k,new}$ et $\overline{v}_{k,new}$ définissent la position du client v_{k-1} dans sa nouvelle route k. Les figures 5.2 et 5.3 illustrent les routes impliquées dans une chaîne d'éjection avant et après le mécanisme d'éjection.

Le voisinage défini par les chaînes d'éjection est évidemment de très grande taille, si l'on considère toutes les chaînes possibles. En conséquence, bien que notre procédure d'éjection examine un voisinage beaucoup plus

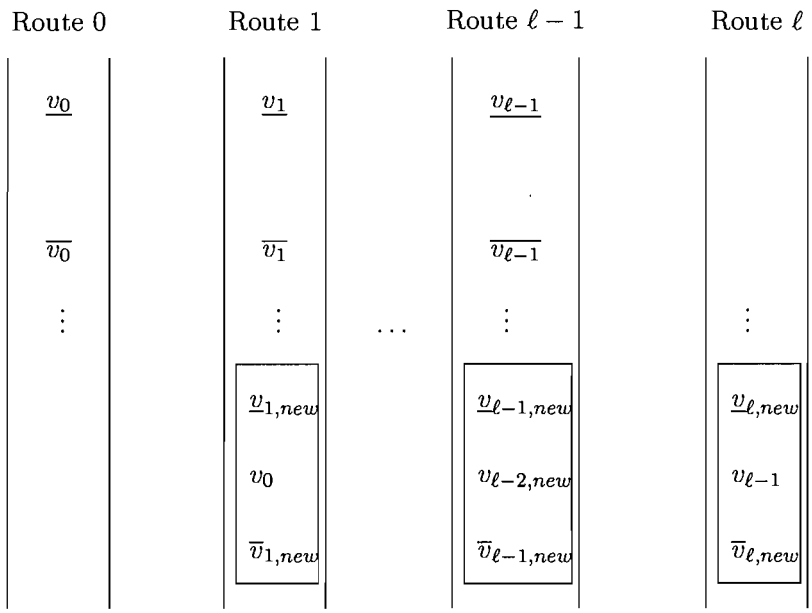


FIGURE 5.3 – Schéma représentant le résultat d'une chaîne d'éjection de longueur ℓ

grand que les simples insertions ou échanges de deux sommets (qui correspondent, respectivement, à des chaînes et cycles de longueur 1), il reste qu'il s'agit d'une heuristique qui n'examine qu'un sous-ensemble de mouvements possibles.

5.3.1 Construction du graphe d'éjection

Les chaînes d'éjection sont identifiées ici en appliquant un algorithme de Plus Courts Chemins (PCC) dans un graphe d'éjection (tel que décrit dans Gendreau *et al.* [16]). Dans ce graphe, on retrouve les sommets clients du

graphe original et des sommets artificiels qui représentent les routes des véhicules. On retrouve également deux types d'arcs. D'abord, des arcs d'éjection (v_i, v_j), où v_i et v_j sont des sommets clients, qui représentent l'éjection du client v_j par le client v_i. Le coût associé à cet arc correspond au coût d'insertion du client v_i dans la route qui dessert v_j, en considérant que v_j n'en fait plus partie, duquel on soustrait le gain obtenu par le retrait du client v_i de sa route originale. Ensuite, on retrouve des arcs d'insertion (v_i, d_k), où v_i est un sommet client et d_k un sommet véhicule, qui représentent l'insertion du client v_i dans la route du véhicule d_k. Le coût associé à un tel arc est simplement le coût d'insertion de v_i dans la route du véhicule d_k, duquel on soustrait le gain obtenu par le retrait du client v_i de sa route originale. Dans les deux cas, le coût d'insertion correspond à la meilleure insertion possible. Il faut noter qu'on ne considère que les éjections réalisables qui respectent les contraintes de capacité (une description plus détaillée est fournie plus loin). Les coûts des arcs sont calculés au début du lancement de l'évaluation du voisinage et sont conservés dans une matrice de coûts. On dispose également d'une matrice de positions d'insertion I où l'on conserve la position d'insertion de chaque client dans sa nouvelle route. Par exemple, si le sommet v_{k-1} est éjecté de sa route et est inséré entre les sommets $\underline{v}_{k,new}$ et $\overline{v}_{k,new}$ dans la nouvelle route k, on enregistre le client $\underline{v}_{k,new}$ pour retenir la position d'insertion de v_{k-1}. Enfin, une matrice de succession S nous permet de suivre la suite des éjections dans une chaîne. Un indice additionnel k permet également d'obtenir directement le *k-ième* client dans une chaîne. Avec les matrices de succession S et de positions d'insertion I, il est facile de recréer une chaine d'éjection et de l'appliquer à la solution courante.

Une particularité intéressante ici est l'inclusion des clients qui sont affectés au transporteur externe dans le graphe d'éjection, en plus d'un sommet artificiel pour la pseudo-route du transporteur externe (car il n'y a pas de séquencement dans cette route). L'insertion d'un client dans la pseudo-route représente l'affectation du client au transporteur externe. L'éjection d'un client

de cette pseudo-route représente l'introduction du client dans une route de la flotte interne. Ainsi, une chaîne d'éjection qui inclut la pseudo-route aura pour effet de remplacer un client par un autre dans l'ensemble des clients desservis par le transporteur externe (et vice-versa pour la flotte interne).

Une fois le graphe d'éjection construit, le but est alors de trouver un chemin de coût minimal dans ce graphe. Un tel chemin commence à un sommet client et se termine par une insertion à un sommet véhicule. Ceci est réalisé à l'aide de l'algorithme de *Floyd-Warshall* [14] qui peut trouver le meilleur chemin pour chaque paire (sommet client, sommet véhicule) dans le graphe. Soit un graphe G avec un ensemble de n sommets numérotés de 1 à n et soit $dist(k, i, j)$ la longueur du chemin le plus court du sommet i au sommet j, en utilisant au plus les sommets 1, ..., k comme sommets intermédiaires. Pour chaque paire de sommets i, j, $dist(0, i, j)$ est d'abord initialisé au coût original de l'arc (i,j), si cet arc existe. Sinon, cette valeur est mise à l'infini. À chaque itération, on introduit un nouveau sommet intermédiaire $k+1$ qui peut être utilisé ou non. Il y a donc deux possibilités : ou bien on améliore le meilleur chemin connu (passant, au plus, par les sommets 1, ..., k) en permettant le passage par $k+1$, ou bien on ne l'améliore pas, auquel cas le meilleur chemin connu demeure le même.

Pour un graphe orienté G = (V,A), nous avons donc la récurrence suivante :

$\forall i, j \in V,\ i \neq j$:

$$dist(k+1, i, j) = \min(dist(k, i, j)\ ,\ dist(k, i, k+1) + dist(k, k+1, j)).$$

5.3.2 Contraintes sur les chemins

Bien que l'algorithme de *Floyd-Warshall* soit relativement simple, son implémentation dans notre contexte est beaucoup plus complexe. En effet, il faut vérifier que les contraintes de capacité sont respectées. Plus difficile encore, comme il est permis à une route d'être visitée plusieurs fois dans une même chaîne d'éjection, il faut vérifier que la matrice des coûts est

toujours valide. Dans ce dernier cas, des restrictions sont imposées quant aux possibilités d'éjection de sommets. Bien sûr, nous aurions pu restreindre l'ensemble des chaînes d'éjection considérées à celles où un client au plus est éjecté d'une route, comme dans Gendreau *et al.* [16]. Cependant, notre approche permet d'augmenter l'ensemble des chaînes considérées.

Nous avons donc ajouté deux types de contraintes au modèle de chemin plus court, soit (1) des contraintes de capacité pour les véhicules et (2) des contraintes de positions sur les clients qui sont éjectés, autant dans la route originale que dans la route d'accueil (afin que la matrice des coûts demeure valide et nous assure de l'exactitude des coûts des chemins calculés). Toutes ces conditions sont vérifiées par une procédure, appelée *TR*, qui teste l'admissibilité des chaînes d'éjection considérées. En ce qui concerne les contraintes de capacité des véhicules, *TR* vérifie si une chaîne d'éjection satisfait ces contraintes pour tous les véhicules impliqués. Dans l'algorithme de *Floyd-Warshall*, de nouveaux chemins dans le graphe d'éjection sont obtenus en fusionnant deux chemins connus, soit un premier chemin menant du sommet origine au nouveau sommet intermédiaire et un second chemin menant de ce sommet intermédiaire au sommet destination. Il est donc facile de vérifier si les contraintes de capacité sont respectées. En ce qui concerne les contraintes de position, un statut est associé à chaque sommet qui correspond à sa position par rapport aux sommets faisant partie du chemin considéré dans le graphe d'éjection. Chaque sommet dispose d'un statut particulier parmi les six statuts suivants :

1. Le sommet est éjecté ;
2. Le sommet précède le sommet éjecté dans sa nouvelle route ;
3. Le sommet précède le sommet éjecté dans sa route originale ;
4. Le sommet succède au sommet éjecté dans sa route originale ;
5. Le sommet succède au sommet éjecté dans sa nouvelle route ;
6. Le sommet ne possède aucun des statuts précédents et n'est pas impliqué dans la chaîne d'éjection.

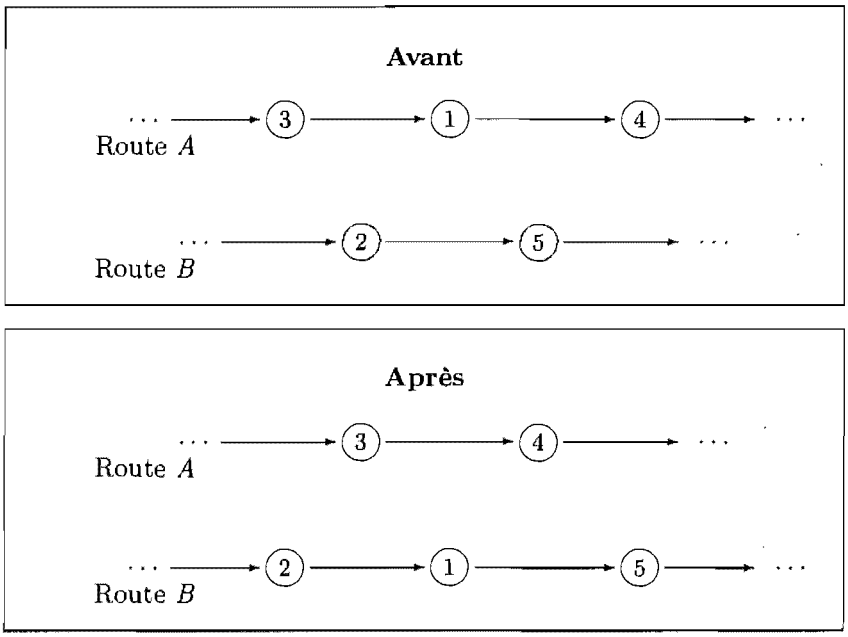


FIGURE 5.4 – Schéma représentant le statut des sommets impliqués dans l'éjection du sommet 1 de la route A à la route B

La figure 5.4 illustre les différents statuts possibles. Dans la figure, le sommet 1 est éjecté de la route A et inséré dans la route B (note : les indices des sommets correspondent à leur statut).

Considérons un chemin (v_a, ..., v_c) et supposons qu'un chemin (v_a, ..., v_b, ..., v_c), passant par un nouveau sommet intermédiaire v_b, est meilleur que celui qui ne passe pas par v_b. Pour chaque client, on considère le statut du client par rapport à la portion de chemin allant de v_a à v_b, ainsi que le statut du client par rapport à la portion de chemin allant de v_b à v_c. On exploite ensuite ces deux statuts de la façon suivante :

- Si au moins un des deux statuts est égal à 6 (i.e., ce client n'est pas impliqué dans la chaîne d'éjection ou n'est impliqué que dans l'une des deux portions de cette chaîne) OU si l'un des deux statuts est égal à 4

ou 5 tandis que l'autre est égal à 2 ou 3 (i.e., ce client est le successeur d'un client éjecté et le prédécesseur d'un autre client éjecté, dans la route originale ou dans la route d'accueil), alors le client est valide ;

- Sinon, le client n'est pas valide.

Si tous les clients sont valides, alors la matrice des coûts est toujours valide.

5.3.3 Complexité de calcul

L'algorithme permettant de déterminer la meilleure chaîne d'éjection comprend 3 parties :

- Initialiser la matrice des coûts A, la matrice des positions d'insertion I et la matrice de succession S ;
- Appliquer l'algorithme de *Floyd-Warshall* ;
- Déterminer et appliquer à la solution courante la chaîne d'éjection correspondant au meilleur chemin trouvé.

L'initialisation des matrices implique la mise à jour de n^2 valeurs, où chaque mise à jour consiste à trouver la meilleure insertion possible. Donc, on obtient une complexité de $O(n^3)$. Quant à l'algorithme de *Floyd-Warshall*, n^3 combinaisons de clients (i, l, j) sont considérées, et pour chacune d'entre elles la validité du nouveau chemin considéré est vérifiée en $O(n) + O(m) \equiv O(m + n)$, où $O(m)$ correspond à la vérification des contraintes de capacité et $O(n)$ à la vérification des statuts des clients. Lorsque l'algorithme est terminé, il faut choisir le meilleur chemin parmi toutes les paires (sommet client, sommet véhicule), ce qui se fait en $O(nm)$.

Donc, la complexité est dans l'ordre de

$$O(n^3) + O(n^3(n + m)) + O(nm) \equiv O(n^4 + mn^3)$$

5.3.4 Description complète de l'algorithme

Notre algorithme de chaînes d'éjection débute par l'initialisation des matrices de coût (A), de positions d'insertion (I) et de succession (S). Par

la suite, on calcule les meilleurs chemins pour chaque paire (sommet client, sommet véhicule) dans le graphe d'éjection à l'aide de l'algorithme de Floyd-Warshall, en vérifiant les contraintes de capacité des véhicules et de validité de la matrice des coûts. Une fois que l'algorithme se termine, il ne reste plus qu'à appliquer la chaîne d'éjection, correspondant au meilleur chemin trouvé, à la solution courante.

5.3.5 Critère d'activation pour les chaînes d'éjection

Puisque le but du voisinage avec chaînes d'éjection est d'obtenir des solutions de la plus grande qualité possible, l'idéal serait de l'utiliser à chaque itération de la recherche tabou. Cependant, cela ne serait pas du tout raisonnable en termes de temps de calcul. Il faut donc trouver un critère pour décider du moment où l'on applique ce voisinage. Dans un premier temps, l'activation de ce voisinage se fera toujours dans un mode de descente avec arrêt au premier minimum local rencontré. Ensuite, cette descente ne sera appliquée qu'à partir de solutions réalisables qui correspondent à des minima locaux par rapport aux structures de voisinage utilisées par la recherche tabou (i.e., déplacement et échange de clients). Enfin, le coût de la solution courante doit être assez proche du coût de la meilleure solution rencontrée. Ceci s'exprime comme un écart en pourcentage, dénoté ϵ, par rapport à cette meilleure solution.

5.4 Pseudo-code de *RTE*

Cette section décrit notre algorithme *RTE* en pseudo-code. Le tableau 5.1 rappelle d'abord la notation utilisée pour décrire l'algorithme RT au chapitre 3 et introduit également certains éléments additionnels propres au nouvel algorithme RTE.

L'algorithme débute en créant la solution initiale avec *EC*. On initialise par la suite les critères d'aspiration en fonction de cette solution initiale. La

recherche tabou comme telle débute à l'étape 5. À chacune des λ^{max} itérations, l'algorithme recherche parmi toutes les solutions voisines de la solution courante celles qui ne contiennent que des attributs non tabou ou des attributs qui satisfont le critère d'aspiration (étape 5.2). Aux étapes 5.3 et 5.4, on identifie dans ce sous-ensemble de solutions celle qui minimise la fonction $h(s)$, telle que décrite à la section 4.2.6. On effectue ensuite le mouvement requis pour générer cette solution et on met à jour τ_i^k et ρ_i^k, qui correspondent à la dernière itération tabou associée à l'attribut (i,k) et au nombre de fois où l'attribut (i,k) a été impliqué dans un mouvement, respectivement (étapes 5.5 et 5.6). À l'étape 5.7, une post-optimisation est effectuée sur chacune des routes de la nouvelle solution à l'aide de la procédure *4-opt**. Si cette nouvelle solution est réalisable, si elle correspond à un minimum local par rapport à la structure de voisinage de la recherche tabou (i.e., déplacement d'un client d'une route à une autre et échange de deux clients entre deux routes) et si elle est suffisamment proche de la meilleure solution rencontrée, on applique une descente locale basée sur les chaînes d'éjection (étape 5.9). Enfin, s'il y a amélioration par rapport à la meilleure solution, on effectue les mises à jour requises.

RTE

1. Créer la solution initiale s ;
2. Initialiser $\alpha \leftarrow 1$, $\lambda \leftarrow 0$;
3. $s^* \leftarrow s$;
4. Pour tous les attributs (i,k) dans l'espace-solution, faire :
   4.1 Initialiser $\rho_i^k \leftarrow 0$ et $\tau_i^k \leftarrow 0$;
   4.2 Si $(i,k) \in B(s)$ et s est réalisable, alors $\sigma_i^k \leftarrow f(s)$, sinon $\sigma_i^k \leftarrow \infty$;
5. Tant que $\lambda \leq \lambda^{max}$, faire :
   5.1 $M(s) \leftarrow \emptyset$;

5.2 $\forall\ \overline{s} \in N(s)$, faire :

- Si $\forall (i,k) \in B(\overline{s}) \backslash B(s)$:
  $(\tau_i^k < \lambda)$ ou ($\overline{s}$ est réalisable et $f(\overline{s}) < \sigma_i^k$),
  alors $M(s) \leftarrow M(s) \cup \{\overline{s}\}$;

5.3 $\forall\ \overline{s} \in M(s)$, faire :

- Si $g(\overline{s}) \geq g(s)$, alors :
  - $\rho^{min} \leftarrow min_{(i,k) \in B(\overline{s}) \backslash B(s)} \rho_i^k$;
  - $h(\overline{s}) \leftarrow g(\overline{s}) + \gamma \sqrt{nm} f(s) \frac{\rho^{min}}{\lambda}$;

  Sinon, $h(\overline{s}) \leftarrow g(\overline{s})$;

5.4 Trouver une solution s' qui minimise $h(\overline{s})$, $\overline{s} \in M(s)$;

5.5 $\forall\ (i,k) \in B(s) \backslash B(s')$, faire ;

- Enlever le client i de la route k ;
- $\tau_i^k \leftarrow \lambda + \theta$;

5.6 $\forall\ (i,k) \in B(s') \backslash B(s)$, faire :

- Insérer le client i dans la route k ;
- $\rho_i^k \leftarrow \rho_i^k + 1$;

5.7 Appliquer la procédure *4-opt** pour chaque route de s' qui a été modifiée afin d'obtenir s'' ;

5.8 $s \leftarrow s''$;

5.9 Si s'' est réalisable, alors :

5.9.1 Si s'' est un minimum local et $\frac{f(s'') - f(s^*)}{f(s^*)} < \epsilon$:

- Tant que $ejection(s'')$ apporte une amélioration :
  $s'' \leftarrow ejection(s'')$
- Si $f(s'') < f(s^*)$, alors : $s^* \leftarrow s''$;

  Sinon, si $f(s) < f(s^*)$, alors $s^* \leftarrow s$;

5.9.2 $\forall\ (i,k) \in B(s)$, $\sigma_i^k \leftarrow min\{\sigma_i^k, f(s)\}$;

5.9.3 $\alpha \leftarrow \frac{\alpha}{(1+\delta)}$;

5.10 Sinon $\alpha \leftarrow \alpha(1+\delta)$;

5.11 $\lambda \leftarrow \lambda + 1$;

6. Retourner s^*.

TABLEAU 5.1 – Notation

Notation	Interprétation
$B(s)$	Ensemble des attributs de la solution s
$f(s)$	coût de la solution s
$q(s)$	Demande excédentaire
$g(s)$	$f(s) + \alpha q(s)$
$h(s)$	$g(s) + \gamma\sqrt{nm} f(s) \frac{\rho^{min}}{\lambda}$
$N(s)$	Voisinage de la solution s
$M(s)$	Sous-ensemble de $N(s)$
s	Solution courante
$\overline{s}$	Solution dans $M(s)$
s'	Meilleure solution dans $M(s)$
s''	solution s' modifiée par *4-opt*
s^*	Meilleure solution trouvée
$\underline{s}$	Dernière solution réalisable rencontrée
α	Paramètre de pénalité pour la demande excédentaire
γ	Ajustement du paramètre de diversification
δ	Paramètre de mise à jour pour α
θ	Durée tabou
λ^{max}	Nombre total d'itérations
λ	Itération courante
ρ_i^k	Nombre de fois que l'attribut (i,k) est introduit dans la solution
σ_i^k	Critère d'aspiration de l'attribut (i,k)
τ_i^k	Dernière itération avec statut tabou pour l'attribut (i,k)
ϕ	Paramètre de perturbation de l'heuristique de construction IM
β	Paramètre de perturbation de l'heuristique de construction EC
ϵ	Critère d'activation des chaînes d'éjection

Chapitre 6

Résultats numériques

Dans cette section, nous présentons des résultats numériques obtenus sur des problèmes tests standards. Après une analyse de sensibilité des paramètres, nous comparons les résultats obtenus par RTE avec ceux de RT et de RIP.

6.1 Composition des problèmes tests

L'ensemble test utilisé est composé de 34 instances homogènes et de 34 instances hétérogènes. Ces instances ont été conçues pour le *VRP* et ont été modifiées par la suite pour le *VRPPC* par Bolduc *et al.* [3].

6.1.1 Problèmes homogènes

Toutes les instances pour le problème du *VRPPC* homogène proviennent d'instances pour le VRP. Quatorze d'entre elles sont décrites dans Christofides, Mingozzi et Toth [6] et sont dénotées *CE*, tandis que les 20 autres instances sont décrites dans Golden *et al.* [24], et sont dénotées *G*. Ces instances ont été modifiées pour le *VRPPC* de la façon suivante par Bolduc *et al.* [3]. Tout d'abord, le nombre de véhicules disponibles a été réduit afin de justifier la présence d'un transporteur externe. Pour chacune des instances, le nombre de véhicules dans la flotte privée est fixé à $\lceil 0.8\overline{q}/\overline{Q} \rceil$, où $\overline{q}$ est la

demande totale des clients et $\overline{Q}$ est la capacité totale de la flotte de véhicules originale. Les coûts variables sont fixés à 1 pour chaque unité de distance et les coûts fixes, où $f_k = f$ pour chaque véhicule k, correspondent à la longueur moyenne des routes des véhicules dans la meilleure solution connue de l'instance VRP originale (arrondie à la vingtaine près). Le coût e_i du transporteur externe pour un client i est fixé à $f/\overline{n} + \mu_i c_{0i}$, où $\overline{n}$ représente le nombre moyen de clients desservis par chacun des véhicules dans la meilleure solution connue de l'instance VRP originale, et où μ_i est un coefficient qui dépend de la demande du client i.

Pour obtenir ce dernier coefficient, il faut d'abord définir q_{min} et q_{max}, qui sont respectivement la plus petite demande et la plus grande demande dans l'ensemble des clients. Soit $\eta = (q_{max} - q_{min})/3$, alors :

$$\mu_i = \begin{cases} 1 & \text{si } q_i \in [q_{min}, q_{min} + \eta[\\ 1.5 & \text{si } q_i \in [q_{min} + \eta, q_{min} + 2\eta[\\ 2 & \text{si } q_i \in [q_{min} + 2\eta, q_{max}[\end{cases}$$

6.1.2 Problèmes hétérogènes

Les instances hétérogènes sont produites à partir des instances homogènes et sont dénotées par *CE-H* et *G-H* dans le tableau 6.2. Tout comme pour les instances homogènes, certaines modifications ont été apportées afin de les adapter au *VRPPC*. Les flottes de véhicules pour les instances *CE-H* et *G-H* sont composées de trois types de véhicules A, B et C que l'on retrouve en nombre fini, dénotés par m_A, m_B et m_C, respectivement. Les capacités et les coûts fixes de ces trois types sont 80% (type A), 100% (type B) et 120% (type C) de ceux de la flotte pour l'instance homogène correspondante. Le nombre de véhicules a aussi été modifié de sorte que la capacité de la flotte interne soit égale à environ 80% de la demande totale. Il arrive parfois qu'un certain type de véhicules ne fait pas partie de la flotte disponible. Un X apparaît alors dans le tableau 6.2 pour ce type. Toutes ces instances sont disponibles à l'adresse :

TABLEAU 6.1 – Caractéristiques des instances homogènes

Instance	n	m	Q	f
CE-01	50	4	160	120
CE-02	75	9	140	100
CE-03	100	6	200	140
CE-04	150	9	200	120
CE-05	199	13	200	100
CE-06	50	4	160	140
CE-07	75	9	140	120
CE-08	100	6	200	160
CE-09	150	10	200	120
CE-10	199	13	200	120
CE-11	120	6	200	180
CE-12	100	8	200	120
CE-13	120	6	200	260
CE-14	100	7	200	140
G-01	240	7	550	820
G-02	320	8	700	1060
G-03	400	8	900	1380
G-04	480	8	1000	1720
G-05	200	4	900	1620
G-06	280	5	900	1700
G-07	360	7	900	1460
G-08	440	8	900	1480
G-09	255	11	1000	60
G-10	323	13	1000	60
G-11	399	14	1000	80
G-12	483	15	1000	80
G-13	252	21	1000	60
G-14	320	23	1000	60
G-15	396	26	1000	60
G-16	480	29	1000	60
G-17	240	18	200	40
G-18	300	22	200	60
G-19	360	26	200	60
G-20	420	31	2000	60

http ://mcbolduc.com/VRPPC/test.htm.

6.2 Analyse de sensibilité

Dans la suite, nous rapportons des expériences visant à optimiser les différents paramètres de l'algorithme. Dépendamment du paramètre considéré, jusqu'à cinq résultats différents peuvent être rapportés, soit : la valeur de la meilleure solution obtenue et la valeur de la solution moyenne sur l'ensemble des redémarrages de la recherche tabou, en faisant appel ou non aux chaînes d'éjection (ces valeurs s'expriment comme un écart en pourcentage avec la meilleure solution connue). Enfin, le temps de calcul total en secondes est rapporté.

Les paramètres ont été optimisés sur un sous-ensemble d'instances choisies au hasard. Nous avons choisi 5 instances pour les classes *CE* et *CE-H*, et 6 instances pour les classes *G* et *G-H*, soit :

- CE-02, CE-03, CE-04, CE-09, CE-13 ;
- CE-H-03, CE-H-04, CE-H-07, CE-H-10, CE-H-12 ;
- G-02, G-05, G-10, G-15, G-16, G-20 ;
- G-H-03, G-H-07, G-H-10, G-H-13, G-H-16, G-H-19 ;

Tous les résultats qui suivent sont obtenus en lançant la recherche sur les sous-ensembles d'instances énoncés ci-haut. Les paramètres considérés sont : la procédure de création de la solution initiale (*IM* vs. *EC*), avec le paramètre de perturbation correspondant (ϕ vs. β) ; le critère ϵ pour activer les chaînes d'éjection ; le nombre d'itérations de la recherche tabou à chaque redémarrage ; et finalement, le nombre de redémarrages. Tous les résultats pour RTE et RT ont été obtenus sous le système d'exploitation Linux, avec un processeur DualCore AMD Opteron 275 de 2.2GHz. Les résultats ont donc été générés dans les mêmes conditions que Côté et Potvin [10], et dans des conditions similaires à celles rapportées pour RIP dans Bolduc *et al.* [3] où un processeur Xeon 3.6GHz a été utilisé.

TABLEAU 6.2 – Caractéristiques des instances hétérogènes

		A			B			C		
Instance	n	m_A	Q_A	f_A	m_B	Q_B	f_B	m_C	Q_C	f_C
CE-H-01	50	2	160	140	2	192	168	X	X	X
CE-H-02	75	4	112	80	5	168	120	X	X	X
CE-H-03	100	2	160	112	2	200	140	2	240	168
CE-H-04	150	2	160	96	4	200	120	3	240	144
CE-H-05	199	7	160	80	5	200	100	2	240	120
CE-H-06	50	1	128	112	2	160	140	1	192	168
CE-H-07	75	4	112	96	3	140	120	2	168	144
CE-H-08	100	1	160	128	1	200	160	4	240	192
CE-H-09	150	4	160	96	3	200	120	3	240	144
CE-H-10	199	2	160	96	5	200	120	6	240	144
CE-H-11	120	2	160	144	2	200	180	2	240	216
CE-H-12	100	2	160	96	3	200	120	3	240	144
CE-H-13	120	1	160	208	4	200	260	1	240	312
CE-H-14	100	1	160	96	1	200	120	5	240	144
G-H-01	240	3	440	656	1	550	820	3	660	984
G-H-02	320	2	560	848	2	700	1060	4	840	1272
G-H-03	400	3	720	1104	3	900	1380	2	1080	1656
G-H-04	480	2	800	1376	4	1000	1720	2	1200	2064
G-H-05	200	2	720	1296	2	900	1620	X	X	X
G-H-06	280	3	720	1360	2	900	1700	1	1080	2040
G-H-07	360	3	720	1168	1	900	1460	3	1080	1752
G-H-08	440	1	720	1184	2	900	1480	5	1080	1776
G-H-09	255	6	800	48	3	1000	60	3	1200	72
G-H-10	323	3	800	48	3	1000	60	6	1200	72
G-H-11	399	6	800	64	8	1000	80	1	1200	96
G-H-12	483	6	800	64	6	1000	80	4	1200	96
G-H-13	252	6	800	48	4	1000	60	10	1200	72
G-H-14	320	11	800	48	2	1000	60	11	1200	72
G-H-15	396	7	800	48	9	1000	60	10	1200	72
G-H-16	480	12	800	48	6	1000	60	11	1200	72
G-H-17	240	4	160	32	7	200	40	6	240	48
G-H-18	300	7	160	48	9	200	60	6	240	72
G-H-19	360	9	160	48	7	200	60	10	240	72
G-H-20	420	16	160	48	6	200	60	10	240	72

6.3 Création de la solution initiale

La solution initiale est le point de départ de la recherche tabou. Comme le résultat final peut dépendre de cette solution initiale, il faut bien choisir la méthode d'initialisation. Les résultats rapportés dans cette section sont obtenus en lançant *RTE* avec 10 redémarrages et 25 000 itérations par redémarrage (pour un total de 250 000 itérations). Pour ces expériences, nous avons comparé les deux méthodes de création de la solution initiale *IM* et *EC* en faisant varier leur paramètre respectif, ϕ et β (la version déterministe de chaque méthode est obtenue lorsque $\phi = 1$ et $\beta = 0$, respectivement). Il faut aussi noter que la recherche tabou a été lancée sans activer le voisinage des chaînes d'éjections, dû à des temps de calcul importants sur les plus grandes instances.

Les tableaux 6.3 et 6.4 présentent les résultats obtenus pour les instances homogènes et hétérogènes, respectivement. Le tableau 6.5 présente ensuite les résultats sur l'ensemble des instances CE, CE-H, G et G-H. Ce dernier tableau permet de constater que l'heuristique *EC* avec $\beta = 0.1$ est la meilleure en moyenne, autant pour la meilleure solution que pour la solution moyenne obtenue après 10 redémarrages. Bien que supplantée par d'autres approches sur des catégories particulières d'instances, *EC* avec $\beta = 0.1$ a finalement été retenue car nous voulons la méthode de construction la plus robuste possible (i.e., la moins influencée par le type d'instances). Pour ce qui est des temps de calcul, ils sont pratiquement équivalents pour chacune des méthodes.

TABLEAU 6.3 – Résultats avec différentes méthodes d'initialisation - flotte homogène

Instances	Méthode d'initialisation	Meilleure	Moyenne	Temps (sec)
CE	EC - $\beta = 0.0$	0.60%	1.04%	594
	EC - $\beta = 0.1$	0.51%	1.13%	597
	EC - $\beta = 0.2$	0.91%	1.11%	591
	EC - $\beta = 0.3$	0.84%	1.18%	597
	EC - $\beta = 0.4$	0.60%	1.13%	597
	EC - $\beta = 0.5$	0.66%	1.17%	593
	IM - $\phi = 1$	0.46%	1.11%	599
	IM - $\phi = 2$	0.83%	1.16%	602
	IM - $\phi = 3$	0.77%	1.14%	595
	IM - $\phi = 4$	0.86%	1.25%	591
	IM - $\phi = 5$	0.55%	1.09%	598
G	EC - $\beta = 0.0$	1.56%	1.81%	5 445
	EC - $\beta = 0.1$	0.95%	1.62%	5 674
	EC - $\beta = 0.2$	1.05%	1.62%	6 135
	EC - $\beta = 0.3$	1.21%	1.84%	5 100
	EC - $\beta = 0.4$	0.85%	1.65%	5 021
	EC - $\beta = 0.5$	1.00%	1.80%	5 058
	IM - $\phi = 1$	1.40%	1.67%	4 897
	IM - $\phi = 2$	1.41%	2.65%	5 280
	IM - $\phi = 3$	1.37%	2.09%	5 878
	IM - $\phi = 4$	1.32%	2.31%	5 254
	IM - $\phi = 5$	1.35%	2.03%	5 476

TABLEAU 6.4 – Résultats avec différentes méthodes d'initialisation - flotte hétérogène

Instances	Méthode d'initialisation	Meilleure	Moyenne	Temps (sec)
CE-H	EC - $\beta = 0.0$	0.85%	1.19%	667
	EC - $\beta = 0.1$	0.21%	0.99%	664
	EC - $\beta = 0.2$	0.60%	1.08%	672
	EC - $\beta = 0.3$	0.44%	1.01%	681
	EC - $\beta = 0.4$	0.60%	1.08%	668
	EC - $\beta = 0.5$	0.58%	1.13%	670
	IM - $\phi = 1$	0.71%	0.97%	669
	IM - $\phi = 2$	0.66%	1.05%	668
	IM - $\phi = 3$	0.32%	0.93%	664
	IM - $\phi = 4$	0.65%	1.05%	668
	IM - $\phi = 5$	0.51%	1.08%	668
G-H	EC - $\beta = 0.0$	0.72%	1.10%	6 914
	EC - $\beta = 0.1$	0.53%	1.26%	6 234
	EC - $\beta = 0.2$	0.66%	1.50%	6 256
	EC - $\beta = 0.3$	0.85%	1.61%	6 407
	EC - $\beta = 0.4$	0.94%	1.66%	6 640
	EC - $\beta = 0.5$	0.80%	1.23%	6 304
	IM - $\phi = 1$	1.34%	2.11%	7 164
	IM - $\phi = 2$	1.69%	2.32%	6 594
	IM - $\phi = 3$	1.29%	1.90%	7 399
	IM - $\phi = 4$	1.25%	1.92%	6 906
	IM - $\phi = 5$	1.42%	2.13%	6 982

TABLEAU 6.5 – Résultats avec différentes méthodes d'initialisation - flotte homogène et hétérogène

Méthode d'initialisation	Meilleure	Moyenne	Temps (sec)
EC - $\beta = 0.0$	0.95%	1.30%	3 657
EC - $\beta = 0.1$	**0.57%**	**1.27%**	3 534
EC - $\beta = 0.2$	0.81%	1.35%	3 667
EC - $\beta = 0.3$	0.85%	1.44%	3 429
EC - $\beta = 0.4$	0.76%	1.41%	3 468
EC - $\beta = 0.5$	0.77%	1.35%	3 386
IM - $\phi = 1$	1.01%	1.50%	3 577
IM - $\phi = 2$	1.18%	1.86%	3 527
IM - $\phi = 3$	0.97%	1.56%	3 907
IM - $\phi = 4$	1.04%	1.68%	3 603
IM - $\phi = 5$	0.99%	1.63%	3 685

6.4 Critère d'activation des chaînes d'éjection

Le seuil ϵ qui permet d'activer la descente locale basée sur les chaînes d'éjection (i.e., la valeur du minimum local courant doit être à moins de $\epsilon\%$ de la valeur de la meilleure solution rencontrée) a un impact important sur la fréquence d'application des chaînes d'éjection et donc sur les temps de calcul et la qualité de la solution finale. Les résultats qui suivent ont été obtenus en lançant la recherche tabou avec 10 redémarrages et 25 000 itérations par redémarrage. Ils permettent de comparer la recherche tabou avec et sans chaînes d'éjection et donc de mesurer l'apport de ce nouveau voisinage.

TABLEAU 6.6 – Résultats avec différents critères d'activation

ϵ	MeS	MeA	MoS	MoA	AMe	AMo	Temps (sec)
0.05%	0.58%	0.54%	1.18%	1.16%	5.45%	1.73%	5 141
0.10%	0.52%	0.45%	1.04%	1.02%	12.20%	2.05%	8 484
0.15%	0.48%	0.47%	1.08%	1.05%	3.41%	2.46%	10 856
0.20%	0.47%	0.44%	0.98%	0.95%	6.27%	2.41%	21 344

Dans le tableau 6.6, *MeS* et *MeA* désignent les meilleures solutions obtenues après 10 redémarrages, sans et avec chaînes d'éjection, respectivement ; *MoS* et *MoA* désignent la moyenne des valeurs des solutions obtenues, sans et avec chaînes d'éjection, pour les 10 redémarrages, respectivement ; enfin, *AMe* et *AMo* correspondent à l'amélioration apportée par les chaînes d'éjection sur la meilleure solution obtenue et sur la moyenne des solutions, respectivement. En l'occurrence,

$$AMe = \frac{MeS - MeA}{MeS} \quad \text{et} \quad AMo = \frac{MoS - MoA}{MoS}.$$

Dans le tableau 6.6, on remarque une amélioration importante de la meilleure solution obtenue pour $\epsilon = 0.10\%$. Bien qu'une amélioration de la moyenne des solutions est aussi observée jusqu'à $\epsilon = 0.15\%$, nous retenons $\epsilon = 0.10\%$, afin de ne pas trop augmenter les temps de calcul.

6.5 Nombre d'itérations à chaque redémarrage

Le nombre d'itérations par redémarrage définit l'intensité de la recherche tabou à partir de la solution initiale et affecte directement les temps de calcul. La figure 6.1 illustre sous forme graphique l'évolution d'une solution à toutes les 5 000 itérations. Les données sont générées en lancant la recherche une seule fois, donc sans aucun redémarrage.

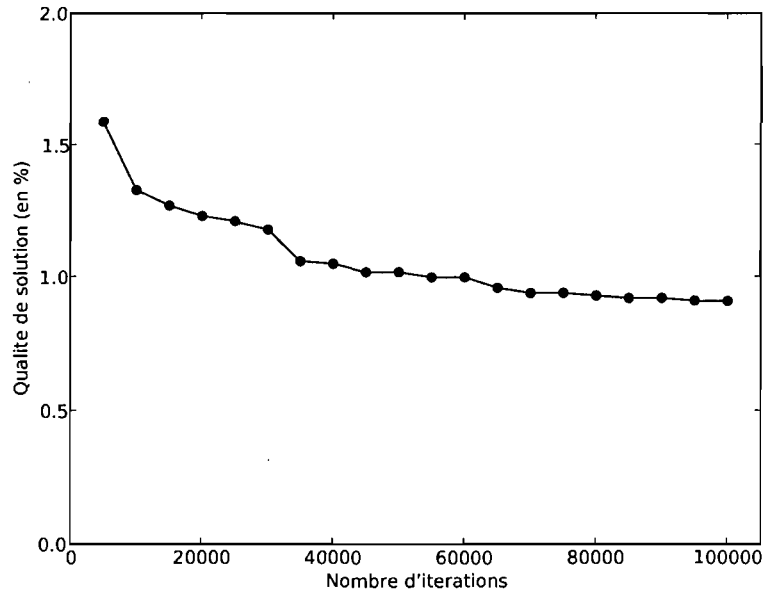


FIGURE 6.1 – Valeur de la solution en fonction du nombre d'itérations

La figure 6.1 représente l'évolution de la qualité de la solution avec le nombre d'itérations. Ici, on s'intéresse au point où la courbe commence à s'aplanir. On remarque qu'entre les itérations 40 000 et 60 000, les améliorations commencent à se faire beaucoup plus modestes. Nous avons donc fixé le nombre d'itérations à 50 000. Des tendances similaires ont été observées à chaque fois que ce type d'expérience a été répété. Le tableau 6.7 présente les

résultats sous forme de tableau. Le temps de calcul total de 3447 secondes correspond à 100 000 itérations.

TABLEAU 6.7 – Résultats avec différents nombres d'itérations

Nb. it.	Qualité de sol.
5 000	1.27%
10 000	1.05%
15 000	0.97%
20 000	0.92%
25 000	0.88%
30 000	0.86%
35 000	0.83%
40 000	0.79%
45 000	0.78%
50 000	0.75%
55 000	0.73%
60 000	0.65%
65 000	0.64%
70 000	0.63%
75 000	0.62%
80 000	0.60%
85 000	0.60%
90 000	0.60%
95 000	0.60%
100 000	0.59%

6.6 Nombre de redémarrages

Afin d'optimiser le nombre de redémarrages, nous avons lancé la recherche plusieurs fois sur notre sous-ensemble de problèmes tests en faisant varier ce nombre. Le tableau 6.8 affiche les résultats obtenus. En examinant ce tableau, on remarque qu'il y a amélioration entre 5 et 10 redémarrages, mais que, par la suite, une sorte de plateau est atteint. On a donc fixé le nombre de redémarrages à 10.

6.7 Résultats globaux

On rapporte dans cette section les résultats sur l'ensemble des 68 problèmes tests pour notre recherche tabou *RTE*, la recherche tabou originale *RT* de Côté et Potvin [10] et la métaheuristique *RIP* de Bolduc *et al.* [3]. Les résultats pour *RTE* sont obtenus en lançant 10 redémarrages de la recherche tabou avec 50 000 itérations par redémarrage. Nous rapportons dans les tableaux 6.9 et 6.10 la valeur de la meilleure solution obtenue suite à ces 10 redémarrages (*obj*) ainsi que le temps d'exécution total en secondes (*sec*), pour les instances homogènes et hétérogènes, respectivement. Pour les résultats de *RT*, nous avons reproduit nous-mêmes les résultats en lançant le code original avec le même nombre de redémarrages et d'itérations par redémarrage. Quant aux résultats de *RIP*, nous prenons les valeurs rapportées dans Bolduc *et al.* [3].

On observe une amélioration au niveau de la qualité des solutions autant pour les instances homogènes qu'hétérogènes. Ainsi, l'écart moyen par rapport à la meilleure solution connue sur l'ensemble des 34 instances homogènes est de 0.47%, 0.64% et 1.07% pour *RTE*, *RT* et *RIP*, respectivement. Lorsque l'on examine les résultats selon le type d'instances, on observe un léger avantage pour *RT* sur les instances *CE* qui sont de plus petite taille. Sur les instances *G* de plus grande taille, *RTE* est de beaucoup supérieure à *RT*. Il reste toutefois que les temps de calcul de *RTE* sont considérables par rapport à ceux de *RT* et *RIP*. Sur les instances hétérogènes, la supériorité de *RTE* est encore plus évidente et est observée autant pour les intances *CE*

TABLEAU 6.8 – Résultats avec différents nombres de redémarrages

Nb de redémarrages	Meilleure Solution
1	0.85%
5	0.42%
10	0.35%
15	0.38%
20	0.34%

que *G*, bien que *RTE* semble encore une fois atteindre son plein potentiel sur les plus grandes instances de type *G*. Sur l'ensemble des 34 instances hétérogènes de type *CE* et *G*, l'écart moyen par rapport à la meilleure solution connue est de 0.34%, 0.97% et 1.02% pour *RTE*, *RT* et *RIP*, respectivement.

Les tableaux 6.11 et 6.12 rapportent les meilleures solutions produites par chacun des algorithmes sur chaque instance. Ces solutions ont été obtenues suite à de multiples expérimentations avec différentes valeurs pour les paramètres. Bolduc *et al.* [3] ainsi que Côté et Potvin [10] rapportent de tels résultats dans leurs articles et nous avons donc décidé de faire de même, bien que nous n'ayons fait aucun effort particulier à cet égard. On remarque néanmoins que notre algorithme *RTE* a produit la meilleure solution connue pour 16 des 34 instances homogènes (dont 8 nouvelles meilleures solutions) et pour 27 des 34 instances hétérogènes (dont 24 nouvelles meilleures solutions).

TABLEAU 6.9 – Résultats globaux pour les instances homogènes

Instances	*RTE*		*RT*		*RIP*	
	obj	sec	obj	sec	obj	sec
CE-01	1 119.47	249	1 119.47	243	1 132.91	25
CE-02	1 814.52	339	1 814.52	330	1 835.76	73
CE-03	1 921.10	810	1 930.66	786	1 959.65	107
CE-04	2 525.17	2 006	2 525.46	1 932	2 545.72	250
CE-05	3 113.58	3 532	3 117.10	3 099	3 172.22	474
CE-06	1 207.47	252	1 207.47	255	1 208.33	25
CE-07	2 006.52	340	2 006.45	327	2 006.52	71
CE-08	2 060.17	816	2 056.59	851	2 082.75	110
CE-09	2 438.43	1 882	2 435.97	1 853	2 443.94	260
CE-10	3 406.82	3 457	3 401.83	3 111	3 464.90	478
CE-11	2 353.39	1 310	2 332.36	1 263	2 333.03	195
CE-12	1 952.86	595	1 952.86	604	1 953.55	128
CE-13	2 882.70	1 321	2 860.89	1 300	2 864.21	188
CE-14	2 216.97	642	2 216.97	650	2 224.63	110
Moyenne	0.35%	1 254	0.25%	1 186	0.75%	178.1
G-01	14 190.01	11 838	14 218.83	6 383	14 388.58	651
G-02	19 208.52	52 206	19 729.96	12 152	19 505.00	1 178
G-03	24 592.18	15 9404	25 653.58	22 413	24 978.17	2 061
G-04	34 802.08	55 087	36 022.73	38 339	34 957.98	3 027
G-05	14 261.31	8 478	14 673.56	8 750	14 683.03	589
G-06	21 498.03	15 911	22 278.99	14 453	22 260.19	1 021
G-07	23 513.06	55 140	24 191.41	20 528	23 963.36	1 628
G-08	30 073.56	57 290	30 627.91	30 599	30 496.18	2 419
G-09	1 325.62	8 191	1 328.14	6 110	1 341.17	832
G-10	1 590.82	17 623	1 590.83	9 388	1 612.09	1 294
G-11	2 173.80	32 843	2 172.28	14 927	2 198.45	2 004
G-12	2 495.02	85 876	2 492.75	23 097	2 521.79	2 900
G-13	2 274.12	5 045	2 278.99	3 608	2 286.91	802
G-14	2 703.31	9 769	2 705.00	6 104	2 750.75	1 251
G-15	3 161.26	19 520	3 158.92	9 248	3 216.99	1 862
G-16	3 638.39	46 751	3 639.11	13 137	3 693.62	2 778
G-17	1 633.35	4 720	1 636.11	4 026	1 701.58	806
G-18	2 710.21	8 920	2 705.90	6 220	2 765.92	1 303
G-19	3 497.72	14 184	3 497.54	10 125	3 576.92	1 903
G-20	4 306.89	24 763	4 311.17	13 689	4 378.13	2 800
Moyenne	0.56%	34 678	0.92%	13 665	1.29%	1 655.5

TABLEAU 6.10 – Résultats globaux pour les instances hétérogènes

Instances	*RTE*		*RT*		*RIP*	
	obj	sec	obj	sec	obj	sec
CE-H-01	1 191.70	260	1 191.70	257	1 192.72	26
CE-H-02	1 791.21	348	1 795.51	337	1 798.26	72
CE-H-03	1 917.96	808	1 926.33	790	1 934.85	105
CE-H-04	2 481.68	1 985	2 481.64	1 956	2 493.93	251
CE-H-05	3 143.01	3 424	3 143.92	2 959	3 195.66	490
CE-H-06	1 206.82	251	1 206.82	254	1 210.23	25
CE-H-07	2 031.85	320	2 035.90	325	2 042.79	74
CE-H-08	1 986.51	845	1 991.23	814	2 015.72	112
CE-H-09	2 447.58	1 930	2 445.49	1 889	2 445.88	267
CE-H-10	3 272.37	3 424	3 271.70	3 095	3 304.69	482
CE-H-11	2 336.51	1 339	2 325.74	1 270	2 308.76	188
CE-H-12	1 915.05	604	1 912.47	607	1 908.74	130
CE-H-13	2 868.13	1 365	2 872.14	1 250	2 842.18	195
CE-H-14	1 907.75	672	1 925.46	658	1 920.36	114
Moyenne	0.41%	1 255	0.53%	1 176	0.70%	180.8
G-H-01	14 194.13	11 937	14 174.27	6 425	14 408.31	647
G-H-02	18 537.70	49 553	19 056.69	12 849	18 663.15	12 54
G-H-03	25 177.92	119 962	25 899.93	22 587	25 561.55	2 053
G-H-04	34 991.21	40 791	35 988.06	40 957	35 495.66	2 904
G-H-05	15 411.82	7 542	15 895.94	7 996	16 138.50	512
G-H-06	19 859.30	19 541	20 381.35	13 569	20 329.04	1 005
G-H-07	23 481.28	91 673	23 915.77	22 625	24 184.83	1 608
G-H-08	27 334.84	186 252	28 121.26	34 082	27 710.66	2 584
G-H-09	1 329.27	18 293	1 331.11	5 927	1 346.03	814
G-H-10	1 555.59	15 643	1 554.96	10 871	1 575.82	1 332
G-H-11	2 195.83	32 079	2 191.23	14 455	2 218.91	2 140
G-H-12	2 482.92	42 240	2 535.00	21 083	2 510.07	2 970
G-H-13	2 237.38	18 013	2 231.88	4 058	2 253.45	733
G-H-14	2 684.70	10 429	2 685.51	6 303	2 711.81	1 246
G-H-15	3 127.33	21 112	3 123.60	9 766	3 156.93	1 895
G-H-16	3 621.85	62 174	3 853.21	15 712	3 649.09	2 785
G-H-17	1 664.08	5 226	1 674.91	3 969	1 705.48	762
G-H-18	2 708.73	11 325	2 722.32	6 177	2 759.99	1 299
G-H-19	3 443.59	16 271	3 445.85	8 418	3 517.48	1 892
G-H-20	4 314.16	29 698	4 306.53	13 922	4 413.82	2 733
Moyenne	0.30%	40 488	1.28%	14 088	1.25%	1 658.4

TABLEAU 6.11 – Meilleures solutions pour les instances homogènes

Instances	*RTE*	*RT*	*RIP*
CE-01	**1 119.47**	**1 119.47**	**1 119.47**
CE-02	**1 814.52**	**1 814.52**	**1 814.52**
CE-03	**1 919.05**	**1 919.05**	1 937.23
CE-04	2 513.85	**2 511.20**	2 528.36
CE-05	3 104.82	**3 092.01**	3 107.04
CE-06	**1 207.47**	**1 207.47**	**1 207.47**
CE-07	**2 004.53**	**2 004.53**	2 006.52
CE-08	2 055.96	2 052.91	**2 052.05**
CE-09	2 426.67	**2 423.25**	2 436.02
CE-10	3 395.70	**3 389.69**	3 407.13
CE-11	2 331.49	**2 330.94**	2 332.21
CE-12	**1 952.86**	**1 952.86**	1 953.55
CE-13	**2 858.83**	2 859.23	**2 858.94**
CE-14	**2 214.11**	**2 214.11**	2 216.68
G-01	**14 157.03**	14 160.77	14 160.77
G-02	**19 140.20**	19 449.92	19 234.03
G-03	**24 539.33**	25 050.77	24 646.79
G-04	**34 502.34**	34 788.83	34 607.12
G-05	**14 214.39**	14 481.03	14 249.82
G-06	**21 498.03**	21 632.24	21 703.54
G-07	**23 405.06**	23 829.88	23 549.53
G-08	**29 814.41**	30 077.79	30 173.53
G-09	1 323.91	**1 319.20**	1 336.91
G-10	1 587.00	**1 583.46**	1 598.76
G-11	2 165.44	**2 164.10**	2 179.71
G-12	2 484.01	**2 478.66**	2 503.71
G-13	2 271.64	**2 265.81**	2 268.32
G-14	2 693.53	**2 692.71**	2 704.01
G-15	3 158.49	**3 154.12**	3 171.20
G-16	3 635.34	**3 629.25**	3 654.20
G-17	1 633.35	**1 631.90**	1 677.22
G-18	2 706.84	**2 695.47**	2 742.72
G-19	3 481.34	**3 445.07**	3 528.36
G-20	4 294.32	**4 265.15**	4 352.95

TABLEAU 6.12 – Meilleures solutions pour les instances hétérogènes

Instances	*RTE*	*RT*	*RIP*
CE-H-01	**1 191.70**	**1 191.70**	**1 191.70**
CE-H-02	**1 789.41**	1 794.60	1 790.67
CE-H-03	**1 913.66**	1 923.55	1 919.05
CE-H-04	**2 468.38**	2 489.60	2 475.16
CE-H-05	**3 128.79**	3 146.95	3 146.45
CE-H-06	**1 204.48**	**1 204.48**	**1 204.48**
CE-H-07	**2 025.98**	2 033.24	**2 025.98**
CE-H-08	**1 983.71**	1 989.89	1 984.36
CE-H-09	**2 437.21**	2 451.80	2 438.73
CE-H-10	**3 260.07**	3 268.96	3 267.85
CE-H-11	2 306.83	2 332.79	**2 303.13**
CE-H-12	**1 908.05**	1 912.47	1 908.74
CE-H-13	**2 834.47**	2 871.55	2 842.18
CE-H-14	1 907.75	1 925.46	**1 907.74**
G-H-01	**14 129.07**	14 256.85	14 251.75
G-H-02	**18 469.83**	18 848.53	18 560.07
G-H-03	**25 135.60**	25 748.06	25 356.63
G-H-04	34 609.71	35 942.52	**34 589.11**
G-H-05	**15 411.82**	16 042.97	15 667.13
G-H-06	**19 778.44**	20 339.23	19 975.32
G-H-07	**23 423.05**	23 814.04	23 510.98
G-H-08	**27 291.12**	28 177.29	27 420.68
G-H-09	**1 325.70**	1 331.35	1 331.83
G-H-10	**1 550.20**	1 554.74	1 561.52
G-H-11	2 188.00	**2 185.07**	2 195.31
G-H-12	2 482.92	**2 479.42**	2 487.38
G-H-13	**2 230.84**	2 235.36	2 239.18
G-H-14	**2 671.03**	2 686.97	2 682.85
G-H-15	**3 113.26**	3 123.49	3 131.89
G-H-16	**3 615.82**	3 624.01	3 629.41
G-H-17	**1 660.20**	1 675.67	1 695.75
G-H-18	2 707.05	**2 704.15**	2 740.05
G-H-19	**3 430.57**	3445.32	3 464.70
G-H-20	4 302.47	**4 301.45**	4 352.35

Chapitre 7

Conclusion

Dans ce travail, nous avons amélioré une heuristique de recherche tabou qui s'était déjà révélée efficace pour les problèmes avec flotte homogène. Ces améliorations ont été apportées en modifiant l'approche de création de la solution initiale ainsi qu'en ajoutant une structure de voisinage basée sur les chaînes d'éjection. La création de la solution initiale, de nature stochastique, permet de diversifier la recherche d'un redémarrage à l'autre. Quant aux chaînes d'éjection, elles permettent à l'algorithme de pousser encore plus loin l'intensité de la recherche. Ces deux ajouts améliorent donc grandement la qualité des solutions obtenues. Cependant, la procédure de chaînes d'éjection est coûteuse en temps de calcul. La force de notre algorithme se situe donc au niveau de la qualité des solutions, et non au niveau des temps d'exécution.

Comme développements futurs, nous voudrions d'abord rendre notre implantation plus efficace afin de réduire les temps d'exécution. Ensuite, nous aimerions intégrer les chaînes d'éjection au sein de la structure de voisinage de la recherche tabou (plutôt que de faire appel à une simple descente locale). Nous aimerions enfin remplacer la stratégie de diversification continue, où l'évaluation des solutions voisines est biaisée vers celles qui sont différentes des solutions précédemment visitées, par un mécanisme de perturbation plus brutal, comme l'application d'un certain nombre de mouvements *4-opt** aléatoires, lorsque la recherche stagne. Dans une perspective plus large, il pour-

rait aussi être intéressant d'explorer d'autres approches métaheuristiques pour résoudre le *VRPPC*.

Bibliographie

[1] Ball, M.O., Assad, A.A., Bodin, L.D., Golden, B.L., 1983. «Planning for Truck Fleet Size in the Presence of a Common-Carrier Option». *Decision Sciences* 14, pp. 103-120.

[2] Bolduc, M.-C., Boctor, F., Renaud, J., 2006. «A Heuristic for the Routing and Carrier Selection Problem». Technical Report 2006-006, Faculté des Sciences de l'Administration, Université Laval.

[3] Bolduc, M.-C., Renaud, J., Boctor, F., Laporte, G., 2008. «A Perturbation Metaheuristic for the Vehicle Routing Problem with Private Fleet and Common Carriers». *Journal of the Operational Research Society* 59, pp. 776-787.

[4] Braysy, O., 2003. «A Reactive Variable Neighborhood Search for the Vehicle Routing Problem with Time Windows». *INFORMS Journal on Computing* 15, pp. 347-368.

[5] Caseau, Y., Laburthe, F., 1999. «Heuristics for Large Constrained Vehicle Routing Problems». *Journal of Heuristics* 5, pp. 281-303.

[6] Christofides, N., Mingozzi, A., Toth, P., 1979. «The Vehicle Routing Problem». Dans *Combinatorial Optimization*, Christofides N., Mingozzi A., Toth P., Sandi C. (eds), Wiley, Chichester, pp. 315-338.

[7] Chu, C.-W., 2005. «A Heuristic Algorithm for the Truckload and Less-than-truckload Problem». *European Journal of Operational Research* 165, pp. 657-667.

[8] Clarke, G., Wright, J., 1964. «Scheduling of Vehicles from a Central Depot to a Number of Delivery Points». *Operations Research* 12, pp. 568-581.

[9] Cordeau, J.-F., Laporte, G., Mercier, A., 2001. «A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows». *Journal of the Operational Research Society* 52, pp. 928-936.

[10] Côté, J.-F., Potvin, J.-Y., 2007. «A Tabu Search Heuristic for the Vehicle Routing Problem with Private Fleet and Common Carrier». Document de travail CIRRELT-2007-08, CIRRELT, Université de Montréal, à paraître dans *European Journal of Operational Research.*

[11] Dantzig, G.B., Ramser, J.H., 1959. «The Truck Dispatching Problem». *Management Science* 6, pp. 80-91.

[12] Diabi, M., Ramesh, R., 1995. «The Distribution Problem with Carrier Service : A Dual Based Penalty Approach». *ORSA Journal on Computing* 7, pp. 24-35.

[13] Feo, T.A., Resende, M.G.C., 1995. «Greedy Randomized Adaptive Search Procedures». *Journal of Global Optimization* 6, pp. 109-133.

[14] Floyd, R.W., 1962. «Algorithm 97 : Shortest Path». *Communications of the ACM* 5, p. 345.

[15] Fredman, M.L., Johnson, D.S., McGeoch, L.A., Ostheimer, G., 1993. «Data Structures for Traveling Salesmen. Symposium on Discrete Algorithms», Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 145-154.

[16] Gendreau, M., Guertin, F., Potvin, J.-Y., Séguin, R., 2006. «Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-ups and Deliveries». *Transportation Research - Part C* 14, pp. 157-174.

[17] Gendreau, M., Hertz, A., Laporte, G., 1994. «A Tabu Search Heuristic for the Vehicle Routing Problem». *Management Science* 40, pp. 1276-1290.

[18] Gendreau, M., Laporte, G., Musaraganyi, C., Taillard, E.D., 1999. «A Tabu Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem». *Computers & Operations Research* 26, pp. 1153-1173.

[19] Gendreau, M., Laporte, G., Potvin, J.-Y., «Metaheuristics for the Capacitated VRP». In Toth, P., Vigo, D., 2002. *The Vehicle Routing Problem*, chapitre 6, pp. 129-154. Philiadelphia : SIAM.

[20] Gendreau, M., Laporte, G., Semet, F., 1998. «A Tabu Search Heuristic for the Undirected Selective Traveling Salesman Problem». *European Journal of Operational Research* 106, pp. 539-545.

[21] Glover, F., 1989. Tabu Search - «Part I», *ORSA Journal on Computing* 1, pp. 190-206.

[22] Glover, F., 1990. Tabu Search - «Part II», *ORSA Journal on Computing* 2, pp. 4-32.
University of Colorado at Boulder.

[23] Glover, F., 1996. «Ejection chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems». *Discrete Applied Mathematics* 65, pp. 223-253.

[24] Golden, B.L., Wasil, E.A., Kelly, J.P., Chao, I.-M., 1998. «The Impact of Metaheuristics on Solving the Vehicle Routing Problem : Algorithms, Problems Tests and Computational Results». T.G. Crainic, G. Laporte (eds.), *Fleet Management and Logistics*, Kluwer, Boston, pp. 33-56.

[25] Klincewicz, J.G., Luss, H., Pilcher, M., 1990. «Fleet Size Planning when Outside Carrier Services Are Available». *Transportation Science* 24, pp. 169-182.

[26] Kulkarni, R.V., Bhave, P.R., 1985. «Integer Programming Formulations of Vehicle Routing Problems». *European Journal of Operational Research* 20, pp. 58-67.

[27] Li, F., Golden, B.L., Wasil, E., 2005. «Very Large-scale Vehicle Routing : New Problems, Algorithms, and Results». *Computers & Operations Research* 32, pp. 1165-1179.

[28] Li, F., Golden, B.L., Wasil, E., 2007. «A Record-to-record Travel Algorithm for Solving the Heterogeneous Fleet Vehicle Routing Problem». *Computers & Operations Research* 34, pp. 2734-2742.

[29] Lin, S., 1965. «Computer Solutions of the Traveling Salesman Problem». *Bell Systems Technical Journal* 44, pp. 2245-2269.

[30] Lin, S., Kernighan, B.W., 1973. «An Effective Heuristics Algorithm for the Traveling Salesman Problem». *Operations Research* 21, pp. 498-516.

[31] Osman, I.H., 1993. «Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem». *Annals of Operations Research* 41, pp. 421-451.

[32] Pureza, V.M., Franca, P.M., 1991. «Vehicle Routing Problems via Tabu Search Metaheuristics». Rapport Technique CRT-347, Centre de Recherche sur les Transports, Montréal, Canada.

[33] Rego, C., 1998a. «Relaxed Tours and Path Ejections for the Traveling Salesman Problem». *European Journal of Operational Research* 106, pp. 522-538.

[34] Rego, C., 1998b. «A Subpath Ejection Method for the Vehicle Routing Problem». *Management Science* 44, pp. 1447-1459.

[35] Rego, C., 2001. «Node-ejection Chains for the Vehicle Routing Problem : Sequential and Parallel Algorithms». *Parallel Computing* 27, pp. 201-222.

[36] Rego, C., Roucairol, C., 1996. «A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem». Osman, I. H., Kelly, J. P. (eds.), *Metaheuristics : Theory and Applications*, Kluwer, Boston, pp. 661-675.

[37] Renaud, J., Boctor, F.F., Laporte, G., 1996. «A Fast Composite Heuristic for the Symmetric Travelling Salesman Problem». *INFORMS Journal on Computing* 8, pp. 134-143.

[38] Rochat, Y., Taillard, E.D., 1995. «Probabilistic Diversification and Intensification in Local Search for Vehicle Routing». *Journal of heuristics* 1, pp. 147-167.

[39] Rousseau, L.-M., Gendreau, M., Pesant, G., 2002. «Using Constraint-based Operators to Solve the Vehicle Routing Problem with Time Windows». *Journal of Heuristics* 8, pp. 43-58.

[40] Sontrop, H.M.J., van der Horn, S.P., Teeuwen, G., & Uetz, M., 2005. «Fast Ejection Chain Algorithms for Vehicle Routing with Time Windows». Blesa, M.J., Blum, C., Roli, A., Sampels, M. (eds.), *Lecture Notes in Computer Science* 3636. Springer, Berlin, pp. 78-89.

[41] Taillard, E.D., 1993. «Parallel Iterative Search Methods for Vehicle Routing Problems». *Networks* 23, pp. 661-673.

[42] Taillard, E.D., 1999. «A Heuristic Method for the Heterogeneous VRP». *Operations Research* 33, pp. 1-14.

[43] Taillard, E.D., Gambardella, L.M., Gendreau, M., Potvin, J.-Y., 2001. «Adaptive Memory Programming : A Unified View of Metaheuristics». *European Journal of Operations Research* 135, pp. 1-16.

[44] Tarantilis, C.D., 2005. «Solving the Vehicle Routing Problem with Adaptive Memory Programming Methodology». *Computers & Operations Research* 32, pp. 2309-2327.

[45] Toth, P., Vigo, D., 1998. «The Granular Tabu Search (and Its Application to the Vehicle Routing Problem)». Technical Report OR/98/9, DEIS, Università di Bologna, Italy.

[46] Volgenant, T., Jonker, R., 1987. «On Some Generalization of the Traveling-Salesman Problem». *Journal of the Operational Research Society* 38, pp. 1073-1079.

[47] Willard, J.A.G., 1989. «Vehicle Routing Using r-optimal Tabu Search». Mémoire de maîtrise, The Management School, Imperial College, London.

[48] Xu, J., Kelly, J.P., 1996. «A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem». *Transportation Science* 30, pp. 379-393.