INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

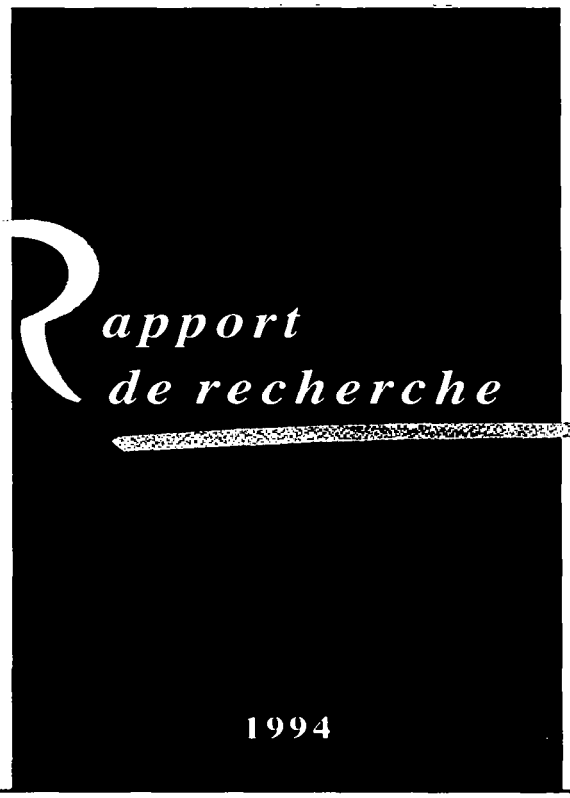
Le problème de tournées de véhicules : étude et résolution approchée

César REGO
Catherine ROUCAIROL

N° 2197
Février 1994

PROGRAMME 1

Architectures parallèles,
bases de données,
réseaux et systèmes distribués



INRIA

Le Problème de Tournées de Véhicules: Etude et Résolution Approchée

César REGO* et Catherine ROUCAIROL**

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet Reflecs

Rapport de recherche n° 2197 — Février 94 — 90 pages

Résumé : Dans ce rapport, on introduit un nouvel algorithme de type "tabou" pour le problème d'optimisation de tournées de véhicules avec contraintes de capacité et de distance. L'algorithme considère des solutions adjacentes obtenues par une suite de déplacements de sommets entre tournées. On utilise ici une stratégie de chaîne d'éjection pour générer des mouvements composés permettant le passage d'une solution à une autre. Au cours de l'algorithme, sont mises en œuvre plusieurs types de mouvements et le passage par des solutions irréalisables est permis.

Les résultats numériques effectués sur un ensemble de problèmes classiques de la littérature indiquent que l'algorithme proposé est compétitif avec plusieurs autres méthodes.

Mots-clé : problème d'optimisation de tournées de véhicules, algorithme de type tabou, stratégie de chaîne d'éjection, mouvements composés.

(Abstract: pto)

*rego@eden.inria.fr
**roucairo@eden.inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 - Télécopie : (33 1) 39 63 53 30

The Vehicle Routing Problem: Study and Approximated Resolution.

Abstract: The purpose of this report is to describe a new tabu search heuristic for the vehicle routing problem with capacity and route length restrictions. The algorithm considers a sequence of adjacent solutions obtained by repetedly removing a vertex from its current route, and reinserting it into another route. This is done by means of an ejection chain strategy which produces compound moves for passing from a current solution to another solution. During the execution of the algorithm, many moves are used and crossing infeasible solutions are allowed.

Numerical tests on a set of a benchmark problems indicate that proposed algorithm is highly competitive with existing heuristics.

Key-words: tabu search, vehicle routing problem, ejection chain strategy, compound moves

Table des matières

Chapitre 1

Introduction

L'une des principales préoccupations des entreprises industrielles est d'améliorer l'efficacité de leurs chaînes logistiques, pour pouvoir organiser au moindre coût un meilleur service et la fluidité de l'écoulement de leurs marchandises. Ainsi un élément fondamental de tout système logistique, est la gestion et la planification des réseaux de distribution des flottes de véhicules.

L'utilisation de modèles mathématiques d'optimisation des tournées de véhicules, a été l'un des plus beaux succès de la Recherche Operationnelle au cours de la dernière décade. Les recherches récentes incluent des efforts significatifs en formulation du problème et en construction, analyse et implantation d'algorithmes. Une collaboration profitable entre spécialistes de la Programmation Mathématique et de l'Optimisation Combinatoire d'une part et les gestionaires de transports d'autre part, s'est traduite par un grand nombre d'implantations réussies de systèmes informatiques d'optimisation de tournées de véhicules (Golden et Assad, 1988, [41]).

L'intérêt du recours aux méthodes quantitatives pour l'optimisation des activités de transport, devient évident dès lorsque que l'on se rend compte de l'importance des coûts de distribution. Dans un rapport préparé pour le "National Council of Physical Distribution Management (NCPDM)", Kearney, 1984, [48] estime les coûts annuels de distribution à 650 milliards de dollars, environ 21% du chiffre d'affaires, rien que dans un seul pays, comme les Etats Unis. De plus, le rapport Kearney indique que la somme des coûts logistiques représente 22.5% des coûts de production. Cette somme est très importante, et est souvent une charge extrêmement lourde pour des entreprises appartenant à presque tous les secteurs industriels.

Les applications pratiques de ce type de problèmes incluent en effet: la distribution de journaux (Golden, 1975, [40]), le ramassage scolaire (Bennett et al., 1972, [7]), la collecte d'ordures (Beltrami et Bodin, 1974, [5]), la fourniture de combustible (Garvin et al., 1957, [29]), la distribution de produits aux hypermarchés et magasins, laveries, la distribution de courrier et la gestion préventive d'inspection de routes...

Outre leur intérêt pratique, les modèles de tournées de véhicules ont été également appliqués à d'autres domaines, n'ayant apparament aucun rapport. Des applications effectives ont été faites en ordonnancement d'atelier, où certaines situations peuvent être modélisés d'une manière identique à celle des problèmes de tournées. Ainsi, à titre d'exemple, le problème de la recherche de de séquences optimales de production sur plusieurs machines parallèles avec des coûts de lancements variables, a été résolu par Parker et al., 1977, [72] à l'aide d'un algorithme de résolution de problèmes de tournées. De même, des problèmes de fabrication d' horaires de travail (Desrochers, 1986, [20]), ou d'optimisation de réseaux de communication (Sanso et al., 1988, [76]) ont été également résolus par des algorithmes conçus à l'origine pour la résolution de problèmes de tournées.

Les modèles d'optimisation de tournées de véhicules ont également une importance théorique fondamentale. Le modéle de base est appelé "Problème du Voyageur de Commerce". Il consiste à trouver la tournée optimale d'un commis voyageur désirant visiter un ensemble de clients (le nom original de ce problème serait dû probablement à A.W. Tucker, et daterait du début des années 30, voir à ce propos le première chapitre de l'ouvrage de Lawler et al., 1985, [12]). Ce problème se trouve au cœur même, d'une importante branche des Mathématiques Discrètes, l'Optimisation Combinatoire (Christofides et al., 1979, [15]).

La recherche de méthodes efficaces de résolution des problèmes de tournées de véhicules, a été à l'origine d'importants développements en Programmation Mathématique et en Optimisation Combinatoire, par la mise au point, l'analyse, et l'implantation d'algorithmes et d'heuristiques plus performants pour résoudre des problèmes toujours plus complexes.

Les techniques qui ont été mise en œuvre pour résoudre ces problèmes sont de deux types :

- celles qui résolvent le problème jusqu'à l'optimalité fondées sur des techniques de partitionement et d'évaluation successives (*Branch & Bound*),

- celles qui résolvent le problème heuristiquement.

Comme les algorithmes exacts ne sont applicables qu'à des problèmes de taille souvent trop petite, nous nous concentrons plutôt sur les algorithmes heuristiques. Récemment, des heuristiques d'optimisation locale telles que le Recuit Simulé *(Simulated Annealing)*, les Algorithmes Génétiques *(Genetic Algorithms)* et la Méthode Tabou *(Tabu Search)* ont permis d'obtenir de très bons résultats sur des problèmes d'Optimisation Combinatoire connus comme difficiles à résoudre.

Après avoir presenté ces deux premières types d'heuristiques, nous etudions la méthode Tabou d'une façon beaucoup plus détailé. Un nouvel algorithme de type Tabou pour la construction de tournées de véhicules avec des contraintes de capacité et de distance et ensuite introduit.

Cet algorithme considère des solutions adjacentes obtenues par une suite de déplacements (mouvements) d'un sommet d'une tournée vers une autre tournée. Au cours de l'algorithme, sont mises en œuvre plusieurs types de mouvements et le passage par des solutions irréalisables est permis.

L'étude de structures de voisinages composés pour définir des mouvements fondés sur l'idée de *chaînes d'éjection* est originale et réprésente une contribution importante de ce rapport.

Les résultats des tests sur plusieurs problèmes de la littérature sont commentés et discutés. Ils prouvent la viabilité de ce type d'approche.

Chapitre 2

Le problème de tournées de véhicules

Le problème de tournées a plusieurs variantes (voir Bodin et al.,1981, [8] pour un tour d'horizon des problèmes de tournées de véhicules). Nous nous intéresserons d'une manière prioritaire au problème qui consiste à optimiser la gestion d'une flotte homogène de véhicules de capacité finie, domiciliés dans un même dépôt, et visitant un ensemble de clients géographiquement dispersés, qui formulent une demande (ou une offre, mais pas les deux à la fois) connue. Le but est de minimiser la distance totale parcourue ou d'autres mesures telles que des coûts, le temps, ...

Nous désignerons ce problème par VRP (*Vehicle Routing Problem*), qui est le nom générique le plus fréquemment utilisé dans la littérature scientifique. Le VRP est aussi connu sous les noms de *"Vehicle Scheduling"* (Clark et Wright, 1964, [18]; Gaskel, 1967, [30]), *"Vehicle Dispaching"* (Christofides et Eilon, 1969, [13]; Dantzig et Ramser, 1959, [19]; Gillet et Miller, 1974, [33]), ou simplemment *"Delivery Problem"* (Hays, 1967, [45]).

Le VRP a été prouvé par Lenstra et Rinnooy Kan, 1981, [57] comme étant NP-difficile, c'est-à-dire qu'il n'existe aucun algorithme connu polynomial en temps pour résoudre le problème. Les algorithmes existants permettent de résoudre des problèmes comportant un peu plus de 25 sommets, mais dès que le problème devient plus grand, les temps de calcul deviennent prohibitifs. A cause du succés limité des méthodes exactes beaucoup d'attention a été portée dans la littérature au développement d'algorithmes heuristiques.

Dans ce chapitre, nous commençons par formuler mathématiquement le problème de tournées de véhicules comme un problème du voyageur de commerce généralisé. Puis nous présenterons des méthodes de résolution exactes et approchées.

2.1 Formulations mathématiques

2.1.1 Le problème du voyageur de commerce

Plusieurs modèles de tournées de véhicules sont des variantes et extensions du fameux problème du voyageur de commerce (TSP) (pour une vision globale de ce problème, voir Bellmore et Nemhauser, 1968, [4]).

Le TSP consiste en la détermination du parcours de coût minimal (distance, temps, etc), d'un seul véhicule partant d'une localité, visitant $n-1$ autres localités, et revenant à son départ.

En théorie des graphes, le problème consiste à chercher le circuit hamiltonien de coût minimal dans un graphe complet $G = (V, A)$, valué (coût c_{ij} associé à chaque arc (i, j) de A).

Nous utiliserons les notations suivantes:

n = $\mid V \mid$, nombre de sommets du réseau,
c_{ij} = coût de l'arc (i, j),
b_j = nombre des arcs incidents au sommet j,
a_i = nombre des arcs partant du sommet i,

$$x_{ij} = \begin{cases} 1 & \text{si l'arc } (i,j) \text{ appartient à la tournée,} \\ 0 & \text{sinon.} \end{cases}$$

Une formulation fondée sur l'affectation consiste à rechercher la matrice $X = (x_{ij})$ de variables de décision en résolvant le problème de programmation linéaire en nombres entières suivant:

$$\text{Minimiser} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.1}$$

sous

$$\sum_{i=1}^{n} x_{ij} = b_j = 1 \qquad j = 1, \cdots, n \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = a_i = 1 \qquad i = 1, \cdots, n \tag{2.3}$$

$$X = (x_{ij}) \in S \tag{2.4}$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, \cdots, n \tag{2.5}$$

L'ensemble S est choisi de façon à interdire les solutions qui sont des sous-tours (tours ne passant que par un sous-ensemble de n clients) satisfaisant les contraintes

d'affectation (2.2), (2.3) et (2.5). Dans la littérature, il a été proposé pour S, les ensembles suivants:

$$S = \left\{(x_{ij}) \mid \sum_{i \in Q} \sum_{j \notin Q} x_{ij} > 1, \forall Q \subset V\right\} \tag{2.6}$$

$$S = \left\{(x_{ij}) \mid \sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq \mid Q \mid -1, \forall Q \subset V\right\} \tag{2.7}$$

$$S = \{(x_{ij}) \mid y_i - y_j + n x_{ij} \leq n - 1, \forall y_i \in R; i, j \in V; i \neq j\} \tag{2.8}$$

On peut remarquer que S contient environ 2^n contraintes d'élimination de sous-tours dans les cas (2.6) et (2.7), mais seulement $n^2 - 3n + 2$ contraintes dans l'ingénieuse formulation (2.8) proposé par Miller, Tucker et Zemlin, 1960, [63]. L'utilisation de l'un au l'autre de ces ensembles dépend essentiellement de la méthode adoptée pour la résolution du probléme.

Algorithmiquement, les contraintes (2.7) ont été trés utilisées, principalement dans les approches Lagrangiennes pour le TSP, proposées initialement par Held et Karp, 1970, [46]. On trouvera d'autres approches autant heuristiques qu'optimales dans les articles suivants: Lin, 1965, [58], Lin et Kernighan, 1973, [59], Christofides et Eilon, 1972, [14] et Little et al., 1963, [60].

Le problème du voyageur de commerce peut être interprété comme une tournée de véhicules avec un seul dépôt et un seul véhicule de capacité supérieure à la totalité de la demande. Ce modèle peut être étendu à un nombre plus grand de véhicules. On peut aussi considérer des contraintes de capacité, de longueur de tournée, ...

2.1.2 Le problème des m voyageurs de commerce

Le problème des m voyageurs de commerce (m-TSP) est une généralisation du TSP au cas où on désire construire m circuits hamiltoniens de coût total minimal, ayant un sommet commun (le dépôt). Ce problème peut être considèré comme une version simplifiée du VRP où la capacité de chaque véhicule est supérieure à la totalité de la demande (ou considèrée infinie) et où les m véhicules sont utilisés.

En posant $a_1 = b_1 = m$ et $a_i = b_i = 1$ $(i \neq 1, j \neq 1)$, dans le modèle (2.1)-(2.5), on obtient la formulation du m-TSP. Trois articles publiés indépendament, Bellmore et al., 1974, [3], Orloffe, 1974, [68] et Svestka et al., 1973, [79], obtiennent des formulations équivalentes pour le m-TSP, dérivées de formulations pour le TSP et en conséquence prouvent que le m-TSP n'est pas plus difficile que le 1-TSP. L'équivalence s'obtient en créant m copies du point de départ, chacune connectée à tous les autres sommets exactement comme si c'était une nouvelle origine (avec les mêmes distances). Les m copies du point de départ ne sont pas connectées, ou sont

connectées par des arcs dont les distances excédent la totalité des distances entre sommets.

2.1.3 Le problème de tournées de véhicules

Le problème de tournées de véhicules (VRP) a été initialement considèré par Dantzig et Ramser, 1959, [19], qui ont développé une approche heuristique en utilisant des idées de programmation linéaire et d'agrégation de sommets. Le problème est une généralisation du m-TSP au cas où chaque sommet est affecté d'un poids (demande) connu, et où la somme totale des poids des sommets appartenant à un même circuit doit être bornée par une constante (capacité). De plus, on considère parfois que la somme totale des poids de ces arcs est aussi bornée par une autre constante (temps maximal par tournée).

Une formulation mathématique pour le problème qui sera traité dans ce mémoire est présentée ci-après.

Les notations sont:

- Constantes:

n = nombre de sommets
m = nombre de véhicules
D_k = capacité du véhicule k
T_k = temps maximal de parcours du véhicule k
d_i = demande du sommet i ($d_1 = 0$)
t_i^k = temps nécessaire au véhicule k pour décharger ou charger dans le sommet i ($t_1^k = 0$)
t_{ij}^k = temps nécessaire au véhicule k pour traverser l'arc (i,j) ($t_{ii}^k = \infty$)
c_{ij} = coût ou distance de sommet i au sommet j
X = matrice d'éléments $x_{ij} = \sum_{k=1}^{m} x_{ij}^k$

- Variables de décision:

$$x_{ij}^k = \begin{cases} 1 & \text{si (i,j) est parcouru par le véhicule k,} \\ 0 & \text{sinon.} \end{cases}$$

$$\text{(VRP1)} \quad \text{Minimiser} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \sum_{k=1}^{m} x_{ij}^k \tag{2.9}$$

sous

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ij}^k = 1 \qquad j = 2, \cdots, n \tag{2.10}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{ij}^{k} = 1 \qquad i = 2,\cdots,n \tag{2.11}$$

$$\sum_{i=1}^{n} x_{ip}^{k} - \sum_{j=1}^{n} x_{pj}^{k} = 0 \qquad k = 1,\cdots,m; p = 1,\cdots,n \tag{2.12}$$

$$\sum_{i=1}^{n} d_i \left(\sum_{j=1}^{n} x_{ij}^{k} \right) \leq D_k \qquad k = 1,\cdots,m \tag{2.13}$$

$$\sum_{i=1}^{n} t_i^k \sum_{j=1}^{n} x_{ij}^{k} + \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij} x_{ij}^{k} \leq T_k \qquad k = 1,\cdots,m \tag{2.14}$$

$$\sum_{j=2}^{n} x_{1j}^{k} \leq 1 \qquad k = 1,\cdots,m \tag{2.15}$$

$$\sum_{i=2}^{n} x_{i1}^{k} \leq 1 \qquad k = 1,\cdots,m \tag{2.16}$$

$$X = (x_{ij}) \in S \tag{2.17}$$

$$x_{ij}^{k} \in \{0,1\} \qquad i,j = 1,\cdots,n \tag{2.18}$$

La fontion *Objectif* (2.9) consiste à minimiser le coût total de transport. Les équations (2.10) et (2.11) assurent que chaque sommet de demande soit servi par un seul véhicule. La continuité d'une tournée est répresentée par les équations (2.12): si un véhicule entre dans un sommet, il faudra qu'il en sorte. Les équations (2.13) sont les contraintes de capacité d'un véhicule; de façon similaire, les équations (2.14) réprésente la contante de temps total par tournée. Les équations (2.15) et (2.16) assurent que la disponibilité d'un véhicule n'est pas dépassée. Finalement, la contrainte d'élimination des sous-tours peut être exprimée par une des équations spécifiées ci-dessus.

Une revue exhaustive des problèmes de tournées de véhicules est donnée par Bodin et al.,(1983).

2.2 Les méthodes de résolution

Le VRP appartient au noyau des problème de gestion de distribution et a été intensivement étudié depuis les trois dernières décades (voir Christofides, Mingozzi et Toth, 1979, [15]; Bodin, Golden, Assad et Ball, 1983, [9]; Christofides, 1985, [12]; Laporte et Nobert, 1987, [53]; Golden et Assad, 1988, [41].

Le VRP appartient également à la grande famille des problèmes combinatoires NP-difficiles (Lenstra et Rinnoy Kan, 1981, [57]), et jusqu'à présent il n'est possible de résoudre d'une manière exacte que des petites instances du problème.

Pour un tour d'horizon des algorithmes exacts et approchés, voir Laporte, 1992, [51].

2.2.1 Méthodes exactes

Une méthode est exacte si elle trouve la solution optimale du problème. D'après Laporte et Nobert, 1987, [53], les méthodes exactes pour résoudre le VRP sont généralement de trois types:

- les méthodes de recherche arborescente (*Branch & Bound*),
- la programmation dynamique,
- la programmation linéaire en nombres entiers.

Comme le nombre d'algorithmes exactes proposés est trop grand, on ne va donner que trois exemples réprésentatifs de ce type d'algorithmes.

Un algorithme Branch & Bound

L'algorithme suivant dévéloppé par Laporte, Mercure et Nobert, 1986, [52] exploite les ressemblances entre le VRP et le m-TSP. Soit $G = (V, A)$ un graphe dont $V = \{v_1, v_2, \cdots, v_n\}$ réprésente l'ensemble des sommets avec le dépôt localisé en v_1 et $A = \{(v_i, v_j) \mid v_i, v_j \in V; i \neq j\}$ est l'ensemble des arcs. Une matrice de distances $C = (c_{i,j})$ est associée à A.

D'après Lenstra et Rinnooy Kan, 1975, [56], si m_{max} est une borne supérieure de m, le m-TSP peut se transformer dans un 1-TSP de la façon suivante.

1. Augmenter le nombre de sommets par introdution de $m_{max} - 1$ dépôts artificiels;
   définir $n' = n + m_{max} - 1$; $V' = \{v_1, v_2, \cdots, v_{n'}\}$ et
   $A' = A \cup \{(v_i, v_j) \mid v_i, v_j \in V', v_i \neq v_j, v_i \text{ ou } v_j \in V' \setminus V\}$.
2. Définir la matrice $C' = (c'_{ij})$ associée à A' telle que :

$$c'_{ij} = \begin{cases} c_{ij} & (v_i, v_j \in V) \\ c_{i1} & (v_i \in V \setminus \{v_1\}, v_j \in V' \setminus V) \\ c_{1j} & (v_i \in V' \setminus V, v_j \in V \setminus \{v_1\}) \\ \gamma & (v_i, v_j \in (V' \setminus V) \cap \{v_1\}) \end{cases}$$

où γ est un facteur de pondération entre la minimisation de la distance et le nombre de véhicules :

$\gamma = \infty$ correspond à la recherche d'une distance minimale pour m_{max} véhicules,

$\gamma = 0$ d'une distance minimale pour plus de m_{max} véhicules,

$\gamma = -\infty$ d'une distance minimale pour un nombre minimal de véhicules.

Le VRP peut donc être formulé de la façon suivante :

$$x_{ij} = \begin{cases} 1 & \text{si l'arc } (i,j) \in A' \text{ apparaît dans la solution optimale,} \\ 0 & \text{sinon.} \end{cases}$$

$$\text{(VRP2)} \quad \text{Minimiser} \quad \sum_{i=1}^{n'} \sum_{j=1}^{n'} c'_{ij} x_{ij} \tag{2.19}$$

sous

$$\sum_{i=1}^{n'} x_{ij} = 1 \qquad j = 1, \cdots, n' \tag{2.20}$$

$$\sum_{j=1}^{n'} x_{ij} = 1 \qquad i = 1, \cdots, n' \tag{2.21}$$

$$\sum_{i,j \in S} x_{ij} \leq \mid S \mid -v(S) \qquad S \subset V \setminus \{v_1\}; S \geq 2 \tag{2.22}$$

$$x_{ij} \in \{0,1\} \qquad i,j = 1, \cdots, n'; i \neq j. \tag{2.23}$$

Dans cette formulation, les équations (2.19), (2.20), (2.21) et (2.23) définissent un problème d'affectation modifié (les affectations dans la diagonale principale sont interdites). Les contraintes d'élimination de sous-tours sont exprimées par l'équation (2.22): $v(S)$ est une borne inférieure pour le nombre de véhicules dans la solution optimale. Cela assure qu'aucun sommet n'est déconnecté de $V' \setminus V$ dans la solution optimale et la capacité du véhicule qui parcourt les sommets du sous-ensemble S de $V \setminus \{v_1\}$ ne sera jamais d épassée. Ces contraintes sont fondées sur les contraintes de connectivités suivantes :

$$\sum_{i \in V' \setminus S} \sum_{j \in S} x_{ij} \geq v(S), \qquad \forall S \subset V \setminus \{v_1\}, \mid S \mid \geq 2.$$

avec

$$\mid S \mid = \sum_{i,j \in S} x_{ij} + \sum_{i \in S} \sum_{j \in V' \setminus V} x_{ij}.$$

La valeur de $v(S)$ dépend du type de VRP considèré. Si on ne considère que les contraintes de capacité, on a:

$$v(S) = \left\lceil \frac{\sum_{i \in S} d_i}{D} \right\rceil$$

où

$$\lceil t \rceil = \begin{cases} \text{le plus petit entier supérieur ou égal à } t \text{ si } t \geq 1, \\ 1 \text{ sinon.} \end{cases}$$

Si l'on considère aussi les contraintes de distances, $v(S)$ ne peut pas être facilement déterminé d'avance. Il faut alors qui $v(S)$ soit déterminé au cours de l'algorithme Branch & Bound.

Les solutions optimales du (VRP2) sont obtenues par la résolution d'un algorithme du type Branch & Bound dont les sous-problèmes sont problèmes d'affectation. La méthode considère la relaxation du problème d'affectation modifié (MAP) et utilise cette relaxation jusqu'à ce que toutes les tournées soient admissibles.

Les sous-tours pour lesquels l'ensemble des tournées est inclus dans $V' \setminus V$ n'ont pas besoin d'être éliminés.

La solution du VRP est obtenue à partir de la solution de (VRP2) selon les règles suivantes:

1. si $i \in V \setminus \{v_1\}$ et $v_j \in V' \setminus V$, alors remplacer (v_i, v_j) par (v_i, v_1),
2. si $i \in V' \setminus V$ et $v_j \in V \setminus \{v_i\}$, alors remplacer (v_i, v_j) par (v_1, v_j),
3. si $v_i, v_j \in V' \setminus V$, alors enlever (v_i, v_j).

Pour plus de details, voir aussi Laporte, 1992, [51].

Programmation dynamique

Il est parfaitement connu que seuls un très petit nombre de problèmes d'Optimisation Combinatoire peuvent être résolus par la programmation dynamique. L'exploision du nombre d'états rend l'utilisation de cette technique impossible suit à des en temps de calcul prohibitifs .

La programmation dynamique pour des VRPs a été initialement proposée par Eilon, Watson-Gandy et Christofides, 1971, [24].

Soient:

m	–	le nombre de véhicules,
$c(S)$	–	le coût d'une tournée comprenant un sous-ensemble de sommets $S \subset V \setminus \{v_1\}$,
$f_k(U)$	–	le coût minimal correspondant à l'utilisation de k véhicules pour livrer un sous-ensemble $U \subset V \setminus \{v_1\}$,

alors le coût mininal peut se déterminer à travers la relation de récurrence suivante:

$$f_k(U) = \begin{cases} c(U) & k = 1 \\ \min\limits_{U^* \subset U \subseteq V \setminus \{v_1\}} [f_{k-1}(U \setminus U^*) + c(U^*)] & k > 1. \end{cases} \quad (2.24)$$

Le coût de la solution est égal à $f_m(V \setminus \{v_1\})$ et la solution optimale correspond aux sous-ensembles optimisés U^* en 2.24.

Il devient évident que si on calcule $f_k(U)$ pour tous les valeurs de k et tous les sous-ensembles U de $V \setminus \{v_1\}$, le temps devient excessifs pour la plupart des problèmes. La relaxation de l'espace d'états est une méthode efficace de réduction du nombre d'états.

Cette technique a été introduite par Christofides, Mingozzi et Toth, 1981, [17], elle fournit une borne inférieure de la solution optimale. Cette borne peut alors, être utilisée dans une procédure de séparation et évaluation progressive (Branch & Bound) dans des méthodes de relaxation, ou pour des critères de dominance. Par exemple, dans le cas du VRP, les ensembles U et U^* satisfont les relations suivantes:

$$\sum_{i \in V \setminus \{v_1\}} d_i - (m-k)D \leq \sum_{i \in U} \leq k\ D \qquad k = 1, \cdots, m \quad (2.25)$$

et

$$\sum_{i \in U} d_i - (k-1)D \leq \sum_{i \in U^*} \leq D \qquad k = 1, \cdots, m \quad (2.26)$$

L'idée de relaxation d'espace d'états et la suivante:

Considérons la relation de recurrence d'un programme dynamique quelconque:

$$\text{(DP)} \quad f_{0,i}(0,j) = \min_{k \in \Omega^{-1}(j)} [f_{0,i-1}(0,k) + c_i(k,j)]\,, \quad (2.27)$$

où

- $f_{0,i}(0,j)$ est le coût pour faire passer le système de l'état 0 à l'étape 0, à l'état j à l'étape i,
- $\Omega^{-1}(j)$ est l'ensemble de tous les états à partir desquels on peut atteindre directement l'état j,
- $c_i(k,j)$ est le coût pour faire passer le système de l'état k à l'étape $i-1$, à l'état j à l'étape i.

Soient, $g(.)$ une application définie de S vers un espace d'états G de cardinalité beaucoup plus faible que celle de S, et soit un ensemble :

S	l'espace d'état associé au problème (2.27),
$g(.)$	une application définie de S dans un espace d'états G de cardinalité beaucoup plus faible que celle de S,
$F^{-1}(g(j))$	un ensemble satisfaisant:,

$$k \in \Omega^{-1}(j) \Rightarrow g(k) \in F^{-1}(g(j))$$

La relation de recurrence (2.27) devient alors:

$$f_{0,i}(g(0), g(j)) = \min_{t \in F^{-1}(g(j))} [f_{0,i-1}(g(0), t) + c_i^*(t, g(i))], \tag{2.28}$$

où

$$c_i^*(t, g(j)) = \min[c_i(k, l) \mid g(k) = t, g(j) = g(l)]$$

Il en résulte la relation:

$$f_{0,i}(g(0), g(i)) \leq f_{0,i}(0, i) \tag{2.29}$$

Cette relation montre que la formule de recurrence appliquée dans l'espace d'état G, qui est "l'image" de S par $g(.)$, fournit une borne inférieure de la solution optimale.

En générale, la relaxation de l'espace d'état est intéressante, si l'application $g(.)$ satisfait les conditions minimales suivantes:

1. $F^{-1}(.)$ est facile à déterminer; cela se vérifie si $g(.)$ est séparable et on peut alors calculer $g(U)$ et $r, g(U \setminus \{r\})$.
2. l'optimisation de (2.29) doit être faite sur un domaine réduit, où la détermination d'une borne inférieure de $c_i^*(t, g(j))$ doit être aisée.

Christofides, Mingozzi et Toth, 1981, [17] ont utilisé la relaxation pour le VRP suivant.

Soit $f_k(U, r)$ le coût le plus petit attribué à l'ensemble U de sommets, en utilisant k véhicules, les derniers sommets etant numérotés $\{2, \cdots, r\}$
$(k \leq r \leq n)$.

Soit $c(U, r)$ le coût du TSP correspondant à l'ensemble des sommets $U \cup \{v_1\}$, où r est le dernier sommet avant le dépôt. La formule de récurence est alors:

$$f_k(U) = \begin{cases} \min\left[f_k(U, r-1), \min_{u^* \in U}\{f_{k-1}(U \setminus U^*, r-1) + c(U^*, r)\}\right], & k, r > 1 \\ c(U, r), \quad k = 1 \end{cases}$$

sous la contrainte (2.25).

Pour ce problème, l'application $g(.)$ est donnée par:

$$g(U) = \sum_{i \in U} d_i.$$

et la récurrence (2.28) devient:

$$f_k(g(U), r) = \min \left[f_k(g(U), r-1), \min_p \{ f_{k-1}(g(U) - p, r-1) + c^*(p, r) \} \right] \tag{2.30}$$

sous la contrainte:

$$g(N) - (m-1)D \leq p \leq \min(g(U), D). \tag{2.31}$$

En utlisant cette relaxation et d'autres relaxations décrites dans Christofides, Mingozzi et Toth, 1981, [16], des bornes inférieures ont été obtenues pour dix problèmes de 10 à 25 sommets. Plus récemment Christofides, 1985, [12] a donné la résolution exacte d'un VRP particulier à 53 sommets tenant compte de contraintes additionelles telles que véhicules de différentes capacitées, fenêtres de temps...

Cet auteur suggère que des améliorations pour ce type de relaxation de l'espace d'état peuvent êtres obtenues par des méthodes de pénalitées comme la relaxation Lagrangiennes (en utilisant l'optimisation de sous-gradient).

Programmation en nombres entiers

Fisher et Jaikumar, 1978, [26], 1981, [27] ont développé une formulation en terme de flôts de VRPs avec contraintes de capacité et fenêtres de temps (le sommet v_i doit être visité dans un intervalle de temps $[a_i, b_i]$). Cette formulation utilise des variables à trois indices pour représenter le passage d'un véhicule par un arc (v_i, v_j).

Les notations sont partiellement données dans le paragraphe 2. De plus sont introduites les variables de décision suivantes:

$$y_i^k = \begin{cases} 1 & \text{si le sommet i est visité par le véhicule k,} \\ 0 & \text{sinon.} \end{cases}$$

$$\text{(VRP3)} \quad \text{Minimiser} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \sum_{k=1}^{m} x_{ij}^k \tag{2.32}$$

sous

$$\sum_{i=1}^{n} d_i y_i^k \leq D_k \qquad k = 1, \cdots, m \tag{2.33}$$

$$\sum_{k=1}^{m} y_i^k = \begin{cases} m & i = 1 \\ 1 & i = 2, \cdots, n \end{cases}$$

$$\sum_{i=1}^{n} x_{ij}^k = y_i^k \qquad j = 1, \cdots, n; k = 1, \cdots, m \tag{2.34}$$

$$\sum_{j=1}^{n} x_{ij}^k = y_i^k \qquad i = 1, \cdots, n; k = 1, \cdots, m \tag{2.35}$$

$$\sum_{i,j \in S} x_{ij}^k \leq \mid S \mid -1 \qquad S \subset V; \mid S \mid \geq 2; k = 1 \cdots, m \tag{2.36}$$

$$t_j \geq t_i + t_{ij} - (1 - x_{ij}^k)M$$

$$t_j \leq t_i + t_{ij} + (1 - x_{ij}^k)M \qquad i, j = 1, \cdots; k = 1, \cdots, m \tag{2.37}$$

$$a_i \leq t_i \leq b_i \qquad i = 2, \cdots, n \tag{2.38}$$

$$x_{ij}^k \in \{0, 1\} \qquad i, j = 1, \cdots, n; k = 1, \cdots, m \tag{2.39}$$

$$y_i^k \in \{0, 1\} \qquad i, \cdots, n; k = 1, \cdots, m \tag{2.40}$$

où M est un nombre arbitraire suffisament grand.

Dans cette formulation sont inclus deux problèmes d'Optimisation Combinatoire bien connus :

1. le problème d'affectation généralisé (GAP) obtenu en relaxant les contraintes (2.36), (2.37) et (2.38),
2. le problème du voyageur de commerce avec des fenêtres de temps (TSP-TW) si on fixe y_i^k pour vérifier les contraintes (2.34)-(2.39) pour un véhicule k donné.

Les auteurs ont proposé un algorithme fondé sur la décomposition de Benders.

La décompositon de Benders, appelée également décompositon par partitionnement des variables, s'applique aux programmes mathématiques ne comportant que des variables couplantes, c'est a dire, ayant la structure générale suivante:

$$\text{Minimiser} \qquad Z = CX + FY$$

sous

$$AX + BY \geq D$$
$$X > 0, Y \in S$$

où A est une matrice (m,n), B est une matrice (m,p), C et X sont des n-vectors, Y et F sont des p-vecteurs, D est un m-vecteur, et S est un sous-ensemble de R^p.

A l'origine, cette méthode a été utilisée pour résoudre des problèmes mixtes avec certaines variables continues et d'autres discrètes (Benders, 1962, [6]. L'idée générale de la méthode consiste à fixer la valeur de Y, puis à résoudre le programme linéaire en X, obtenant ainsi une "meilleure" valeur de Y... Pour une présentation détaillée de l'algorithme, on pourra se référer à l'ouvrage classique de Lasdon, 1970, [55] ou à Minoux, 1983, [64].

L'algorithme de Fisher et Jaikumar résout tour à tour, un problème d'affectation généralisée (GAP) pour affecter des sommets à des tournées, et un problème du voyageur de commerce avec des fenêtres de temps (TSPTW) pour déterminer la meilleure tournée pour chaque véhicule.

Chaque TSPTW est résolu par une méthode de coupes dans le but d'obtenir des valeurs duales optimales définissant un sous-gradient du coût optimal de la tournée. Par la suite, ils résolvent le GAP en approximant la fonction *Objectif* par une combinaison linéaire des sous-gradients. Ils réitèrent de cette manière jusqu'à ce qu'ils aboutissent à une solution optimale ou sub-optimale.

Sur le plan théorique, cette approche peut être intéressante pour résoudre des problèmes de tournées plus complexes. Plusieurs types de contraintes suplémentaires de type "sac à dos" (Knapsack) peuvent être intégrées dans la sous-structure GAP, voir par exemple Martello et Toth, 1990, [62] et Desrochers, Desrosiers et Solomon, 1992,[21] . A titre d'exemple, nous pouvons citer le cas où pour un véhicule donné, il existe plusieurs contraintes de capacités, relatives à divers types de produits transportés. D'autres types de contraintes peuvent également être intégrées dans la sous-structure TSPTW. Ainsi, si les tournées doivent vérifier des contraintes de précédence (des clients doivent être visités avant d'autres), nous aurons à résoudre un problème du voyageur de commerce avec des contraintes de fenêtres de temps et de précédence.

Christofides et al., 1981, [16], Laporte et al., 1985, 1986, [54, 52], et Laporte, 1992, [51] présentent des algorithmes exacts pour le VRP. Magnanti, 1981, [61] donne les principales formulations de base, et les algorithmes de résolution de problèmes de tournées de véhicules. Laporte et Nobert, 1987, [53] font le point sur les algorithmes exacts permettant la résolution du VRP.

Les algorithmes exacts pour résoudre les problèmes NP-difficiles nécessitent un nombre de calculs qui augmente exponentiellement avec la taille du problème. En effet, expérimentalement, les algorithmes exacts pour le VRP ne peuvent permettre de résoudre des problèmes de taille supérieure à 25 clients. En fait, aucune de ces méthodes n'a été mis en pratique, car le temps pour résoudre un VRP de taille réaliste deviendrait prohibitif. Nous nous intéresserons d'une manière prioritaire aux méthodes approchées.

2.2.2 Méthodes approchées

La plupart des travaux réalisés sur le VRP sont liés à des méthodes approchées, souvent appelées heuristiques. Elles se contentent d'obtenir des solutions aussi bonnes que possible en temps raisonnable mais ne garantissent pas leur optimalité. Les récents développements de la théorie de l'Optimisation Combinatoire ont grandement enrichi les heuristiques, ce qui s'est concrétement traduit par un grand nombre d'applications réussies (Fisher et Rinnooy Kan, 1988, [28]).

Les heuristiques qui ont été développées pour le VRP sont fondées sur des heuristiques initialement développées pour le TSP. En élargissant le schéma proposé par Christofides, les méthodes approchées peuvent être classées en quatre grandes familles:

- méthodes constructives,
- méthodes en deux phases,
- méthodes d'optimisation incomplète,
- méthodes d'amélioration.

Méthodes constructives

Les méthodes de ce type construisent progressivement des tournées de véhicules par insertion à chaque étape d'un client selon plusieurs critères de mesures de gain.

Un des méthodes les plus connues de cette famille est la méthode de Clarke & Wright, 1964, [18] dans sa version séquentielle et parallèle. Dans l'implantation parallèle, les tournées sont formées simultanément. Au début, chaque client est servi par un véhicule distinct. En satisfaisant les demandes des clients i et j par le même véhicule, on réduit le coût de la solution. En supposant que le dépôt soit le client d'indice 1, le coût de la visite des clients i et j indépendament par deux véhicules sera:

$$c_{1i} + c_{i1} + c_{1j} + c_{j1}$$

Tandis que le coût d'un véhicule pour visiter i et j à la suite dans la même tournée est:

$$c_{1i} + c_{ij} + c_{j1}$$

Donc de la liaison directe de i à j, résulte l'économie (saving) suivante:

$$s_1(i,j) = c_{i1} + c_{1j} - c_{ij}$$

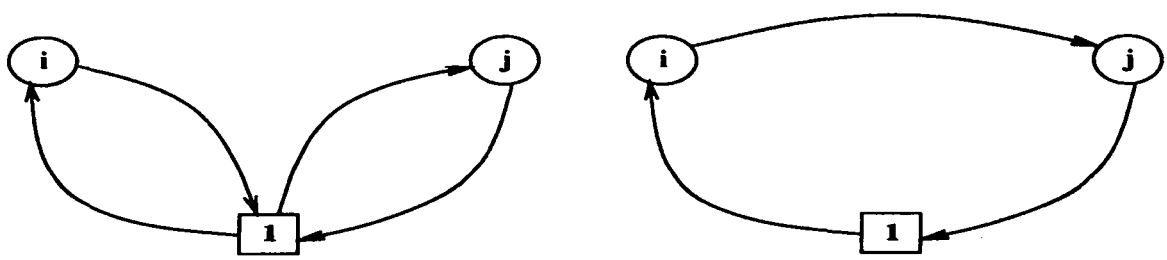


Figure 2.1 : Ajouter la liaison (i,j) et enlever $(i,1)$ et $(1,j)$

Ces "savings" sont rangés par ordre décroissant. L'algorithme rassemble sur une même tournée les clients i et j correspondant au saving $s_1(i,j)$ le plus fort et que ne violent pas les contraintes du problème jusqu'à ce qu'aucune liaison ne soit plus possible.

L'algorithme peut alors s'écrire de la façon suivante.

1. Construction d'une solution initiale

$$S = \left(\bigcup_{i=2}^{n}(1,i)\right) \cup \left(\bigcup_{j=2}^{n}(j,1)\right)$$

2. Calculer pour $i,j = 2,\cdots,n;\ i \neq j$ les *savings*:

$$s_1(i,j) = c_{i1} + c_{1j} - c_{ij}$$

3. Ranger les *savings* par ordre décroissant

4. Tant qu'il existe des "savings" $s_1(i,j) > 0$ non encore examinés faire

   si i et j appartient à des arcs $(i,1)$ et $(1,j)$ de différentes tournées,
   a) déterminer l'arc (i^*,j^*) non encore considéré tel que:
   $s_1(i^*,j^*) = \max\{s_1(i,j)\}$
   b) essayer de rassembler sur une même tournée les sommets i et j:
   $S' = S \cup \{(i,j)\} \setminus \{(i,1),(1,j)\}$.
   Si S' ne viole pas les contraintes du problème, poser $S \leftarrow S'$.

   fin Tant que

5. Enregistrer S solution du problème.

La méthode des savings a une complexité temporelle en $O(n^2 \log n)$ et une complexité spatiale en $O(n^2)$. Dans la version séquentielle au lieu de calculer toute la matrice des savings, un sommet quelconque est choisi pour un regroupement initial. En chaque itération de l'algorithme la liaison correspondante au plus fort saving, par rapport au dernier sommet ajouté, est sélectionnée. Une fois qu'une tournée (*cluster*) ne peut être ettendue, une nouvelle tournée est formée jusqu'à ce que tous les sommets soient servis.

La complexité spatiale peut être réduite à travers l'utilisation de structures de donnés appropriées (voir, par exemple Nelson et al., 1985, [67] et Paessens, 1988, [71]). De plus, plusieurs auteurs ont proposé des modifications pour l'algorithme de base.

Gaskell, 1967, [30] et Yello, 1970, [83] ont indépendament introduit le concept de *savings ponderés* $s_1(i,j) - \theta c_{ij}$, où θ est un paramètre positif.

En faisant varier θ, on peut accorder plus au moins d'importance à la distance entre les points i et j en fonction de leurs positions par rapport au dépôt central.

D'autres, tels que Mole et Jameson, 1976, [65], ont utilisé des méthodes constructives par généralisation du critère des savings et introduction de deux paramétres λ et μ. La méthode de Mole et Jameson différe de l'algorithme de Clark & Wright par le fait qu'un nouveau client n'est pas systématiquement rattaché à l'une des extrémités d'une route, mais il peut être inséré entre deux clients déjá désservis. Pour cela, les deux critères suivants sont utilisés:

$$\psi(i,l,j) = c_{il} + c_{lj} - \mu c_{ij}$$
$$\sigma(i,l,j) = \lambda c_{0l} - \psi(i,l,j)$$

Ainsi, pour chaque client l non encore affecté à un routage R, on calcule d'abord l'insertion admissible qui minimise le premier critère. Ensuite le meilleur client l^* à être vraiment inséré dans un routage est celui qui maximise le deuxième critère.

Concrétement l'algorithme est le suivant.

1. Initialiser $n-1$ tournées

$$R_t = (1, i_t, 1) \quad (t = 1, \cdots, n-1)$$
$$K = \{R_1, \cdots, R_{n-1}\}$$

2. Choisir la première tournée d'indexe $t = 1$

3. Tant qu'il existe encore des sommets non affectés à des tournées faire

   (a) Tant qu'il est possible d'insérer un sommet dans la tournée R_t faire

i. soit S_t l'ensemble des sommets non encore affectés à une tournée et où l'insertion dans une tournée t est admissible. Pour chaque sommet $k \in S_t$, considérer les arcs $(r,s) \in R_t$. Déterminer les sommets i_k et j_k correspondant à la meilleure insertion de k tels que:

$$\psi(i_k, k, j_k) = \min_{r,s \in R_t; k \in S_t} \{\psi(r,k,s)\}$$

ii. insérer le sommet k^* dans la tournée R_t tel que:

$$\sigma(i_{k^*}, k^*, j_{k^*}) = \min_{i_k, j_k \in R_t; k \in S_t} \{\sigma(i_k, k, j_k)\}$$

iii. optimiser la tournée R_t par une procédure k-opt (Lin et Kernighan, 1973,[59].

fin Tant que

(b) Poser

$$K = K \setminus \{R_t\}$$
$$t = t + 1$$

fin Tant que

Alors, à partir de la définition ci-dessus de $\psi(i,l,j)$ et $\sigma(i,l,j)$, il devient évident qu' en changeant les valeurs de λ et μ, il est possible d'avoir différents critères de choix pour le meilleur client à insérer.

Quand on augmente λ, on favorise la construction de tournées radiales. De même, quand on augmente μ, les longues liaisons sont pénalisées.

Selon l'idée proposée par Dror et Levy, 1986, [23] plusieurs auteurs tels que Altinkemer et Gavish, 1991, [1] ont développé des algorithmes sophistiqués en considèrant les *savings* associés à tous les échanges possibles entre deux tournées pour la résolution d'un problème de couplage pondéré (*Weight Matching Problem*), et en maximisant les économies obtenues en chaque itération.

Méthodes en deux-phases

Celles-ci peuvent être subdivisées en trois classes. La première s'insère dans le groupe des méthodes "Regroupement d'abord-Tournée ensuite" (*"Cluster first-route second"*), qui accordent la priorité au regroupement des sommets, sans spécifier l'ordre de la tournée. Dans la dernière phase, les tournées s'obtiennent pour chaque véhicule en utilisant un des algorithmes connus pour le problème du voyageur de commerce (TSP).

Gillet et Miller, 1974, [33] utilisent l'algorithme *"sweep"* pour la première phase, où la localisation des clients est représentée dans un système de coordonnées polaires

dont l'origine est le dépôt central. Les clients sont affectés à un véhicule tant que sa capacité n'est pas dépassée. Le processus est donc répété jusqu'à ce que tous les clients soient affectés.

On assume que les sommets sont représentés par leurs coordonnées polaires (θ_i, ρ_i), où θ_i est l'angle et ρ_i est la longueur du rayon. On donne la valeur $\theta_i^* = 0$ au sommet arbitraire i^* et on calcule les angles par réfèrence à $(1, i^*)$. Du point de vue de l'implémentation de l'algorithme, on considère que les sommets sont rangés par ordre croissant.

1. Choisir une première tournée R_t $(t = 1)$,
2. Tant qu'il est possible d'insérer un sommet dans R_t faire
   (a) Tant qu'il existe encore des sommets non affectés faire
      i. sélectionner un sommet i tel que :

$$\theta_i = \min_{k \in S_t} \{\theta_k\}$$

où S_t est l'ensemble des sommets non encore affectés à une tournée, et pour lequel l'insertion dans la tournée R_t est admissible.

   fin Tant que
   (b) Poser $t = t + 1$

   fin Tant que
3. Optimiser chaque tournée indépendament par une résolution exacte ou approchée du TSP correspondant.

L'algorithme en deux phases de Christofides, Mingozzi et Toth, 1979, [15] est plus sophistiqué puisqu'il considère plusieurs regroupements de tests gèrés par des paramètres de contrôle. Dans la phase I, l'algorithme construit k routes selon un processus d'insertion de coût minimal. Une fois qu'une tournée est choisie toutes les insertions se font sur cette tournée jusqu'à ce qu'aucun client ne soit plus possible d'être insérér. Dans la phase suivante, il commence avec k routes en attachant des clients à chaque tournée suivant le coût d'insertion en tournées alternatives. Dans ce cas, les insertions peuvent se faire parallèlement en différentes tournées. L'algorithme termine dès que tous les clients sont servis. Les deux phases sont répétées jusqu'à ce que tous les clients fassent partie d'une tournée.

Procédure générale

0. Tant qu'il existe des sommets non affectés à des tournées faire

   (a) Phase I

   (b) Phase II

   fin Tant que

Phase I - Procédure séquentielle

1. Choisir une première tournée d'indice $t = 1$

2. Tant qu'il existe encore des sommets non affectés à des tournées faire

   (a) sélectionner un sommet i_t pour initialiser la tournée R_t,
   pour chaque sommet i, calculer :

$$\delta_i = c_{1i} + \lambda c_{ii_t} \qquad (\lambda \geq 1)$$

   (b) Tant qu'il est possible d'insérer un sommet dans R_t faire

      i. insérer dans la tournée R_t le sommet i^* tel que :

$$\delta_{i^*} = \min_{i \in S_t} \{\lambda_i\},$$

      où S_t est l'ensemble des sommet non encore affectés à une tournée, dont l'insertion dans la tournée R_t est admissible,

      ii. optimiser la tournée R_t en utilisant des méthodes k-opt (Lin et Kernighan, 1973, [59]).

   fin Tant que

   (c) Poser $t = t + 1$

   fin Tant que

Phase II - Procédure parallèle

5. Initialiser k tournées

$$R_t = (1, i_t, 1) \qquad (t = 1, \cdots, k),$$

où k est le nombre de tournées obtenues à la fin de la première phase, et i_t est le même sommet choisi dans le pas 2.
Soit $K = \{R_1, \cdots, R_k\}$ l'ensemble de ces tournées.

6. Tant que $K \neq \emptyset$ faire

   (a) Tant qu'il existe encore des sommets non affectés à une tournée faire

      i. Pour chaque sommet i non encore affecté à une tournée, et pour chaque tournée $R_t \in K$, calculer :

$$\epsilon_{ti} = c_{1i} + \mu c_{ii_t} - c_{1i_t} \qquad (\mu \geq 1)$$

      affecter le sommet i à la tournée R_{t^*} tel que :

$$\epsilon_{t^*i} = \min_{R_t \in K} \{\epsilon_{ti}\}$$

   fin Tant que

   (b) Choisir une tournée $R_t \in K$.
   Poser :

$$K = K \setminus \{R_t\}$$

   Pour chaque sommet i associé à R_t, calculer :

$$\tau_i = \epsilon_{t'i} - \epsilon_{ti}$$

   où t' est donné par

$$\epsilon_{t'i} = \min_{R_t \in K} \{\epsilon_{ti}\}$$

   (c) Tant qu'il est possible d'insérer un sommet dans la tournée R_t faire

      i. Insérer dans la tournée R_t le sommet i^* tel que :

$$\tau_{t^*} = \max_{i \in S_t} \{\tau_i\}$$

      où S_t est l'ensemble des sommets non encore inséres et associés à R_t dont l'insertion dans cette tournée est admissible,

      ii. optimiser la tournée R_t par une procédure k-opt (Lin et Kernighan , 1973, [59]).

   fin Tant que

fin Tant que

Une amélioration de cette méthode a été proposée par Toth, 1984, [82].

Dans la deuxième classe, on trouve les méthodes "Tournée d'abord-Regroupement ensuite" (*"Route first-Cluster second"*), initialement proposées par Beasley,

1983, [2] et par Haimovich et Rinnooy Kan, 1985, [43]. Typiquement ces algorithmes commencent par construire une "bonne" tournée pour le TSP global, puis le divisent ensuite en des tournées admissibles pour le VRP.

Finalement, plusieures procédures itératives ont été proposées. Parmi ces méthodes, se trouve l'heuristique d'affectation généralisée de Fisher et Jaikumar, 1981, [27]. Elle est fondée sur l'idée d'une formulation mathématique du problème permettant son écriture sous la forme d'un problème d'affectation généralisée. La formulation est celle du (VRP3) qui a été présentée dans la section 2.2.1, sans les contraintes (2.37) et (2.38) de fenêtres de temps.

L'idée de base de cette approche peut être décrite par la formulation du (VRP3) (2.32)-(2.40) comme un problème d'affectation généralisée non linéaire (car la fonction Objectif n'est pas linéaire), de la façon suivante:

$$\text{(P1)} \quad \text{Minimiser} \quad \sum_{k=1}^{m} f(y_k) \tag{2.41}$$

sous

$$\sum_{i=1}^{n} d_i y_i^k \leq D_k \qquad k = 1, \cdots, m \tag{2.42}$$

$$\sum_{k=1}^{m} y_i^k = \begin{cases} m & i = 1 \\ 1 & i = 2, \cdots, n \end{cases}$$

$$y_i^k \in \{0,1\} \qquad i, \cdots, n; k = 1, \cdots, m \tag{2.43}$$

où $f(y_k)$ est le coût optimal du TSP de la tournée k comprenent l'ensemble de sommets $N(y_k) = \{i \mid y_i^k = 1\}$.

La fonction $f(y_k)$ peut se définir mathématiquement par:

$$\text{(P2)} \quad \text{Minimiser} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \sum_{k=1}^{m} x_{ij}^k \tag{2.44}$$

sous

$$\sum_{i=1}^{n} x_{ij}^k = y_i^k \qquad j = 1, \cdots, n; k = 1, \cdots, m \tag{2.45}$$

$$\sum_{j=1}^{n} x_{ij}^k = y_i^k \qquad i = 1, \cdots, n; k = 1, \cdots, m \tag{2.46}$$

$$\sum_{i,j \in S} x_{ij}^k \leq \mid S \mid -1 \qquad S \subset V; \mid S \mid \geq 2; k = 1 \cdots, m \tag{2.47}$$

$$x_{ij}^k \in \{0,1\} \qquad i, j = 1, \cdots, n; k = 1, \cdots, m \tag{2.48}$$

Comme la fonction $f(y_k)$ est difficile à calculer exactement, l'équation (2.41) est remplacée par

$$\sum_{k=1}^{m} \sum_{i=1}^{n} d_i^k y_i^k$$

où $\sum_{i=1}^{n} d_i^k y_i^k$ est une approximation linéaire de $f(y_k)$ et le coeficient d_i^k réprésente le coût d'insertion du sommet v_i dans la tournée du véhicule k.

Il existe plusieurs méthodes pour construire une approximation linéaire de $f(y_k)$. Les auteurs ont choisi pour d_i^k le critère suivant:

$$d_i^k = \min[c_{1i} + c_{ii_k} + c_{i_k 1}, c_{1i_k} + c_{i_k i} + c_{i1}] - [c_{1i_k} + c_{i_k 1}]$$

L'heuristique consiste à déterminer d'une part l'affectation optimale des clients aux véhicules, et d'autre part de la tournée de chaque véhicule.

Concrétement, l'algorithme est le suivant.

Phase I

1. Choisir k ($k << n$) sommets pour être les "semences" des regroupements (*clusters*) et affecter un véhicule à chacun.

2. Pour chaque sommet i et pour chaque regroupement k, calculer le coût d'insertion d_i^k relativement à la semence du regroupement.

3. Résoudre le problème d'affectation généralisé (P1).

Phase II

4. Résoudre le TSP pour chaque ensemble de sommets $N(y_k)$.

Méthodes d'optimisation incomplète

Les algorithmes d'optimisation incomplète consistent à obtenir une "bonne" solution à partir de recherches arborescentes incomplètes. Il s'agit essentiellement d'algorithmes du type *"Branch & Bound"* dirigés par des méthodes heuristiques de façon à hâter leur fin. Pour cela, un algorithme énumératif quelconque peut être utilisé (voir Laporte et Nobert, 1987, [53]). Christofides, Mingozzi et Toth, 1979, [15] ont conçu une procédure de recherche arborescente fondée sur l'algorithme initial de Christofides (1976).

Méthodes d'amélioration

Ces méthodes ont été initialement mise en œuvre par Croes (1958) et Lin, 1965, [58] pour le TSP. Elles sont fondées sur le concept de k-échanges. Lin et Kernighan, 1973, [59] ont généralisé cette approche, très utilisée par la suite dans d'autres problèmes. Dans le contexte de tournées de véhicules, Christofides et Eilon, 1969, [13] et Russell, 1977, [75] ont adapté cette approche pour le classique VRP, et Psaraftis, 1983, [73] l'a utilisé pour le problème connu sous le nom de *"dial-a-ride"*, transport à la demande.

Les autres, Meta-heuristiques telles que le Recuit Simulé (*Simulated Annealing*) et la Recherche Tabou (*Tabu Search*) peuvent être vues comme des méthodes d'amélioration. Il s'agit de méthodes itératives dans lesquelles on part d'une solution initiale et on examine en chaque étape le voisinage de la solution courante. Une meilleure solution y est recherchée de façon à atteindre un minimum local.

Dans le chapitre 5 seront montrés des résultats expérimentaux permettant comparer les performances entre ces heuristiques pour différents problèmes de tests.

Chapitre 3

Heuristiques récentes d'optimisation locale

Nous effectuons dans ce chapitre, une présentation succinte des récents heuristiques d'optimisation locale tells que le Recuit Simulé et les Algorithmes Genétiques. Bien que la Méthode Tabou appartienne à cette famille d'heuristiques, mais nous ne la présenterons que dans le chapitre suivant, de façon beaucoup plus detaillée.

3.1 Définitions et notations

On présente ici une description du problème d'après de Werra.

Le problème d'optimisation combinatoire à résoudre peut se définir comme suit.

Soient X l'ensemble des solutions admissibles d'un problème initial P, et f une fonction de coût à valeurs réelles définie sur X. i

On veut trouver une solution s^* de P qui miminise f.

- Une définition du problème (P) est alors:
  $min\{f(s) \mid s \in X \subset R^n\}$
  X étant fini

  On définit:

- m, un mouvement permettant de passer d'une solution s à une autre s'

  $s :\longrightarrow s'$
  $s' = s \oplus m$

- M, l'ensemble des modifications acceptables et par conséquent,
  $M(s)$ l'ensemble des modifications applicables à la solution s,

$M(s) = \{m \in M \mid s \oplus m \in X\}$

- $N(s)$, l'ensemble des solutions atteignables à partir de s par un mouvement m, (le voisinage de s),

  $N(s) = \{s' \mid \exists m \in M(s) \text{ avec } s' = s \oplus m\}$

- l'ensemble X et la definition du voisinage permettent de définir un graphe d'espace d'états $G = (V, A)$

  $\forall s, s' \in V, \exists (s, s') \in A \text{ t.q. } s' \in N(s)$

- un problème d'optimisation local (P1),

  $min\{f(x) \mid s \in V^* \subseteq N(s)\}$

Comme point de départ, nous comparerons les méthodes qui suivent avec une méthode générale de descente comme suit:

1. choisir une solution initiale $s \in X$
   $stop := faux$;
2. tant que $stop = faux$ faire
   - engendrer un échantillon $V^* \subseteq N(s)$;
   - rechercher $s' \in V^*$ telle que $f(s') = min_{x \in V^*} f(x)$;
   - si $f(s') \geq f(s)$
     alors $stop := vrai$;
     sinon $s := s'$;

Cette méthode conduit à un optimum local, dont la valeur peut être fort éloignée de l'optimum global.

Sont apparues récemment de nouvelles heuristiques s'inspirant de la méthode décrite ci-dessus, mais guidant la recherche pour qu'elle ne s'arrête pas aux optima locaux. Pour cela, ces méthodes doivent parfois accepter de détérieorer la valeur de la solution courante afin de sortir d'un optimum local. Ce faisant, on risque de cycler et donc de revenir à l'optimum local visité précédemment. Pour éviter ce cyclage, chaque méthode propose une solution.

Parmi ces méthodes, se trouvent le Recuit Simulé et Tabou.

3.2 Le recuit simulé

Le Recuit Simulé est une méthode issue de la Mécanique Statistique.

L' application à l'Optimisation Combinatoire a été proposé par Kirkpatrick et al., 1983, [49].

En Métallurgie, l'obtention d'un cristal parfait se fait grâce à la méthode du recuit. On porte un métal à une température suffisament elevée pour qu'il soit dans l'état liquide. Puis, à partir de cet état, on abaisse la température, et de ce fait les atomes se réorganisent en une autre nouvelle structure.

Une même structure initiale peut donner différentes structures finales selon la façon dont on baisse la température. Si celle-ci baisse trop brutalement, on risque d'atteindre un état métastable qui ne correspond pas à l'état fondamental, minimum absolu d'énergie interne, mais un minimum local de l'énergie. On a obtenu un verre.

Il est donc essentiel que l'abaissement de la température se fasse très lentement et très régulièrement. Pour faire disparaitre d'éventuels défauts dans la structure du cristal, on utilise la technique du recuit. On réchauffe un peu le métal, afin que les atomes aient plus de liberté de mouvement. En chauffant suffisamment, on donne assez d'énergie pour qu'ils sortent de l'optimum local. En abaissant à nouveau la température régulièrement, ils pourront atteindre l'optimum global.

C'est en s'inspirant de ce procédé que le recuit a été développé. Il s'agit d'un algorithme stochastique, permettant une détérioration de la valeur courante selon une certaine probabilité. Cette probabilité dépend de l'importance de la détérioration et du degré d'avancement de la recherche.

Cette méthode n'exclut donc pas entièrement le cyclage, mais l'autorise selon une certaine probabilité.

Concrétement, l'algorithme est le suivant:

1. Choisir une solution initiale s;
2. Choisir une température initiale T;
3. Tant que *le système n'est pas gelé* faire
   (le système est gelé lorsque la température est devenue très basse et que par suite la solution ne peut pas plus évoluer)
   (a) Tant que *l'équilibrage à T n'est pas atteint* faire
      - Choisir aléatoirement un mouvement élémentaire
        (un 2-échange par exemple)
        $s' := s \oplus m$;
      - Calculer la variation de coût
        $\Delta f := f(s') - f(s)$;

- Si $\Delta f \leq 0$ (amélioration de la solution)
  alors $s := s'$; (modification acceptée)
  sinon $s := s'$ avec la probabilité $exp^{-\frac{\Delta f}{T}}$;

(b) Réduire la température;

L'obtention d'un cristal sans défaut revient donc à minimiser une grandeur physique (l'énergie) en évitant les minima locaux, alors que le nombre d'états possibles est immense (le nombre de molécules étant lui-même très élevé, de l'ordre de $10^{23}/cm^3$).

En Optimisation Combinatoire, on est confronté à un problème du même type: minimiser une fonction sur un grand nombre de combinaisons possibles (en général suffisamment grand pour empêcher l'examen de tous les possibilités en un temps raisonnable). Et là aussi, on essaie d'éviter les minima locaux. C'est sur cette idée que l'analogie entre la Mécanique Statistique et les problèmes d'Optimisation Combinatoire a été mise à profit. La solution optimale joue le rôle de l'état cristallin, et les minima locaux remplaceront les états métastables. Une différence fondamentale apparaît dès alors: il n'y a plus ici la notion de température, elle deviendra simplement un paramètre de contrôle lorsqu'on simule un recuit (d'où le nom "recuit simulé", ou "simulated annealing" en anglais).

Cette méthode dispose d'une convergence théorique prouvée sur certaines conditions, mais l'effort de calcul nécessaire peut être très important.

Depuis, les performances du recuit ont fait des adeptes.

Parmi les disciplines où il a donné déjà de bons résultats, on peut notamment citer:

- l'Optimisation Combinatoire (voyageur de commerce, couplage de longueur minimum, tournées de véhicules...),
- la CAO (conception de circuits VLSI, placement de composants,...),
- le traitement d'images (restituition d'images brouillées,...),
- la reconnaissance des formes (codes postaux...).

On trouvera dans le chapitre 5, les résultats données pour une de ces applications, par Osman, 1993, [69] au VRP.

3.3 Les algorithmes génétiques

Les algorithmes génétiques ont été crées pour imiter des processus d'évolution naturelle. Ces méthodes se fondent sur les procédés de reproduction et de sélection génétiques.

En partant d'un ensemble de solutions faisables (*population*) de cardinalité n *(popsize)*, on affecte à chaque solution une certaine probabilité d'être sélectionée comme père, cette probabilité dépendant du coût de la solution.

On procède ensuite au croisement entre ces solutions pour obtenir un ou plusieurs enfants par paire de solutions. Les pères choisis pour être croisés le sont en fonction des probabilités calculées précédemment. On sélectionne alors les n enfants en fonction du coût de chaque solution, et on recommence.

A la manière d'un phénomène naturel, la méthode repose sur l'idée qu'au fur et à mesure que les générations se renouvellent, la valeur moyenne des solutions faisables s'améliore.

Un algorithne génétique a donc la forme suivante:

1. Créer la population initiale: *popsize* solutions;
2. Tant que *le critère d'arrêt n'est pas vérifié* faire
   (a) **Evaluation**
      - Evaluer chaque individu i de la population;

   (b) **Sélection**
      - Calculer, d'après l'ensemble de ces évaluations, la justesse proportionnelle *(fitness)* $f(i)$ de chacune de ces solutions et ainsi la probabilité $p(i)$ de chacune d'entre elles d'être choisie comme "reproductice";
      - Constituer un ensemble de couples d'individus compte tenu de cette probabilité;

   (c) **Reproduction**
      - Application des opérateurs génétiques pour sélectioner des couples;
      - Combiner chaque couple afin de produire une (ou plusieurs) nouvelle solution issue des deux parents;
      - Si le nombre des solutions produits est supérieur à *popsize* alors garder les *popsize* meilleurs "enfants"; (une nouvelle génération de taille *popsize* est donc constitué)

A cet algorithme de base peut être ajouté une "régénération" aléatoire de la population. Celle-ci peut être obtenue par un opérateur génétique appelé mutation : selon une certaine féquence, paramètre du programme, créer des enfants dont une partie des gènes sont aléatoires (mutants) et donc indépendants de ceux des deux parents, le reste de la solution étant construit classiquement par combinaison des gènes des parents.

Plus encore que le recuit simulé et la méthode Tabou, une marge de man-œuvre considèrable est laissée à l'utilisateur dévéloppant un algorithme génétique en vue de résoudre un problème particulier.

En fait, même si les similitudes de cette méthode avec le modèle biologique la rendent très séduisante et attractive, sa mise en œuvre est extrèmement difficile et nécessite une longue phase d'expérimentations.

Les résultats obtenus semblent, de surcroit, se montrer en deça de ceux de la méthode tabou et même du recuit simulé pour résoudre des problèmes d'Optimisation Combinatoire. C'est le cas pour la résolution du problème du voyageur de commerce, Grefenstette et al., 1989, [25] et Jog et al., 1989, [70]. C'est également le cas par exemple pour la résolution du problème d'affectation quadratique.

Certains de ces résultats sont toutefois tout à fait honorables et permettent de classer l'algorithme génétique parmi les méthodes efficaces pour la résolution de ces problèmes.

Remarquer qu'aucune application de ces algorithmes au VRP n'est encore connue.

Chapitre 4

Présentation de la méthode Tabou

Un grand nombre de problèmes d'optimisation trouvés dans la pratique (télécomunications, logistique, planning financier, transports et production) a motivé le développement des techniques d'optimisation.

Les chercheurs ont adapté des idées de plusieurs domaines dans le but de développer des procédures d'optimisation plus puissantes. Les méthodes de réseaux de flux sont par exemple fondées sur des idées appartenant aux modèles d'électricité et hydraulique. Le Recuit Simulé est fondé sur des processus physiques en métallurgie, tandis que les méthodes génétiques essayent d'imiter les phénomènes biologiques d'évolution naturelle. De même, la méthode Tabou peut s'apparenter à une technique fondée sur des concepts d'Intelligence Artificielle.

Tabou est actuellement une heuristique qui donne de très bons resultats pour bon nombre de problèmes d'optimisation. Les succès de la méthode Tabou pour une variété de problèmes classiques et pratiques d'optimisation sont reportés dans un tour d'horizon fait par Glover, 1989, [35]. En particulier, le succès de cette approche pour le problème classique du voyageur de commerce (TSP) (Malek et al.,[13]) a motivé l'application de la méthode Tabou à d'autres classiques et à des problèmes plus généaux. Ainsi, le coloriage de graphes (Hertz et de Werra, 1987, [47]), l'affectation quadratique (QAP) (Skorin-Kapov, 1990, [78], Taillard, 1991, [80] et Chakrapani et Skorin-Kapov, 1991, [11]) et enfin les tournées de véhicules (VRP) (Pureza et França, 1991, [74]; Osman, 1993 [69]; Taillard, 1992 [81]; Gendreau, Hertz et Laporte, 1990, [31] et Semmet et Taillard, 1992, [77]).

Initialement élaborée par Glover, 1986, [34] et indépendament par Hansen, [44] pendant le même période, la méthode est directement inspirée des procédures d'améliorations itératives.

4.1 Formulation générale

Une fonction *Objectif* f doit être minimisée dans un ensemble X de solutions admissibles. Un voisinage $N(s)$ est défini pour chaque solution s de X.

L'ensemble X et la définition du voisinage se traduisent par un graphe d'espace d'états G (qui peut être infini). La méthode Tabou est une procédure essentiellement itérative qui en partant d'une solution initiale tente d'atteindre la solution optimale, en exécutant en chaque pas des mouvements dans un graphe d'espace d'états G. Chaque pas consiste d'abord à engendrer une collection V^* de solutions dans le voisinage $N(s)$ de la solution courante s. Puis passant pour la meilleure solution $s' \in V^*$, même si cela entraîne une augmentation de la fonction *Objectif* que l'on veut minimiser.

Les solutions consécutivement visités en chaque étape de la recherche créent un chemin orienté dans G. Normalement trouver la meilleure solution dans V^* n'est pas toujours une tâche triviale. On doit au besoin résoudre un problème d'optimisation "locale" $min\{f(x) \mid s \in V^* \subseteq N(s)\}$. Selon la taille du voisinage choisie, le problème d'optimisation locale peut se résoudre exactement ou de façon approchée.

En acceptant de détériorer la valeur de la solution courante, on permet de "s'éloigner" d'un optimum local, mais cela peut induire des phénomènes de cyclage, (parcours répétitif du même ensemble de solutions). Pour éviter ce cyclage la méthode consiste en l'introduction de restrictions Tabou servant à interdire les mouvements précédents pendant un certain nombre d'itérations. Ces mouvements sont induits dans une liste cyclique T constamment tenue à jour: la *liste Tabou.*

A chaque itération l'élement le plus ancien est remplacé par le dernier mouvement (ou le mouvement inverse).

L'algorithme dans sa version la plus simple est donc le suivant:

1. Choisir une solution initiale $s \in X$
   $s^* := s$;
   $T := \emptyset$

2. Tant que le *critère d'arrêt n'est pas verifié* faire

   - Engendrer un échantillon $V^* \subseteq N(s) - T$;
   - Rechercher $s' \in V^*$ telle que $f(s') = min_{x \in V^*} f(x)$;
   - $s := s'$;
   - Si $f(s') \geq f(s^*)$
     alors $s^* := s'$;
   - Mettre à jour la liste T

4.2 Les bases de la méthode

La méthode Tabou est une méta-heuristique qui guide la recherche dans l'espace très complexe des solutions.

La philosophie de la méthode Tabou est d'exploiter une collection de principes de façon à résoudre un problème intelligement. L'élément fondamental de la méthode est l'utilisation de structures de mémoire flexibles au lieu de structures de mémoire rígides (comme dans la recherche *Branch & Bound* et A^*) ou encore de systèmes sans mémoire (comme le *Recuit Simulé* ou autres approches aléatoires).

Les structures de mémoire de la recherche Tabou opérent par référence à quatre types de mesures: *ancienneté, fréquence, qualité* et *influence*. Ces mesures sont évaluées à partir d'un ensemble de structures logiques construites au cours du processus de la recherche.

La méthode Tabou est fondée sur trois structures de mémoire flexible, celles-ci correspondent à trois stratégies de recherche.

4.2.1 Mémoire à court terme

La mémoire à court terme correspond à une *stratégie d'exploration "agressive"*, recherchant le meilleur mouvement possible (de plus forte évaluation) sujet à certaines contraintes. Ces contraintes enveloppent les restrictions Tabou, qui sont concus pour prévenir des mouvements inverses ou parfois la répétition de certains mouvements. L'objectif principal des restrictions Tabou est de permettre à la méthode de sortir des points autour de l'optimum local tandis qu'il fait des mouvements de qualité élevée en chaque étape, amenant le processus dans d'autres directions.

Le choix du "meilleur mouvement" admissible (qui peut être ou non une amélioration de la solution courante) est fondé sur l'hypothèse que les mouvements de plus forte évaluation permettront d'atteindre une solution optimale ou proche de celle-ci.

Le niveau de la recherche Tabou et la mémoire introduite dans le processus d'optimisation, dépendent principalement du choix des mouvements considérés, et donc de la complexité du sous-problème à résoudre en chaque étape de la procédure. Si le voisinage considéré est large, la recherche est plus générale et de plus *haut-niveau*. En particulier, quand les mouvements disponibles sont nombreux, il devient approprié de considérer une approche sous forme de *liste de candidats* pour réduire la complexité du calcul afin d'évaluer l'ensemble des alternatives en chaque itération.

Pour des problèmes où les mouvements disponibles ne sont pas nombreux ou difficiles à évaluer (ou lorsque le calcul parallèle réduit effectivement la tâche d'évaluation), la liste des candidats consiste en tous ces mouvements.

4.2.2 Mémoire à moyen terme

Quand une région (dans le sens définit par un voisinage de *bas-niveau*, induit par simples modifications) de X semble contenir de bonnes solutions, une procédure intelligente est d'intensifier la recherche dans cette région. Le but de la mémoire à moyen terme est précisement de procéder à cette *stratégie d'intensification.*

Ainsi, la mémorisation des meilleures solutions peut permettre par exemple, de dégager quelques proprietés communes à ces solutions et de définir, de cette manière, la région des solutions possédant ces proprietés. Par exemple dans le VRP, les arcs insérés en différentes bonnes routes ne sont qu'une petite fraction de l'ensemble des arcs possibles. Alors pour approfondir la recherche autour d'une bonne solution, on peut considèrer l'ensemble (très limité) des solutions antérieures. Comme le voisinage est alors trop petit, les itérations sont beaucoup plus rapides et la procédure de recherche peut effectuer une évaluation plus complète dans un intervalle donné de temps.

Généralement pour procéder à une intensification, on se place en une solution qui paraît satisfaisante et on lance la recherche en se limitant à cette région. On peut par exemple exhiber une propriété des solutions appartenant à cette région, et rendre Tabou tout le mouvement modifiant cette proprieté. On peut également ajouter une pénalité dans la fonction *Objectif* pour les solutions appartenant à d'autres régions.

4.2.3 Mémoire à long terme

Une technique de recherche intelligente doit éviter que certaines régions ne soient totalement négligées. La fonction de mémoire à long terme de la méthode Tabou est de favoriser une *stratégie de diversification* de la recherche conduisant le processus à explorer des différentes régions. Pour cela on peut relancer la recherche à partir des différentes solutions initiales. Ces solutions initiales peuvent être génerées aléatoirement, mais on ne peut parler en ce cas de mémoire à long terme ni de recherche intelligente car il n'y pas d'apprentissage du passé. Il est plus intéressant de mémoriser des informations au cours de la première recherche, puis de créer une fonction pour générer des nouveaux points de départ pour le processus de diversification.

4.3 Le rôle du statut Tabou

Les restrictions Tabou agissent pour identification d'attributs de mouvements. Par exemple, dans le problème de tournées de véhicules (VRP) un type standard de mouvement consiste à transférer un élément d'un routage dans un autre. Une varieté d'attributs peuvent être associés à ce mouvement: l'identification de l'élément transféré, la demande de cet élément, les routages impliqués dans le transfert, le nombre d'éléments de chaque routage, ...

Pour interdire des instances ou combinaisons de ces attributs dans les mouvements futurs, le mouvement courant peut empêcher un retour inverse (ou repété, selon le choix).

Par exemple, interdire tous les mouvements qui transfèrent un certain élément de son dernier routage, ou les mouvements qui donnent une valeur de la fonction *Objectif* égale à la valeur avant transfert.

Typiquement, les restrictions Tabou ne sont activées que dans deux cas: ou leurs attributs se produisent dans un nombre limité d'itérations avant l'itération courante (créant la restriction basée sur l'*ancienneté*) ou bien, ils se produisent avec une fréquence supérieure à un nombre suffisament grand d'itérations (créant la restriction fondée sur la *fréquence*). Plus précisement, la restriction Tabou est activée seulement quand les attributs affectés satisfont certains seuils d'*ancienneté* ou de *fréquence*.

Pour exploiter cette notion, on définit un attribut *tabou-actif* pour interdire les mouvements inverses (ou repétés) pendant un intervalle de temps prédéfini par l'*ancienneté* ou la *fréquence*.

Un attribut qui n'est pas *tabou-actif* est appelé *tabou-inactif*. A la condition, pour un attribut de devenir *tabou-actif* ou *tabou-inactif*, on associe ce que l'on appele le *statut-tabou*.

Parfois, un attribut peut être appelé *tabou* ou *non tabou* pour indiquer s'il est *tabou-actif* ou *tabou-inactif*.

Il est aussi important de distinguer "l'attribut tabou" et le "mouvement tabou". Ainsi, un mouvement peut contenir des attributs *tabou-actifs* et être non tabou si ces attributs ne sont pas en nombre suffisant pour activer la restriction tabou.

Les restrictions tabou les plus utilisés consistent à rendre tabou les mouvements inverses, où l'objectif est d'éviter les phénomènes de cyclage et donc de donner plus d'efficacité au processus de recherche.

D'un point de vue pratique, les restrictions Tabou sont créees pour sélectionner un ensemble d'attributs et les enregistrer dans une liste Tabou T. Le type d'implantation usuelle et le plus simple consiste à obliger tous ces attributs à rester dans la liste pendant un nombre t d'itérations, où t est la longueur de la liste. Comme chaque nouvel attribut (ou combinaison d'attributs) est ajouté à la liste à chaque itération, l'attribut le plus ancien sort de la liste après $t + 1$ itérations. Ce type de liste Tabou peut être implantée efficacement sous forme de liste circulaire.

Pour réprésenter ce processus, appelons:

$e(i)$ un attribut ("élément") qui est mis Tabou à l'itération i;

q l'index de l'itération courante.

La liste Tabou T est alors définie de la façon suivante:

$$T = (e_{(p)}, \ldots, e_{(q-1)}, e_{(q)})$$

avec $p = t - q + 1$.

Cette liste est accompagnée d'une variable binaire qui réprésente le *statut-tabou* de l'attribut e:

$$statut\_tabou(e) = \begin{cases} 1 & \text{si } e \text{ est tabou,} \\ 0 & \text{sinon.} \end{cases}$$

4.4 La taille de la liste Tabou

La meilleur valeur de t reste à definir. La taille de la liste est à déterminer empiriquement, elle varie avec les problèmes. Il y est clair que choisir une taille optimale est essentiel: une liste trop petite risque de conduire à un cyclage, alors qu'une liste trop grande peut interdire des mouvements intéressants qui nous auraient conduit vers de nouvelles solutions. Souvent cette taille optimale augmente proportionnellement avec la taille du voisinage.

Les règles pour déterminer t sont statiques ou dynamiques (Glover et Laguna, 1992, [38]). Les *régles statiques* choisissent une valeur fixe de t tout au long de la recherche. Les *régles dynamiques* font varier la valeur de t au fur et à mesure. Les expériences pratiques indiquent que généralement les règles dynamiques sont plus robustes que les règles statiques (Glover, Taillard et de Werra, 1992, [39]).

Les régles dynamiques peuvent encore se diviser en deux catégories:

- *régles dynamiques simples*, où l'on choisit t aléatoirement entre deux bornes t_{min} et t_{max};
- *régles dynamiques dépendantes* de l' attribut où l'on choisit t comme pour la régle dynamique simple mais où l'on choisit des plus grands de t_{min} et t_{max} pour les attributs plus attractifs.

Par exemple, on peut choisir t_{min} et t_{max} sur des considérations de *qualité* ou *d'influence*, (Taillard, 1992, [81]).

De plus, une classe de régles dynamiques fondées sur l'introduction de *"moving gaps"* a montré récemment de bons résultats (Chakrapani et Skorin-Kapov, 1991, 1992, [11, 10]).

4.5 Le critère d'aspiration

La restriction de mouvements induite par la liste Tabou peut parfois s'avérer par trop rigide car elle peut interdire un déplacement vers une nouvelle solution très intéressante. Les *critères d'aspiration* autorisent alors la levée du statut Tabou du mouvement correspondant. Il doit être ainsi possible d'annuler le statut Tabou d'un mouvement s'il est vraiment désirable.

L'idée de ce critère est fondé sur le concept *d'influence* (Glover Taillard et de Werra, 1992, [39]), qui mesure le degré de changement induit dans la structure de la solution ou admissibilité.

Un mouvement de *haute-influence* peut ou non être une amélioration de la solution courante, car il est plus difficile de produire une amélioration quand la solution courante est relativement bonne. Théoriquement, si s est la solution courante, la modification m appliquée à s sera malgré tout acceptée si elle posséde un niveau d'aspiration $a(s,m)$ situé en dessous d'une valeur seuil $A(s,m)$. Plus précisement on attribue à $A(s,m)$ un ensemble de valeurs désirées pour la fonction $a(s,m)$.

Ainsi, les conditions d'aspiration peuvent se réprésenter sous de la forme suivante:

$$a_i(s,m) \in A_i(s,m) \quad (i = 1, \cdots, I) \tag{4.1}$$

Le statut d'un mouvement m est levé si au moins une (ou un certain nombre) des conditions 4.1 sont satisfaites.

On peut par exemple définir

$$\begin{aligned} a(s,m) &= f(s \oplus m), \\ A(s,m) &= f(s^*), \end{aligned}$$

où s^* est la meilleure solution trouvée jusqu'à présent.

Le mouvement m sera accepté s'il améliore la valeur de la meilleure solution s^*. La fonction A est appelée *fonction d'aspiration.*

4.6 La recherche d'un voisinage

Pour utiliser avec succès la méthode Tabou dans une application quelconque, il est fondamental de bien caractériser la recherche d'un voisinage.

Dans la recherche d'un voisinage, chaque solution s de X est associée à un ensemble de voisins, $N(s)$ inclus dans X, qu'on appélle le voisinage de s.

Chaque solution s' de $N(s)$ peut être atteinte directement de s par une opération m appelée *mouvement.*

D'habitude, dans la méthode Tabou, les voisinages sont considérés comme symétriques, c'est-à-dire s' est un voisin de s si et seulement si s est un voisin de s'.

D'un point de vue pratique, la recherche d'un voisinage se traduit par un critère de sélection d'un sous-ensemble de variables pouvant changer de valeur. En particulier, dans le cas des problèmes de programmation en variables entières, l'affectation de valeurs 0 ou 1 à une variable peut se rapporter à une variété d'opérations telles que, changer la valeur de l'éxécution d'une tâche sur une machine, ajouter ou enlever des arcs dans un graphe, exécuter un pas de *pivot*, ...

Par exemple, $x_j = 0$ ou 1 peut répresenter le fait que la variable associée est *non-basique* ou *basique* dans un point extréme de l'espace des solutions, comme dans la méthode du simplexe et ses variantes en programmation mathématique.

4.6.1 Les procédures de liste de candidats

Pour des problèmes où le voisinage à examiner est trop grand, et où les éléments de ce voisinage sont difficiles à évaluer, il devient important d'isoler un sous-ensemble de mouvements candidats ayant les caractéristiques désirables, et de seulement les évaluer au lieu du voisinage complet. Ce genre de procédures a été utilisé dans plusieurs méthodes d'optimisation pour réduire la complexité du calcul. La plupart de ces stratégies sont issues des procédures d'optimisation de réseaux (Glover et al., 1974, [37]; Mulvey, 1978, [66]). Dans ces approches, le sous-ensemble de mouvements candidats est représenté par une liste d'éléments (tels que l'indices des variables, des sommets ou des arcs). Une manière simple de construire cette liste est d'engendrer un échantillon d'éléments dans l'espace du voisinage en répétant ce processus si le résultat n'est pas satisfaisant.

Des études d'optimisation de réseaux ont suggéré des approches fondées sur des schémins de mouvements plus systématiques qui ont produit de meilleurs résultats.

Généralement, elles supposent la décomposition du voisinage en sous-en-sembles critiques et en utilisant des régles pour s'assurer qu'un sous-ensemble non examiné à une itération sera obligatoirement examiné à une itération suivante. Ces techniques sont motivées par le fait qu'un mouvement de plus forte évaluation qui n'est pas exécuté dans l'itération courante peut être un bon mouvement aux itérations suivantes.

Une technique pour ne chercher qu'un sous-ensemble du voisinage, consiste alors à garder dans une liste de candidats une collection des mouvements susceptibles d'être les meilleurs (de plus forte qualité). Ainsi, à chaque itération les mouvements appartenant à la liste de candidats sont d'abord examinés. De temps en temps, après un certain nombre d'itérations ou lorsque la qualité de ces mouvements se détériore et passe en dessous d'une valeur seuil prédéterminée, on examine une large portion du voisinage pour reconstruire une nouvelle liste de candidats.

Une autre technique pour construire une liste de candidats consiste à garder quelques attributs de mouvements correspondants à des bonnes solutions et à choisir en chaque itération des mouvements ayant ces attributs. Cette approche a été utilisée par Gendreau, Hertz et Laporte, 1991, [31] dans le contexte du VRP.

La philosophie de "liste de candidats" définit implicitement une stratégie de diversification provoquée par l'exploitation de différentes régions du voisinage à différentes itérations. Ce qui suggère l'inclusion de ces stratégies dans celles de diversification. On peut par exemple considérer le concept de "fréquence" dans la séléction des éléments de la liste de candidats.

4.6.2 Structures de voisinage

L'identification d'un voisinage pour définir des mouvements permettant de passer d'une solution à une autre peut être extrémement important. Par exemple, dans la résolution d'un problème de programmation mathématique, les méthodes qui utilisent des mouvements pour augmenter au diminuer la valeur des variables du problème, produisent certainement des solutions finales différentes de celles qui choisissent des mouvements par des processus de pivot ou de recherche directionnelle.

Les innovations qui ont été faites en Programmation Mathématique essaient de découvrir des voisinages efficaces pour réaliser ces mouvements.

Pour des problèmes d'Optimisation Combinatoire, où la construction de voisinages se fait normalement par des processus constructifs, destructifs ou d'échanges, une amélioration résulte de la combinaison de voisinages pour créer des mouvements plus puissants. Parfois, il est prèférable de considèrer en chaque pas de la recherche, la combinaison des mouvements d'insertion et d'échange. Une autre manière de combiner des voisinages, consiste à générer des mouvements composés dont la séquence de mouvements simples est traitée comme un seul mouvement plus complexe.

Un type spécial d'approche de ces mouvements pour créer des mouvements composés consiste en une succession de pas où chaque élément est affecté à un nouvel état provoquant l'éjection d'un autre élément de son état courant.

L'élément éjecté est alors affecté à un nouvel état, éjectant ainsi un autre élément et créant une chaîne d'opérations. Ce type d'approche est appelée *stratégie de chaîne d'éjection.*

Une des caractéristiques fondamentales de cette approche se traduit par la dépendance entre les mouvements de la chaîne d'éjection dont l'effet cumulatif des pas antérieurs, doit influencer le pas immédiatement suivant.

L'utilisation de stratégies de chaîne d'éjection dans la méthode Tabou a été efficace pour le problème d'affectation généralisée (Laguna et al., 1991, [50]), d'où l'idée de l'étendre à d'autres.

Chapitre 5

Tabou appliqué au problème de tournées de véhicules

Le propos de ce chapitre est de présenter une nouvelle procédure pour la version du VRP décrite dans le paragraphe suivant.

5.1 Définition du problème et notation

Soit $G = (V, A)$ un graphe dont $V = \{v_1, v_2, \cdots, v_n\}$ réprésente l'ensemble des sommets et $A = \{(v_i, v_j) \mid v_i, v_j \in V; i \neq j\}$ l'ensemble des arcs. Le dépôt est localisé en v_1. Une matrice symétrique $C = (c_{ij})$ (où $c_{ij} = c_{ji}$) dont les coûts sont non négatifs et vérifient l'inégalité triangulaire ($c_{ij} \geq c_{ik} + c_{kj}$) est associée à A.

On dispose d'une flotte homogène de m véhicules domiciliés en un même dépôt.

La valeur de m appartient à l'intervalle $[m_{min}, m_{max}]$, où $m_{min} \geq 1$ et $m_{max} \leq n-1$ (si $m_{min} = m_{max}$, alors m est fixé; si $m_{min} = 1$ et $m_{max} = n-1$, alors m est libre).

Quand m n'est pas fixé, on considère normalement un coût fixe associé à l'utilisation d'un véhicule.

Resoudre un VRP consiste alors à construire l'ensemble des tournées de coût minimal tel que :

a) les véhicules visitent $n-1$ clients à partir du dépôt central et y reviennent;

b) tout client $V \setminus \{v_1\}$ est visité une seule fois;

c) toutes les contraintes sont satisfaites.

On considère les contraintes suivantes:

c1) à chaque client est associée une demande non négative d_i ($d_1 = 0$); un véhicule ne peut pas transporter plus que sa capacité D;

c2) tout client nécessite un temps de service t_i ($t_1 = 0$); la longueur totale d'une tournée quelconque (distance plus temps de service) ne peut dépasser une borne T prédéfinie.

Dans la version du problème que nous traitons, le nombre de véhicules est une variable de décision, à la différence de la formulation présentée au chapitre 2, qui considère ce nombre comme une constante.

5.2 Etat de l'art

Plusieurs approches de type tabou ont été développées pour des problèmes de tournées de véhicules, chacune utilisant différents types de mouvements pour passer d'une solution à une autre.

Pureza et França, 1991, [74] utilisent des échanges de sommets entre deux tournées et Osman, 1993, [69] combine des mouvements 2-échanges. Ce type de mouvements, initialement proposés par Lin, 1965, [58] pour le TSP (parfois appelé 2-opt, car il cherche une amélioration sur deux arêtes à chaque itération) consiste en le remplacement de deux arêtes (v_i, v_{i+1}) et (v_j, v_{j+1}) par deux autres (cf figure 5.1).

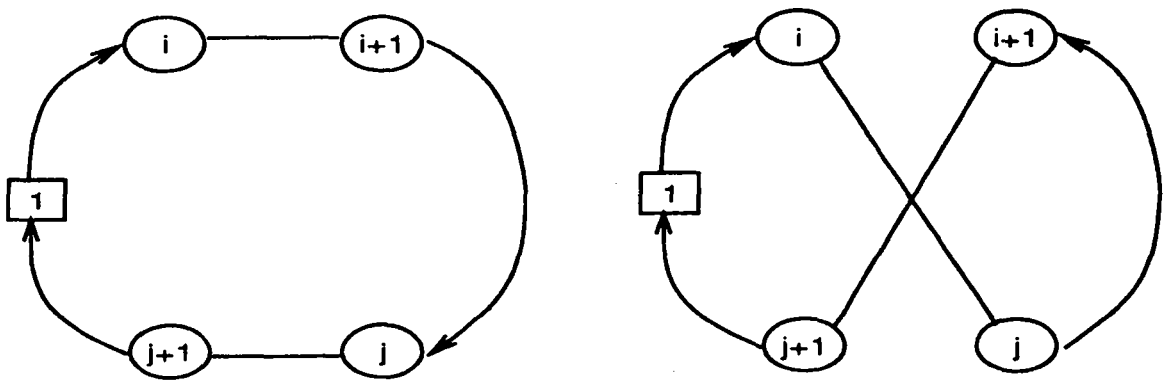


Figure 5.1 : Un mouvement 2-opt

Il faut remarquer que l'orientation du chemin $(v_{i+1}, \cdots, v_j)$ devient inverse de celle de la nouvelle tournée.

La transformation de la tournée produit une amélioration si et seulement si:

$$c_{ij} + c_{i+1,j+1} < c_{i,i+1} + c_{j,j+1}.$$

Cette procédure peut être généralisée en prenant en compte des modifications sur k-arêtes, avec $k > 2$ (procédure k-opt). Mais si on a bien sûr des chances plus grandes d'approcher l'optimum, c'est à un prix élevé: le nombre de transformations envisageables à chaque étape (c'est aussi le nombre de configurations qu'il faudra

essayer pour savoir si on est bien en un optimum local) est de l'ordre de n^k, ce qui est beaucoup trop important pour une heuristique itérative quand k dépasse 2.

On ne pourra pas en général se permettre d'appliquer systématiquement k-opt sous peine de perdre en temps, le gain en qualité. Par contre, il peut être intéressant d'améliorer une procédure 2-opt par l'application de temps à autre, par exemple, de procédures 3-opt, pour aider la méthode à aboutir à l'un des minima locaux. Deux possibilités pour réaliser des mouvements 3-opt sont illustrées sur la figure 5.2.

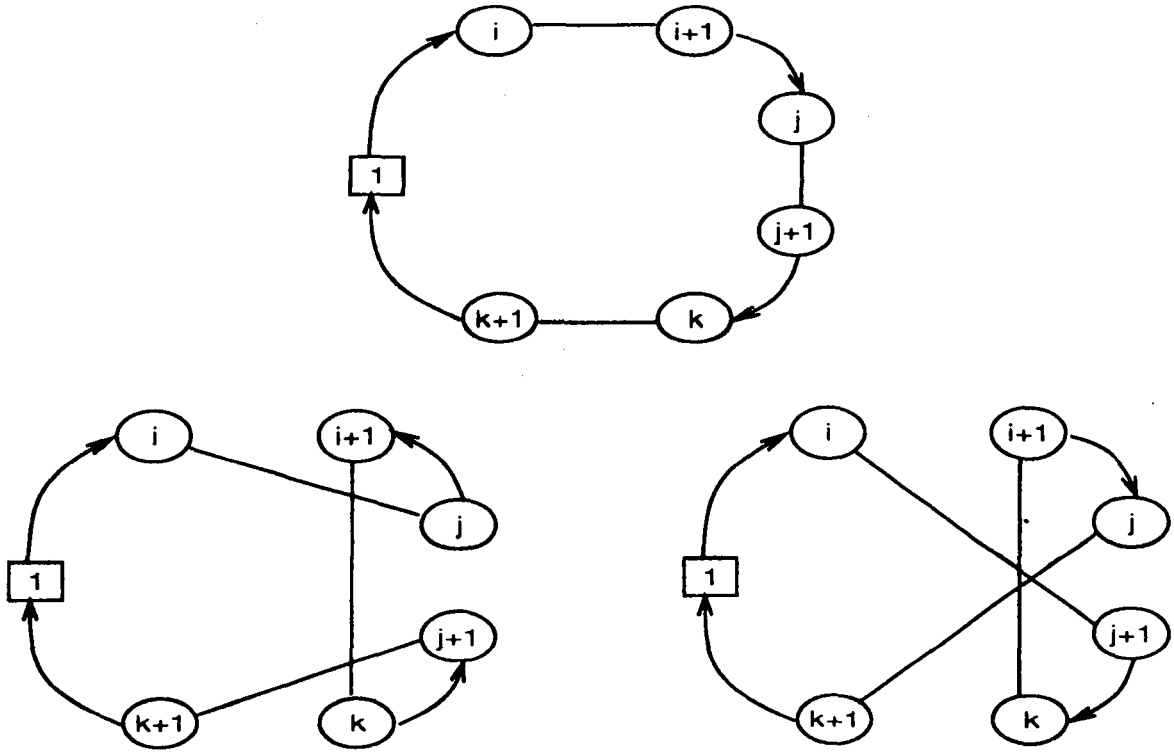


Figure 5.2 : Deux possibilitée de réaliser un mouvement 3-opt

Une autre algorithme a été développé par Semmet et Taillard, 1992, [77] pour la résolution d'un problème réel avec plusieurs contraintes additionnelles, différentes de celles considèrées dans ce rapport. Le mouvement considèré consiste en l'insertion d'un sommet d'une tournée dans une autre.

Une procédure d'insertion a été aussi mise en œuvre par Gendreau, Hertz et Laporte, 1992, [32] qui à la fois transfèrent un sommet d'une tournée dans une autre en exécutant une réoptimisation locale de la tournée courante. Dans les mouvements d'insertion, un sommet i est sélectioné pour être inseré entre deux autres sommets v_p et v_q d'un arc (v_p, v_q) d'une tournée. Alors, le prédécesseur v_{i-1} et le successeur v_{i+1} du sommet v_i sont reliés de telle façon que v_{i-1} devienne le nouveau prédécesseur de

v_{i+1} et le sommet v_i soit le nouveau successeur v_{p+1} de v_p et le nouveau prédecésseur v_{q-1} de v_q. L'insertion d'un sommet est donc évaluée par l'expression :

$$c_{i-1,i+1} - c_{i-1,i} - c_{i,i+1} + c_{p,i} + c_{i,p+1} - c_{p,p+1}$$

qui réprésente la différence de coût de la fonction *Objectif* avant et après le mouvement d'insertion.

Finalement, Taillard, 1992, [81] a utilisé des méthodes par partition de l'ensemble de sommets en différents regroupements (*clusters*). Les "clusters" sont traités indépendamment par la transition des sommets d'une tournée vers une autre.

Cette méthode introduite par Karp, s'appelle aussi "diviser pour régner" ("*divide and conquer*"). Elle consiste d'abord à partager un gros problème en plusieurs sous-problèmes de taille beaucoup plus petite, puis à traiter isolément chaque sous-problème, et enfin à raccorder les solutions partielles entre elles. Il faut trouver un compromis entre la taille des sous-problèmes (pour qu'ils soient faciles à traiter individuellement) et leur nombre (pour que le raccord ne soit trop difficile à mettre en œuvre). C'est ce compromis qui n'est pas toujours facile à obtenir; ce qu'on gagne en découpant le problème ne doit pas être perdu (en temps ou en qualité) quand on "recolle" les solutions partielles de chaque sous-problème entre elles.

Il faut remarquer que les techniques utilisées dans ces algorithmes, essayent d'accroître leurs performances par combinaison de mouvements d'insertion et d'échange. D'autres manières de combiner des voisinages fondées sur la notion de chaîne d'éjection vont être utilisées dans notre algorithme.

5.3 Structures de voisinages composés et chaînes d'éjection

L'utilisation de voisinages composés ajoute un autre niveau de sophistication aux procédures de génération de mouvements. Plusieurs types de structures de voisinages composés ont été utilisées par d'autres méthodes et peuvent se définir comme une compression d'une séquence de mouvements simples dans un seul mouvement composé. Ce type d'approche est appelé *chaîne d'éjection.*

Les chaînes d'éjection combinent et généralisent des idées fondées sur des chemins alternatifs de la théorie de graphes (Berge, 1962; Harary, 1969), la construction de réseaux fondées sur des échanges dans l'optimisation de matroïdes (Lawler 1976; Edmons, 1979), et la restriction de structures pour résoudre des problèmes de programmation en nombres entiers (Glover, 1964). Tous ces domaines ont une composante de recherche de voisinages et suggérent une grande varieté de nouvelles approches pour résoudre les problèmes d'Optimisation Combinatoire.

En interprétant la définition de F. Glover, 1991, [36] nous avons conservé pour une chaîne d'éjection les caractéristiques suivantes:

1. le voisinage d'un mouvement simple est inclus à chaque niveau dans une séquence de voisinages traduisant ainsi des mouvements plus complexes,
2. l'évaluation d'un mouvement à chaque niveau ne dépend pas des mouvements effectués aux niveaux antérieurs de la chaîne,
3. le passage d'un niveau au suivant reproduit une structure incomplète (ou inadmissible) mais garde un haut niveau de légitimité,
4. à chaque niveau, l'obtention d'une structure complète s'obtient par application des mouvements appropriés à la fermeture de la chaîne.

Nous définissons et étudions dans la suite, la chaîne d'éjection que nous avons conçue.

Niveaux d'une chaîne d'éjection

Une chaîne d'éjection peut être vue comme une séquence de niveaux, chacun constitué par trois sommets consécutifs dans une tournée. Pour une meilleure visualisation, on peut imaginer les niveaux de la chaîne comme une pile verticale $P = \{(v_{i-1}, v_i, v_{i+1}), \cdots, (v_{t-1}, v_t, v_{t+1})\}$. La chaîne peut être identifiée par les *sommets centraux* $\Gamma = \{v_i, \cdots, v_t\}$.

Le sommet v_i est *l'extrémité initiale* de la chaîne P; le sommet v_t est *l'extrémité terminale* de la chaîne P. Les ensembles $\Gamma^{-1} = \{v_{i-1}, \cdots, v_{t-1}\}$ de prédécesseurs et $\Gamma^{+1} = \{v_{i+1}, \cdots, v_{t+1}\}$ de successeurs des sommets centraux, sont appelés resp. *sommets d'entrée* et *sommets de sortie*. L' ensemble $W = \Gamma^{-1} \cup \Gamma \cup \Gamma^{+1}$ est l'ensemble des *sommets composants* de la chaîne d'éjection.

Mouvement de transition

Une chaîne réprésente une modification ou transition partielle de la tournée courante vers une nouvelle tournée, déterminée par une séquence de *mouvements de transition*. Chaque sommet central de niveau k, y éjecte le sommet central de niveau $k+1$, et l'éjection du sommet terminal détermine la fin de la chaîne d'éjection. Plus précisement, la transition peut se caractériser par l'enlevement consécutivement des arcs (v^k_{p-1}, v^k_p), (v^k_p, v^k_{p+1}) et la création des arcs (v^{k+1}_{q-1}, v^k_p), (v^k_p, v^{k+1}_{q+1}), où k identifie le niveau courant de la chaîne.

Le sommet initial laisse sa place vide et le sommet terminal est détaché, restant ainsi libre pour être replacé.

Le schéma d'une chaîne d'éjection est donné sur la figure 5.3. Les liaisons qui ont été modifiées par le mouvement de transition sont réprésentées par les arcs diagonaux, en trait plein, qui formeront les arcs des nouvelles tournées. Les arcs horizontaux, en pontillé, sont ceux qui ont été enlevés des anciennes tournées.

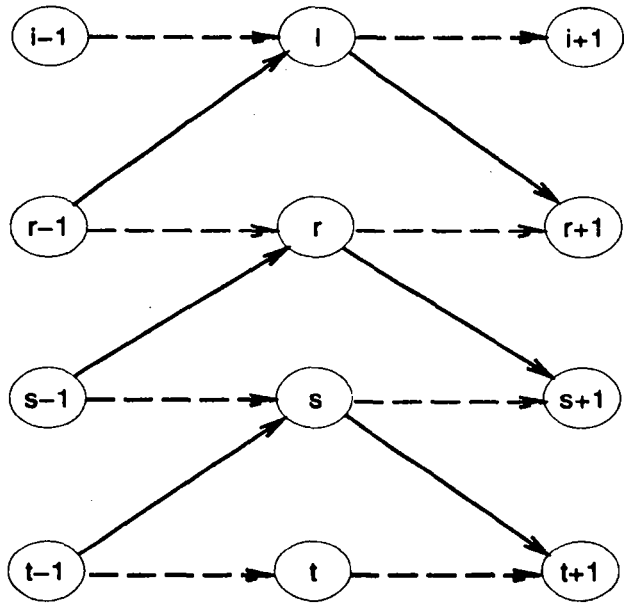


Figure 5.3 : Une chaîne d'éjection de quatre niveaux

Légitimité d'une chaîne d'éjection

Pour qu'une chaîne d'éjection ait une structure légitime, il suffit que chaque sommet central n'arrive qu'une fois dans la chaîne, c'est-à-dire qu'il ne réapparaisse pas comme une autre sommet composant.

Si un sommet apparaît comme un sommet d'entrée (resp. un sommet de sortie), il peut réapparaître une et une seule fois comme un sommet de sortie (resp. un sommet d'entrée), sans violer cette restriction.

Une exception est possible pour le sommet v_1 (dépôt) qui peut réapparaître plusieurs fois comme un sommet d'entrée et (ou) un sommet de sortie à une place quelconque. Par contre, il ne peut jamais devenir un sommet central.

Coût d'une transition

La caractérisation précédente de la légitimité suppose qu'aucun arc ne puisse être ajouté ou enlevé plus d'une fois par mouvement de transition. Ainsi, la modification associée à l'éjection $v_r^k \rightarrow v_s^{k+1}$ est donnée par :

$$c(v_{s-1}^{k+1}, v_r^k) + c(v_r^k, v_{s+1}^{k+1}) - c(v_{r-1}^k, v_r^k) - c(v_r^k, v_{r+1}^k)$$

Mouvements de fermeture

Pour compléter la transition de la tournée courante vers une nouvelle tournée, sont utilisés des *mouvements de fermeture*, qui ne laissent pas de places vides et n'enlèvent pas de sommets.

Ces mouvements sont de deux types:

Type I remplacer l'extrémité initiale v_i par l'extrémité terminale v_t.
Type II ajouter un arc (v_{i-1}, v_{i+1}) pour lier les sommets respectif d'entrée et de sortie de l'extrémité initiale, et insérer v_t entre deux sommets consécutifs v_p et $v_q = v_{p+1}$.

Une illustration de ces mouvements est donnée sur les figures 5.4 et 5.5.

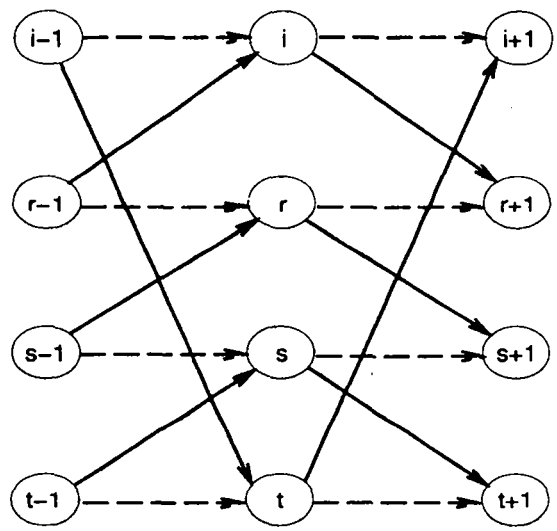


Figure 5.4 : Une chaîne avec un mouvement de fermeture du Type I

Coût d'une nouvelle solution

A partir des deux derniers mouvements, le coût de la modification de la solution peut être calculée.

Le mouvement du Type I modifie la chaîne d'éjection en ajoutant des arcs (v_{i-1}, v_t) et (v_t, v_{i+1}), tandis que le mouvement du Type II modifie la chaîne en ajoutant les arcs (v_{i-1}, v_{i+1}), (v_p, v_t), (v_t, v_q) et en ôtant l'arc (v_p, v_q).

On peut, alors séparer les coûts des opérations effectuées dans la chaîne d'éjection en quatre composantes :

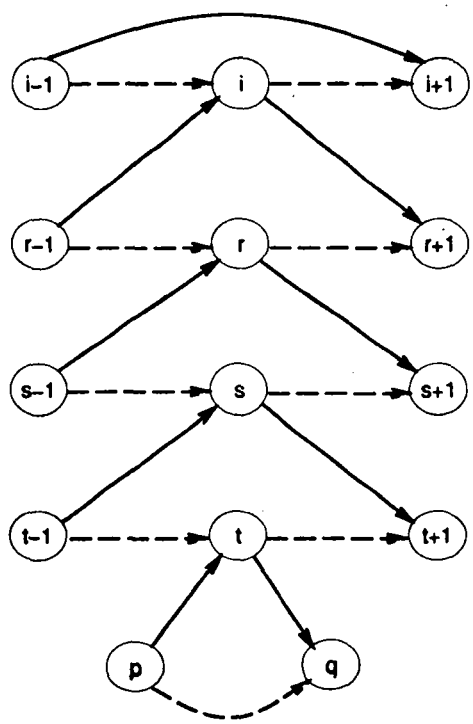


Figure 5.5 : Une chaîne avec un mouvement de fermeture du Type II

1. le remplacement de v_i par v_t,

$$\Psi(v_t, v_i) = c(v_{i-1}, v_t) + c(v_t, v_{i+1})$$

2. la liaison entre le prédécesseur et le successeur de v_i,

$$\Omega(v_i) = c(v_{i-1}, v_{i+1})$$

3. l'insertion de v_t entre v_p et v_q

$$\Phi(v_t, v_p) = c(v_p, v_t) + c(v_t, v_{p+1}) - c(v_p, v_{p+1})$$

4. l'éjection de v_s débutée par v_r pour chaque niveau k ($k = 0, \cdots, l(t)$),

$$\Upsilon(v_r, v_s) = \sum_{0<k\leq l(t)} \left[c(v_{s-1}^k, v_r^{k-1}) + c(v_r^{k-1}, v_{s+1}^k)\right] - \left(\sum_{0\leq k\leq l(t)} \left[c(v_{r-1}^k, v_r^k) + c(v_r^k, v_{r+1}^k)\right]\right)$$

Les gains correspondants aux chaînes d'éjection sont donnés respectivement par les expressions suivantes:

$$\begin{aligned} \Delta F_I &= \Psi(v_t, v_i) + \Upsilon(v_r, v_s) \\ \Delta F_{II} &= \Omega(v_i) + \Upsilon(v_r, v_s) + \Phi(v_t, v_p) \end{aligned}$$

où ΔF_I et ΔF_{II} sont les différences entre les valeurs de la fonction *Objectif* avant et après les chaînes d'éjection, avec respectivement des mouvements de fermeture de Type I et II.

Valeur d'éjection et de preuve

Le coût de la modification provoquée par le mouvement de transition, est appelé *valeur_ejection*, et la *valeur_preuve_1* ou *valeur_preuve_2* est le coût de modification de la tournée pour des mouvements de fermeture du Type I et II.

Si une chaîne d'éjection satisfait la *condition de légitimité* (en considèrant v_p et v_q comme des *sommets composants* pour les mouvements du Type II) on obtient :

$$\begin{aligned} valeur\_preuve\_1 &= valeur\_ejection + \Psi(v_t, v_i) \\ valeur\_preuve\_2 &= valeur\_ejection + \Omega(v_i) + \Phi(v_t, v_p) \end{aligned}$$

5.4 Description générale de l'algorithme

Ce paragraphe contient la description de l'algorithme TABOU_CHAINE.

On utilise les notations suivantes.

La solution S est définie par un ensemble de m tournées $R_1, \cdots, R_m$ avec $m \in [1, m_{max}]$; $R_r = \{v_1, v_{r_1}, v_{r_2}, \cdots, v_1\}$ et chaque sommet $v_i (i > 1)$ est affecté à une seule tournée.

Ces tournées peuvent être admissibles ou inadmissibles par rapport aux contraintes de capacité et de longueur.

Par convention, on écrit $v_i \in R_r$ si v_i est une composante de R_r,
et $(v_i, v_j) \in R_r$ si v_i et v_j sont deux sommets consécutifs de R_r.

A chaque solution S (admissible ou non), on associe la fonction *Objectif* :

$$F(S) = \sum_r \sum_{(v_i, v_j) \in R_r} c_{ij}.$$

TABOU_CHAINE est décomposé en sept phases.

0. Initialisation,

1. Recherche Préliminaire,

2. Intensification,
3. Diversification,
4. Demi-Oscillation,
5. Oscillation,
6. Post-Optimisation.

Dans ces phases sont utilisés différents paramètres pour la gestion de la liste Tabou. Comme Taillard, 1991, [80] et Gendreau, Hertz et Laporte, 1990, [31] le nombre d'itérations t où un mouvement reste dans la liste tabou n'est pas un nombre constant, mais tiré aleatoirement uniformement dans un intervalle $[t_{min}, t_{max}]$.

Pour chaque phase de l'algorithme, plusieurs valeurs de ces paramètres sont testés dans l'intervalle $[3, 20]$ et les meilleurs choisis.

La passage d'une solution S à une solution S' se fait à partir de mouvements composés obtenus par des voisinages successifs exploités à chaque niveau d'une chaîne d'éjection. Dans les phases de recherche préliminaire et d'intensification, des mouvements de fermeture du Type II sont utilisés, dans les autres phases, les mouvements du Type I.

Pour les mouvements de fermeture du Type I, on commence par choisir un sommet v_i qui devient l'extrémité initiale de la chaîne d'éjection. Pour cela, on déconnecte le sommet v_i de sa position courante et on sélectionne un sommet v_j qui sera éjecté par v_i. Pour le choisir, on identifie le coût associé à la composante de déconnection du sommet v_i:

$$d(v_i) = -[c(v_{i-1}, v_i) + c(v_i, v_{i+1})]$$

L'éjection provoquée par le mouvement de transition place le sommet v_i à la place de v_j et peut s'exprimer comme suit :

$$m(v_i, v_j) = c(v_{j-1}, v_i) + c(v_i, v_{j+1}) - c(v_{j-1}, v_j) - c(v_j, v_{j+1})$$

Le sommet v_i choisi est celui qui minimise la somme $d(v_i) + m(v_i, v_j)$.

Cette procédure d'éjection peut se généraliser aux autres niveaux de la chaîne. Comme l'indépendence entre les quantités $m(v_i, v_j)$ est gardée par la condition de légitimité dans le déroulement de la chaîne d'éjection, il est mieux de précalculer ces valeurs, car elles seront souvent utilisées.

Ainsi, de façon à augmenter la performance de l'algorithme, on utilise une matrice M d'éléments m_{ij} pour stocker les valeurs d'éjection $m(v_i, v_j)$.

Nous nous limiterons à chaque itération au calcul des valeurs m_{ij} correspondant aux sommets composants de la chaîne d'éjection.

A chaque niveau de la chaîne d'éjection, la méthode choisit le meilleur sommet v_j donné par le meilleur mouvement de transition (éjection). La modification de la valeur de la fonction *Objectif* est calculée pour le mouvement de fermeture correspondant.

A chaque itération, le mouvement choisi est celui qui correspond au niveau de plus forte évaluation.

Pour les mouvements de fermeture du Type II, on procède d'une façon identique. Dans la sélection du meilleur sommet devait devenir extrémité initiale de la chaîne, on considère en plus le coût associé à la liaison entre le prédécesseur et le successeur du sommet v_i.

On a donc pour $d(v_i)$ la modification suivante:

$$d(v_i) = \Omega(v_i) - [c(v_{i-1}, v_i) + c(v_i, v_{i+1})]$$

L'examen de la modification de la solution se fait par sélection de la meilleure insertion du sommet v_j (entre deux sommets consécutifs v_p et v_q) correspondant au niveau courant de la chaîne d'éjection. De façon à réduire l'effort de calcul dans les itérations suivantes, on fait un précalcul des valeurs d'éjection $m(v_i, v_j)$ et des valeurs d'insertion $\Phi(v_j, v_p)$ qui serons stockées respectivement dans les matrices M d'elements m_{ij} et B d'elements b_{jp}.

Pour garder la légitimité de la chaîne d'éjection, il suffit de stipuler que les sommets v_p et v_q doivent être considérés comme des sommets composants. Ainsi, à chaque itération, on a besoin de recalculer que les valeurs m_{ij} et b_{jp} correspondants aux sommets composants.

5.5 Description des procédures

5.5.1 Phase d'initialisation

La phase d'initialisation consiste en la génération d'une solution initiale S où tous les clients i $(i = 2, \cdots, n)$ sont visités par construction de m_{max} tournées du type (v_1, v_i, v_1).

Pour tester l'influence de la solution initiale sur la qualité de la solution finale nous pouvons utiliser un algorithme constructif quelconque. Ainsi, nous avons choisi l'algorithme de Clark & Wright (voir chapitre 2).

Expérimentalement, notre algorithme trouve de meilleurs solutions finales en partant de la solution initiale S plutot qu'en partant de la solution donnée par l'algorithme de Clark & Wright.

5.5.2 Phase de recherche préliminaire

A chaque itération de la phase de recherche préliminaire, une chaîne d'éjection est construite et le mouvement composé correspondant au niveau de plus forte évaluation est sélectioné, s'il est *non tabou* ou satisfait le critère d'aspiration. L'aspiration est réalisée si la solution obtenue par le mouvement tabou est meilleure que toute solution examinée jusqu'alors.

Pour chaque valeur de preuve, on ne recalcule pas le coût total $F(S)$, mais seulement le gain par rapport à la solution courante.

5.5.3 Phase d'intensification

Dans la phase d'intensification, l'algorithme procède à une recherche préliminaire à partir de la meilleure solution obtenue jusqu'alors. Si une meilleure solution est trouvée, la phase d'intensification est réactivée. Lorsque l'intensification courante ne trouve pas une meilleure solution, alors une phase de diversification est initialisée.

5.5.4 Phase de diversification

La diversification est une phase réalisée pendant un nombre fixe d'itérations. On utilise une mémoire à long terme fondée sur la fréquence. Pour cela, on utilise une matrice MLT d'elements mlt_{ij} pour stocker le nombre de fois où une éjection $v_i \rightarrow v_j$ appartient à un mouvement composé choisi.

A chaque itération, la matrice $\mu.MLT$ est additionnée à la matrice de coûts C d'elements c_{ij} produisant un coût additionnel dans la fonction *Objectif*.

Le paramètre μ est un facteur de pénalité qui dépend de chaque VRP particulier.

En général, le nombre de mouvements possibles augmente avec la taille du problème, donc la fréquence de chaque mouvement va diminuer. Ainsi, pour diversifier la recherche nous choisirons μ uniformement entre $0.1\sqrt{n}$ et $0.5\sqrt{n}$.

5.5.5 Phases d'oscillation et demi-oscillation

Une stratégie d'oscillation est aussi suggèrée par Glover, 1989, [35], Glover, Taillard et de Werra, 1992, [39], et Golver et Laguna, 1992, [38]. Elle a pour but de jouer un rôle intermédiaire entre les stratégies d'intensification et de diversification. Dans notre application, on utilise deux types d'oscillation correspondants à deux phases de l'algorithne: phase de demi-oscillation et d'oscillation. Dans ces phases, on commence par choisir l'extrémité initiale de la chaîne comme dans les phases antérieures, on obtient ainsi un mouvement de preuve initial admissible.

Pour les autres niveaux de la chaîne, dans la phase demi-oscillation, on accepte des mouvements de transition admissibles même si ceci amène à des mouvements

de preuve non admissibles. De même, dans la phase d'oscillation, on accepte les mouvements de transition et de preuve non admissibles.

Ces procédures supposent que le passage par des mouvement totalement ou partiellement inadmissibles peut amener à des mouvements complets admissibles à des niveaux ultérieurs de la chaîne d'éjection.

A la fin de la chaîne d'éjection, on choisit le niveau k^* associé au meilleur mouvement admissible.

5.5.6 Phase de post-optimisation

A chaque itération de toutes les phases décrites ci-dessus, quand un mouvement produit une solution dans l'intervalle $[F(S^*), \alpha.F(S^*)]$ $(\alpha > 1)$, où $F(S^*)$ réprésente la valeur de la meilleure solution obtenue jusqu'alors, la phase de post-optimisation est activée.

Cette phase se traduit par la résolution approchée des TSPs correspondants à toutes les tournées modifiées par le mouvement composé choisi.

5.6 L'algorithme TABOU_CHAINE

L'algorithme TABOU_CHAINE utilise deux procédures centrales:
RECHERCHE_I(L) et RECHERCHE_II(L) correspondant respectivement à deux types de recherche obtenues par les mouvements de fermeture de Type I et II. Ces procédures se servent d'un vecteur $L = (S, t_{min}, t_{max}, \alpha, n_{max})$ défini de la façon suivante:

S	réprésente l'ensemble de tournées $R_1, \cdots, R_m$,
t_{min}, t_{max}	les bornes du nombre d'itérations où un mouvement reste tabou,
n_{max}	le nombre d'itérations sans aucune amélioration de la fonction Objectif,
α	un facteur d'accroissement de la meilleure solution trouvée.

PROCEDURE RECHERCHE_I(L)

Tant que n_{max} n'est pas atteint faire
Pour chaque sommet v_i et v_j $(i, j = 2, \cdots, n; i \neq j)$,
a) identifier les sommets v_{i-1}, v_{j-1} et v_{i+1}, v_{j+1},
b) calculer $d(v_i)$ et $m(v_i, v_j)$ $(v_i \neq v_{i-1} \neq v_{i+1})$.

Identifier la paire $(v_{i^\bullet}, v_{l^\bullet}) \subset N(S)$ telle que:

a) $\Upsilon(v_{i^*}, v_{t^*}) = \min\{d(v_i) + m(v_i, v_t)\}$
b) $(v_{i^*}, v_{t^*}) \notin T$ ou $a(S,m) < F(S^*)$
et définir:
$a(S,m) = F(S) + d(v_{i^*}) + m(v_{i^*}, v_{t^*}) + \Psi(v_{t^*}, v_{i^*})$

Initialiser la chaîne d'éjection:
a) poser :
$k = 0$
$\Gamma = \{v_{i^*}\}$
b) calculer la valeur du mouvement de transition $v_{i^*} \rightarrow v_{t^*}$
$valeur\_ejection(k+1) = \Upsilon(v_{i^*}, v_{t^*})$

Tant que la condition de légitimité est vérifiée faire
Exécuter le mouvement :
a) poser :
$k = k + 1$
$\Gamma = \Gamma \cup \{v_{t^*}\}$
b) calculer le gain ΔF_l correspondant à la modification de S
$valeur\_preuve\_1(k) = valeur\_ejection(k) + \Psi(v_{i^*}, v_{t^*})$
c) déterminer le meilleur niveau l^*
$valeur\_preuve\_1(l^*) = \min\limits_{0<l\leq k}\{valeur\_preuve\_1(l), \infty\}$

Pour chaque sommet v_j $(j = 2, \cdots, n)$ déterminer la paire $(v_{t^*}, v_{j^*}) \subset N(S)$ telle que:
a) $\sigma(v_{j^*}) = \min\{m(v_{t^*}, v_j)\}$
b) $(v_{t^*}, v_{j^*}) \notin T$ ou $a(S,m) < F(S^*)$
et définir:
$a(S,m) = F(S) + valeur\_ejection(k) + m(v_{t^*}, v_j) + \Psi(v_{t^*}, v_j)$
Si v_{j^*} a été trouvé, calculer:
$valeur\_ejection(k+1) = valeur\_ejection(k) + m(v_{t^*}, v_{j^*})$
et enregistrer v_{j^*} le nouveau sommet terminal v_{t^*}.
fin Tant que

Identifier la valeur de la nouvelle solution S
$F(S) = F(S) + valeur\_preuve\_1(l^*)$
Si $F(S) < F(S^*)$ alors S représente les nouvelles solutions S^* et $\tilde{S}$,
réinitialiser le nombre d'itérations sans aucune amélioration de la fonction Objectif,
sinon si $F(S) < F(\tilde{S})$ alors S est la nouvelle solution $\tilde{S}$.

Pour chaque paire (v_i^{k+1}, v_j^k) $(k = 0, \cdots, l^*)$; $v_i, v_j \in \Gamma$, engendrer t uniformement dans l'intervalle $[t_{min}, t_{max}]$ et rendre cette paire tabou pendant t itérations.
Rendre tabou la paire (v_i^0, v_j^1).

Identifier les ensembles:

$$S_1 = \left\{(v_{i-1}^0, v_j^{l^*}), (v_j^{l^*}, v_{i+1}^0)\right\}$$

$$S_2 = \left\{\bigcup_{k=2}^{l^*} \left((v_{i-1}^k, v_j^{k-1}), (v_j^{k-1}, v_{i+1}^k)\right)\right\}$$

$$S_3 = \left\{\bigcup_{k=1}^{l^*} \left((v_{j-1}^k, v_j^k), (v_j^k, v_{j+1}^k)\right)\right\}$$

et déterminer la nouvelle solution S telle que :

$$S = S \cup S_1 \cup S_2 \setminus S_3$$

fin Tant que

PROCEDURE RECHERCHE_II(L)

Tant que n_{max} n'est pas atteint faire

Pour chaque sommet v_i et v_j $(i, j = 2, \cdots, n; i \neq j)$,

a) identifier les sommets v_{i-1}, v_{j-1} et v_{i+1}, v_{j+1},

b) calculer $d(v_i)$, $m(v_i, v_j)$ $(v_i \neq v_{i-1} \neq v_{i+1})$ et $\Phi(v_i, v_j)$.

Identifier la paire $(v_{i^\bullet}, v_{t^\bullet}) \subset N(S)$ telle que:

a) $\Upsilon(v_{i^\bullet}, v_{t^\bullet}) = \min\{d(v_i) + m(v_i, v_t)\}$,

b) $(v_{i^\bullet}, v_{t^\bullet}) \notin T$ ou $a(S, m) < F(S^*)$,

calculer la meilleure insertion du sommet $v_{t^\bullet}$ correspondant au niveau $k + 1$ tel que :

$$\Phi(v_{t^\bullet}, v_{p^\bullet}) = \min_{p \in V \setminus \{v_{i^\bullet}\}} \{\Phi(v_{t^\bullet}, v_p)\}$$

et définir:

$$a(S, m) = F(S) + d(v_{i^\bullet}) + m(v_{i^\bullet}, v_{t^\bullet}) + \Phi(v_{t^\bullet}, v_{i^\bullet})$$

Initialiser la chaîne d'éjection:

a) poser :

$$k = 0$$

$$\Gamma = \{v_{i^\bullet}\}$$

b) calculer la valeur du mouvement de transition $v_{i^\bullet} \rightarrow v_{t^\bullet}$ et la correspondant valeur d'insertion:

$$valeur\_ejection(k + 1) = \Upsilon(v_{i^\bullet}, v_{t^\bullet})$$

$$valeur\_insertion(k + 1) = \Phi(v_{t^\bullet}, v_{p^\bullet})$$

Tant que la condition de légitimité est vérifiée faire
Exécuter le mouvement :
a) poser :
$k = k + 1$
$\Gamma = \Gamma \cup \{v_{t^*}\}$
b) calculer le gain ΔF_{II} correspondant à la modification de S
$valeur\_preuve\_2(k) = valeur\_ejection(k) + \Phi(v_{i^*}, v_{t^*})$
c) déterminer le meilleur niveau l^*
$valeur\_preuve\_2(l^*) = \min\limits_{0<l\leq k}\{valeur\_preuve\_2(l), \infty\}$

Pour chaque sommet v_j $(j = 2, \cdots, n)$ déterminer la paire $(v_{t^*}, v_{j^*}) \subset N(S)$ telle que:
a) $\sigma(v_{j^*}) = \min\{m(v_{t^*}, v_j)\}$,
b) $(v_{t^*}, v_{j^*}) \in T$ ou $a(S, m) < F(S^*)$,
calculer la meilleure insertion du sommet v_j pour le niveau suivant de la chaîne d'éjection telle que :
$\Phi(v_{j^*}, v_{p^*}) = \min\{\Phi(v_{j^*}, v_p)\}$
et définir:
$a(S, m) = F(S) + valeur\_ejection(k) + m(v_{t^*}, v_j) + \Phi(v_{j^*}, v_p)$
Si v_{j^*} et v_{p^*} ont été trouvés, calculer:
$valeur\_ejection(k + 1) = valeur\_ejection(k) + m(v_{t^*}, v_{j^*})$
$valeur\_insertion(k + 1) = \Phi(v_{j^*}, v_{p^*})$
et enregistrer v_{j^*} le nouveau sommet terminal v_{t^*}.
fin Tant que

Identifier la valeur de la nouvelle solution S
$F(S) = F(S) + valeur\_preuve\_2(l^*)$
Si $F(S) < F(S^*)$ alors S représente les nouvelles solutions S^* et $\tilde{S}$,
réinitialiser le nombre d'itérations sans aucune amélioration de la fonction Objectif,
sinon si $F(S) < F(\tilde{S})$ alors S est la nouvelle solution $\tilde{S}$.

Pour chaque paire (v_i^{k+1}, v_j^k) $(k = 0, \cdots, l^*)$; $v_i, v_j \in \Gamma$, engendrer t uniformement dans l'intervalle $[t_{min}, t_{max}]$ et rendre cette paire tabou pendant t itérations.
Rendre tabou la paire (v_i^0, v_j^1).

Identifier les ensembles:
$S_1 = \left\{(v_{i-1}^0, v_{i+1}^0), (v_p^{l^*}, v_t^{l^*}), (v_t^{l^*}, v_{p+1}^{l^*})\right\}$

$$S_2 = \left\{ \bigcup_{k=2}^{l^*} \left((v_{i-1}^k, v_j^{k-1}), (v_j^{k-1}, v_{i+1}^k) \right) \right\}$$

$$S_3 = \left\{ \bigcup_{k=1}^{l^*} \left((v_{j-1}^k, v_j^k), (v_j^k, v_{j+1}^k) \right), (v_p^{l^*}, v_{p+1}^{l^*}) \right\}$$

et déterminer la nouvelle solution S telle que :

$S = S \cup S_1 \cup S_2 \setminus S_3$

fin Tant que

ALGORITHME TABOU_CHAINE

0. **Initialisation**
   Construtions d'une solution initiale $S \in X$, définie par un ensemble de m_{max} tournées $R_1, \cdots, R_{n-1}$, $R_r = \{v_1, v_i, v_1\}$ $(i = 2, \cdots, n; r = i - 1)$.
   Initialiser :

$S^* = S$	la meilleur solution trouvée,
$T = \emptyset$	la liste Tabou,
$MLT = \emptyset$	la mémoire à long terme.

1. **Recherche Préliminaire**
   RECHERCHE_II(L_1)
   Si une amélioration a été trouvée, ou $F(\tilde{S}) \in [F(S^*), \alpha F(S^*)]$, poser resp. $S = S^*$ ou $S = \tilde{S}$ et appeller
   RECHERCHE_I(L_6) $\forall R_r \in S$
   Poser $S = S^*$ et aller à Intensification.

2. **Intensification.**
   Tant qu' une amélioration est trouvée faire
   Initialiser $T = \emptyset$
   RECHERCHE_II(L_2)
   Poser $S = S^*$
   fin Tant que

   Si une amélioration a été trouvée ou $F(\tilde{S}) \in [F(S^*), \alpha F(S^*)]$,
   poser resp. $S = S^*$ ou $S = \tilde{S}$ et appeler RECHERCHE_I(L_6) $\forall R_r \in S$. Poser $S = S^*$ et aller à Diversification.

3. **Diversification.**
   a) Déterminer les fréquences de la mémoire à long terme enregistrées dans les dernieres phases.
   b) RECHERCHE_I(L_3)
   c) Enlever les penalisations ajoutées dans le pas 3.a
   d) Si à la fin du pas 3.b, la solution a été améliorée, appeler RECHERCHE_I(L_6) $\forall R_r \in S$ et aller à Recherche Preliminaire,
   sinon si $F(\tilde{S}) \in [F(S^*), \alpha F(S^*)]$, appeller RECHERCHE_I($L_6$) $\forall R_r \in S$.
   Poser $S = S^*$ et aller à la Demi-Oscillation.

4. **Demi-Oscillation.**
   a) RECHERCHE_I(L_4)
   b) Si la solution s'est améliorée, dans dans un niveau $l^* > 1$, appeller RECHERCHE_I(L_6) $\forall R_r \in S$ et aller à la Recherche Préliminaire,
   Sinon si $F(\tilde{S}) \in [F(S^*), \alpha F(S^*)]$, appeller RECHERCHE_I($L_6$) $\forall R_r \in S$.
   Poser $S = S^*$ et aller à la Oscillation.

5. **Oscillation.**
   a) RECHERCHE_I(L_5)
   b) Si la solution s'est améliorée, dans dans un niveau $l^* > 1$, appeller RECHERCHE_I(L_6) $\forall R_r \in S$ et aller à la Recherche Préliminaire,
   Sinon si $F(\tilde{S}) \in [F(S^*), \alpha F(S^*)]$, appeller RECHERCHE_I($L_6$) $\forall R_r \in S$.
   Poser $S = S^*$ et aller à Post-Optimisation.

6. **Post-Optimisation.**
   RECHERCHE_I(L_6) pour $R_1, \cdots, R_m$ de S
   Si la solution s'est améliorée, poser $S = S^*$ et aller à la Recherche Préliminaire,
   sinon Fin.

Il reste à commenter le choix des paramétres utilisés dans RECHERCHE_I, RECHERCHE_II et TABOU_CHAINE. On obtient différentes phases de l'algorithme à travers de simples modifications du pas 3.c de la procédure RECHERCHE_I.

Ainsi, dans la phase de diversification, on ajoute une pénalisation à la fonction *Objectif* de la façon suivante:

$$valeur\_preuve\_1(l^*) = \min_{0<l\leq K} \{valeur\_preuve\_1(l) + \mu.mlt_{i^*,t^*}, \infty\}$$

De même, pour différencier les phases 3, 4, et 5 de TABOU_CHAINE il suffit que le meilleur niveau l^* soit choisi selon le type d'admissibilité considérée en chacune de ces phases.

Plusieurs expériences ont été faites pour trouver les meilleurs valeurs de t_{min} et t_{max} pour les diverses phases de l'algorithme.

On a finalement adopté $t_{min} = 5$ et $t_{max} = 10$ pour L_1 et L_2; $t_{min} = 8$ et $t_{max} = 13$ pour les phases 3, 4, et 5; et $t_{min} = 3$ et $t_{max} = 7$ pour la phase 6.

Un bon choix de α est aussi important.

Si α est trop petit, la procédure de post-optimisation est rarement utilisée. Par contre si α est trop grand (supérieur à 1.05), on risque d'appeller souvent cette procédure sans aucune amélioration de la valeur de la fonction *Objectif*.

On a trouvé un bon compromis pour $\alpha = 1.02$.

Finalement, la valeur de n_{max} a été prise égale à $10n$ pour L_1 et L_2, $3n$ pour L_3, L_4 et L_5 et $50n'$ pour L_6, où n' est le nombre d'éléments de la tournée correspondante.

Chapitre 6

Analyse des résultats expérimentaux

La performance de l'algorithme TABOU_CHAINE est évaluée pour un ensemble de 14 problèmes tests de la littérature. Ces problèmes sont connus comme difficiles à résoudre et ont été souvent utilisés comme un "benchmark" pour comparer les algorithmes de résolution du VRP.

La taille de ces problèmes varie de 50 à 199 clients. Dans les premiers dix problèmes, les clients sont générés uniformement dans un plan Euclidien, dans les suivants, les clients apparaissent regroupés, comme dans les problèmes réels . Tous les véhicules ont la même capacité et sont domiciliés dans le dépôt. Il n'y a aucune contrainte sur le nombre de véhicules utilisés.

Les données pour les trois premiers problèmes sont issues de Christofides et Eilon, 1969, [13]. Les données pour le problème 4 sont obtenues en ajoutant les clients des problèmes 1 et 3 avec le dépôt et les capacités du problème 3. Le problème 5 est obtenu en ajoutant les premiers 49 clients du problème 2 aux clients du problème 4 avec le dépôt et les capacités des véhicules du problème 4. Dans les premiers cinq problèmes, ne sont pas considérées des contraintes de longueur de tournée. Les problèmes de 6 à 10 sont les mêmes que ceux de 1 à 5 avec des contraintes de longueur de tournée et un temps de chargement (ou déchargement) de 10 unités. Les données des problèmes 11 et 12 sont issues de Christofides, Mingozzi et Toth, 1979, [15]. Ces problèmes ne considérent que des contraintes de capacité. Les problèmes 13 et 14 sont les mêmes que les problèmes 11 et 12 mais avec des contraintes additionnelles de longueur de tournée.

Les caratéristiques de ces problèmes sont données dans le tableau 6.1.

Les comparaisons sont faites entre TABOU_CHAINE et d'autres algorithmes heuristiques dont les solutions ont été publiées pour ces 14 problèmes:

- **CW**, l'algorithme des "savings", Clark et Wright, 1964, [18]

Problème	n	D	T	t
1	50	160	∞	0
2	75	140	∞	0
3	100	200	∞	0
4	150	200	∞	0
5	199	200	∞	0
6	50	160	200	10
7	75	140	160	10
8	100	200	230	10
9	150	200	200	10
10	199	200	200	10
11	120	200	∞	0
12	100	200	∞	0
13	120	200	720	50
14	100	200	1040	90

Tableau 6.1 : Caractéristiques des 14 problèmes tests

- **MJ**, l'algorithme des "savings" généralisé, Mole et Jamesson, 1976, [65]
- **AG**, l'algorithme PSA-T, Altinkemer et Gavish, 1991, [1]
- **DV**, l'algorithme MBSA, Desrochers et Verhoog, 1989, [22]
- **GM**, l'algorithme SWEEP, Gillet et Miller, 1974, [33]
- **CMT1**, l'algorithme en deux-phases, Christofides, Mingozzi et Toth, 1979, [15]
- **FJ**, l'algorithme en deux-phases, Fisher et Jaikumar, 1981, [27]
- **CMT2**, l'algorithme de recherche arborescente incomplet, Christofides, Mingozzi et Toth, 1979, [15]
- **OSA**, l'algorithme Recuit Simulé, Osman, 1993, [69]
- **PF**, l'algorithme Tabou, Pureza et França, 1991, [74]
- **OTS**, l'algorithme Tabou, Osman, 1993, [69]
- **T**, l'algorithme Tabou, Taillard, 1992, [81]
- **GHL** l'algorithme Tabou, Gendreau, Hertz et Laporte, 1992, [32]

Plusieurs remarques ont été faites sur l'analyse des resultats par Gendreau, Hertz et Laporte, 1992, [32]. Ces auteurs ont démontré que les solutions entières sous-estiment probablement la valeur réelle de la solution ce qui entraîne plusieurs difficultées pour la comparaison des résultats. Ainsi, par exemple, dans le problème 1, l'algorithme GHL arrive aux résultats 524.61, 521 et 508 en utilisant au cours de l'algorithme respectivement des distances réelles, arondies par excès ou par défaut. En effet, récemment, Hadjiconstantinou et Christofides, 1993, [42] dans une conférence au Canada, ont prouvé que la valeur 524.61 est la solution optimale de ce problème.

Une autre difficulté provient des problèmes qui considèrent des contraintes de longueur, car certaines tournées peuvent être non-admissibles si l'on considère des distances réelles. Ces auteurs ont encore remarqué que de bien meilleures solutions peuvent être obtenues en violant ces contraintes d'une valeur inférieure à 1 unité.

A notre connaissance, les seules lignes OSA, OTS, T, GHL et TC correspondent à des valeurs de solutions obtenues avec des distances c_{ij} réelles.

Les valeurs des solutions pour ces algorithmes sont données dans le tableau 6.2. Elles sont extraites de Gendreau, Hertz et Laporte, 1992, [32] et peuvent se trouver dans les reférences respectives, sauf pour CW, MJ et GM qui sont données par Christofides, Mingozzi et Toth, 1979, [15].

Les numéros en caractére "gras" représentent les meilleurs solutions avec valeurs de distances réelles. Sur la ligne TC, sont données les valeurs des solutions de TABOU_CHAINE et sur la ligne Z^*, les meilleurs solutions (réelles) connues. Les tournées correspondantes à les solutions de notre algorithme sont présentées dans l'annexe A.

D'après le tableau 6.2, quelques observations peuvent être faites entre la qualité des solutions et la structure du problème. Ainsi, pour les problèmes générés uniformement, l'algorithme GM se montre meilleur que les algorithmes CW, MJ et DV, et un peu plus mauvais que les algorithmes CMT1 et CMT2. Pour les problèmes structurés, le contraire se vérifie, l'algorithme GM est pire que les algorithmes CW, MJ et DV et beaucoup plus mauvais que les algorithmes CMT1 et CMT2 (cf. figure B.1).

TABOU_CHAINE donne toujours des meilleurs résultats que les algorithmes CW, MJ et DV pour tous les 12 problèmes non-structurés ainsi que pour le problème 14 (cf. figure B.2). Par rapport à CMT1 des meilleurs résultats sont aussi obtenues sur 13 problèmes.

L'algorithme GM se montre très compétitif avec notre algorithme pour les problèmes non structurés, mais pourtant TABOU_CHAINE donne de bien meilleurs résultats pour les problèmes structurés (cf. figure B.3). Il faut encore remarquer que l'algorithme GM nécessite de connaître les coordonnées de chaque client, et donc il ne peut pas s'appliquer à des problèmes où seules les matrices de distance (temps) sont disponibles.

Parmi les heuristiques AG, CMT1, FJ et CMT2, aucune analogie n'est détectable entre elles, mais FJ se révèle supérieure.

Probl.	1	2	3	4	5	6	7
n	50	75	100	150	199	50	75
CW	585	900	886	1204	1540	619	976
MJ	575	910	882	1259	1545	599	969
AG	556	855	860	1085	1351	577	939
DV	586	885	889	1133	1424	593	963
GM	532	874	851	1079	1389	560	933
CMT1	547	883	851	1093	1218	565	969
FJ	524	857	833	1014	1420	560	916
CMT2	534	871	851	1064	1386	560	924
OSA	528	838.62	829.18	1058	1378	**555.43**	**909.68**
PF	536	842	851	1081	...?	560	929
OTS	**524.61**	844	835	1044.35	1334.55	**555.43**	911
T	**524.61**	**835.32**	**828.98**	**1029.64**	**1300.89**	**555.43**	**909.68**
GHL	**524.61**	835.77	829.45	1036.16	1322.65	**555.43**	913.23
TC	**524.61**	850.92	842.58	1072.19	1377.46	561.24	938
Z^*	524.61	835.32	826.14	1029.64	1300.89	555.43	909.68

Probl.	8	9	10	11	12	13	14
n	100	150	199	120	100	120	100
CW	973	1426	1800	1079	831	1634	877
MJ	999	1289	1770	1100	879	1590	883
AG	913	1210	1464	1047	834	1551	874
DV	914	1292	1559	1058	828	1562	882
GM	888	1230	1518	1266	937	1770	949
CMT1	915	1245	1508	1066	827	1612	876
FJ	885	1230	1518	...?	824	...?	848
CMT2	885	1217	1509	1092	816?	1608	878
OSA	866.75	1164.12	1417.85	1176	826	**1545.98**	890
PF	887	1227	...?	1049	826	1631	866?
OTS	866.75	1184	1417.85	1042.11	819.59	1547	866.37?
T	**865.94**	**1164.24**	**1403.21**	1073.05	819.56?	1550.15	866.37?
GHL	**865.94**	1177.76	1418.51	1043.47	819.56?	1573.81	866.37?
TC	883	1204	1464	**1039**	823	1576	872
Z^*	865.94	1162.89	1403.21	1039	819.56	1545.93	866.37

Tableau 6.2 : Comparaison entre TABOU_CHAINE (TC) et d'autres heuristiques

TABOU_CHAINE trouve des meilleures solutions que ces algorithmes pour la plupart des problèmes.

TABOU_CHAINE se montre compétitif avec les algorithmes de Recuit Simulé et Tabou publiés récemment. Par exemple par rapport à OSA et PF, notre algorithme trouve des meilleurs solutions que ces algoritmes respectivement sur 5 et 8 problèmes. Il faut encore remarquer que pour les problèmes 5 et 10, des comparaisons ne peuvent être faites avec PF car les solutions avec cet algorithme ne sont pas connues.

Dans le tableau 6.3, nous présentons l'écart en pourcentage de chaque solution à la meilleure solution connue.

Des graphiques illustratifs de cet ensemble de situations sont présentés à l'annexe B.

Probl.	1	2	3	4	5	6	7
n	50	75	100	150	199	50	75
CW	10.3	7.2	6.8	14.5	15.5	10.3	6.8
MJ	8.8	8.2	6.3	18.2	15.8	7.3	6.1
AG	5.6	2.3	3.9	5.1	3.7	3.7	3.1
DV	10.5	5.6	7.1	22.8	8.6	6.3	5.5
GM	1.4	4.4	2.9	4.6	6.3	0.8	2.5
CMT1	4.1	5.4	2.9	5.8	0	1.7	6.1
FJ	0	2.5	0.8	0	8.4	0.8	0.7
CMT2	1.8	4.1	2.9	3.2	6.1	0.8	1.5
OSA	0.6	0.4	0.4	2.7	5.6	0	0
PF	2.1	0.8	2.9	4.8	...?	0.8	2.1
OTS	0	1	1.1	1.4	2.5	0	0.1
T	0	0	0.3	0	0	0	0
GHL	0	0.1	0.4	0.6	1.6	0	0.4
TC	0	1.8	2	4	5.6	1	3
Probl.	8	9	10	11	12	13	14
n	100	150	199	120	100	120	100
CW	11	18.5	22	3.4	1.4	5.4	1.2
MJ	13.3	9.8	20.7	5.3	6.8	2.8	1.9
AG	5.2	3.9	4.2	0.5	1.7	0.3	0.9
DV	5.3	10	10	1.5	1	1	1.8
GM	2.5	5.5	7.6	17.7	12.5	12.7	8.7
CMT1	5.4	6.6	6.9	2.2	0.9	4.1	1.1
FJ	2.2	5.5	7.6	...?	0.5	...?	0
CMT2	2.2	4.4	7	4.6	0?	3.9	1.3
OSA	0.1	0.1	1	11.4	0.8	0	2.7
PF	2.4	5.2	...?	0.7	0.8	5.2	0?
OTS	0.1	1.8	1	0	0	0.1	0?
T	0	0.1	0	2.9	0?	0.3	0?
GHL	0	1.3	1.1	0.1	0?	1.8	0?
TC	1.9	3.4	4.2	0	0.4	1.9	0.7

Tableau 6.3 : Ecart à la meilleure solution connue Z^*

D'une façon générale, TABOU_CHAINE présente des solutions assez proches des meilleures connues. Sur neuf de ces problèmes, ces solutions restent à moins de 2% de la meilleure connue Z^*, avec des cas où ce pourcentage est inférieur à 0.5% (problèmes 1, 11 et 12).

En particulier, notre algorithme trouve la solution **optimale** pour le problème 1. En plus il fournit la meilleure solution connue pour le problème 11, ça veut dire

qu'il n'y a aucun algorithme qui se révele supérieur à TABOU_CHAINE sur tous les problèmes.

Par contre, sur quelques problèmes (4, 5 et 10) les solutions sont moins bonnes, ce qui fait baisser la "qualité" moyenne des solutions.

Une évaluation de la qualitée moyenne des solutions est donnée dans le tableau 6.4. Il faut remarquer que les valeurs pour les heuristiques FJ et PF sont indiquées seulement sur 12 problèmes.

méthode	moyenne des solutions	% au dessus de Z^*		
		min	max	moyenne
CW	1095	1.21	22.04	9.59
MJ	1089.21	1.88	20.72	9.37
AG	1008.29	0.33	5.65	3.15
DV	1054.86	1.02	22.76	6.93
GM	1055.43	0.82	17.68	6.43
CMT1	1012.50	0	6.95	3.81
FJ	952.42	0	8.39	2.41
CMT2	1013.93	0	7.01	3.13
OSA	998.83	0	11.39	1.84
PF	940.42	0	5.22	2.31
OTS	985.19	0	2.52	0.65
T	980.51	0	2.88	0.26
GHL	984.48	0	1.77	0.53
TC	1002	0	5.56	2.13

Tableau 6.4 : Evaluation de la qualitée moyenne des solutions

On peut alors constater que TABOU_CHAINE se révele supérieur à tous les heuristiques "classiques" de résolution du VRP (de CW à CMT2 dans les tableaux 6.2 et 6.3).

Les meta-heuristiques comme le Recuit Simulé et la méthode Tabou nécessitent beaucoup plus de temps de calcul que les heuristiques "classiques". Mais, comme il a été montré, cet effort suplémentaire est justifié par les améliorations de la qualité des solutions obtenues.

La comparaison directe des temps d'exécution entre algorithmes entraîne plusieures difficultés, particulièrement quand le même algorithme est exécuté successivement avec différents paramètres dans un nombre de pas non spécifié, ou quand la régle pour déterminer ces paramètres est indéfinie. De même, pour comparer les temps des algorithmes qui utilisent des "semences", il faudrait les faire exécuter un

bon nombre de fois avec les mêmes paramètres et différentes semences. De plus, les temps devraient être enregistrés pour des valeurs équivalentes des solutions.

Dans le tableau 6.5, nous présentons les temps de calcul (en minutes) correspondants aux meilleures solutions obtenues par TABOU_CHAINE. Ces résultats ont été mesurés sur une station de travail Sun Sparc IPC à 25 MHz et 8 M de mémoire vive.

Problème	n	temps
1	50	28.1
2	75	129.7
3	100	42.2
4	150	91.5
5	199	98.1
6	50	13.1
7	75	31.6
8	100	65.5
9	150	38.5
10	199	76.4
11	120	80.3
12	100	32
13	120	42.5
14	100	24.5

Tableau 6.5 : Temps de CPU (en minutes) correspondantes aux meilleures solutions de TABOU_CHAINE

Chapitre 7

Conclusion

Dans ce rapport, nous avons presenté un nouvel algorithme pour la résolution du VRP. Cet algorithme est original et se différencie des autres approches de la littérature par plusieurs aspects.

Il permet en particulier de montrer que la considération de chaînes d'éjection pour créer des mouvements composés, permet de trouver une méthode efficace de résolution du VRP.

La structure d'une chaîne d'éjection a été définie par une séquence de mouvements de transition et de fermeture. Deux structures de chaînes d'éjection ont été proposées, chacune correspondant à une procédure de recherche différente.

Une chaîne d'éjection avec des mouvements de fermeture du Type I conduit à une "procédure d'échange" de sommets entre tournées. Ce type de mouvement a la particularité de garder constant le nombre de sommets en chaque tournée. Une "procédure d'insertion" est obtenue par l'utilisation de mouvements de fermeture du Type II.

De plus, plusieurs cas de non admissibilité de solutions ont été considérés lors de la construction d'une chaîne d'éjection. Ainsi, un mouvement composé admissible peut s'obtenir à partir d'une séquence de mouvements partiellement ou totalement non-admissibles. Cette approche montre qu'on peut concevoir plusieurs mouvements plus sophistiqués dans une méthode Tabou de façon à "mieux" sortir d'optima locaux.

Finalement, une autre avantage de notre algorithme est sa flexibilité. Il peut être facilement adapté à plusieurs contextes, par exemple, des contraintes comme un nombre de véhicules fixe ou limité, des véhicules avec des différentes caractéristiques, l'affectation de clients à des véhicules spécifiques,...

Bibliographie

[1] Altinkemer (K.) et Gavish (B.). – Parallel savings based heuristic for the delivery problem. *Operations Research*, vol. 39, 1991, pp. 456–469.

[2] Beasley (J. E.). – Route first-cluster second methods for vehicle routing. *Omega*, vol. 11, 1983, pp. 403–408.

[3] Bellmore (M.) et Hong (S.). – Transformation of multisalesman problem to the standard traveling salesman problem. *Journal of the Association for Computing Machinery*, vol. 21, n° 3, jul 1974, pp. 500–504.

[4] Bellmore (M.) et Nenhauser (G. L.). – The traveling salesman problem: a survey. *Operations Research*, vol. 16, 1968, pp. 538o–558.

[5] Beltrami (E.) et Bodin (L.). – Networks and vehicle routing for municipal waste collection. *Networks*, vol. 4, n° 1, 1974, pp. 65–94.

[6] Benders (J. F.). – Partitionning procedures for solving mixed variables programming problems. *Numerische Mathematik*, vol. 4, 1962, pp. 238–252.

[7] Bennett (B.) et Gazis (D.). – School bus routing by computer. *Transportation Research*, vol. 16, n° 4, 1972, pp. 317–325.

[8] Bodin (L.) et Golden (B.). – Classification in vehicle routing and scheduling. *Networks*, vol. 11, 1981, pp. 97–108.

[9] Bodin (L.), Golden (B.), Assad (A.) et Ball (M.). – Routing and scheduling of vehicles and crews: the state of the art. *Computers and Operations Research*, vol. 10, 1983, pp. 63–212.

[10] Chakrapani (J.) et Skorin-Kapov (J.). – *Connection Machine Implementation of a Tabu Search Algorithm for the Traveling Salesman Problem.* – Rapport technique n° HAR-92-10, SUNY at Stony Brook, NY 11794, Harriman School for Management and Policy, 1991.

[11] Chakrapani (J.) et Skorin-Kapov (J.). - *Massively parallel tabu search for the quadratic assignment problem.* - Rapport technique n° HAR-91-06, SUNY at Stony Brook, NY 11794, Harriman School for Management and Policy, 1991.

[12] Christofides (N.). - *The Traveling Salesman Problem. A Guide Tour of Combinatorial Optimisation*, chap. 12, pp. 431–448. - (E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, Eds), Wiley, Chichester, 1985.

[13] Christofides (N.) et Eilon (S.). - An algorithm for the vehicle-dispatching problem. *Operations Research Quarterly*, vol. 20, 1969, pp. 309–318.

[14] Christofides (N.) et Eilon (S.). - Algorithms for large scale tsp's. *Operations Research Quarterly*, vol. 23, 1972, p. 511.

[15] Christofides (N.), Mingozzi (A.) et Toth (P.). - *The Vehicle Routing Problem. Combinatorial Optimisation*, chap. 11, pp. 315–338. - (N. Christofides, A. Mingozzi, P. Toth and C. Sandi, Eds), Wiley, Chichester, 1979.

[16] Christofides (N.), Mingozzi (A.) et Toth (P.). - Exact algorithms for the vehicle routing problems, based on spanning tree ans shortest path relaxation. *Mathematical Programming*, vol. 20, 1981, pp. 255–282.

[17] Christofides (N.), Mingozzi (A.) et Toth (P.). - State space relaxation procedures for the computation of bounds to routing problems. *Networks*, vol. 11, 1981, pp. 145–164.

[18] Clark (G.) et Wright (J. W.). - Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, vol. 12, 1964, pp. 568–581.

[19] Dantzig (G.) et Ramser (J.). - The truck dispatching problem. *Operations Research*, vol. 12, 1959, pp. 81-91.

[20] Desrochers (M.). - *La Frabrication d'Horaires de Travail pour les Conducteurs de l'Autobus par une Méthode de Génération de Colonnes.* - Thèse de PhD, Dépt. d'Informatique et de Recherche Operationnelle, Université de Montréal, 1993.

[21] Desrochers (M.), Desrosiers (J.) et Solomon (M. M.). - A new optitisation algorithm for the vehicle routing problem with time windows. *Operations Research*, vol. 40, 1992, pp. 342–354.

[22] Desrochers (M.) et Verhoog (T.). - *A Matching Based Algorithme for the Vehicle Routing Problem.* - Rapport technique n° Cahier du Gerad G-89-04, Ecole de Haultes Etudes Comerciales de Montréal, 1989.

[23] Dror (M.) et Levy. – A vehicle routing improvement algorithm comparison of a greedy and a matching implementation for the inventory routing. *Computers and Operations Research*, vol. 13, 1986, pp. 33–45.

[24] Eilon (S.), Watson-Gandy (C.) et Christofides (N.). – Wiley, Chichester, 1971.

[25] F. Grefenstette, R. Gopal (B. R.) et Gucht (D. V.). – Genetic algorithms for the traveling salesman problem. Inprocedings on the third international conference on Genetics Algorithms. Arlington, pp. 160–167. – 1989.

[26] Fisher (M. L.) et Jaikumar (R.). – *A Decomposition Algorithm for Large-Scale Vehicle Routing.* – Rapport technique n° 78-11-05, Departement of Decision Sciences, University of Pennsylvania, 1978.

[27] Fisher (M. L.) et Jaikumar (R.). – A generalized assignment heuristic for the vehicle routing. *Networks*, vol. 11, 1981, pp. 109–124.

[28] Fisher (M. L.) et Kan (A. H. G. R.). – The design, analysis and implementation of heuristics. *Management Science*, vol. 34, 1988, pp. 263–265.

[29] Garvin (W.), Crandall (H.), John (J.) et Spellman (R.). – Applications of linear programming in the oil industry. *Management Science*, vol. 3, n° 4, jul 1957, pp. 407–430.

[30] Gaskell (T.). – Bases for the vehicle fleet scheduling. *Operations Research Quarterly*, vol. 18, 1967, pp. 281–295.

[31] Gendreau (M.), Hertz (A.) et Laporte (G.). – *Tabu Search Heuristics for the Vehicle Routing Problem.* – Rapport technique n° CRT-777, Centre de Recherche sur les Transports, Université de Montréal, 1990.

[32] Gendreau (M.), Hertz (A.) et Laporte (G.). – *New Insetion and Pos-Optimization Procedures for the Traveling Salesman Problem.* – Rapport technique n° CRT-708, Centre de Recherche sur les Transports, Université de Montréal, 1992.

[33] Gillet (B. E.) et Miller (L. R.). – A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, vol. 22, 1974, pp. 340–349.

[34] Glover (F.). – Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, vol. 5, 1986, pp. 533–549.

[35] Glover (F.). – Tabu search - part i. *ORSA Journal on Computing*, vol. 1, n° 3, 1989, pp. 190–206.

[36] Glover (F.). – *Multilevel Tabu Search and Embedded Search Neighborhoods for the Traveling Salesman Problem.* – Rapport technique, University of Colorado at Boulder,Graduate School of Business and Administration., jun 1991.

[37] Glover (F.), Karney (D.), Klingman (D.) et Napier (A.). – A computational study on start procedures, basis change criteria, and solution algorithms for transportation problems. *Management Science*, vol. 20, n° 5, 1974, pp. 793–813.

[38] Glover (F.) et Laguna (M.). – Tabu search. *In : Modern Heuristic Techniques for Combinatorial Problems.* University of Colorado at Boulder,Graduate School of Business and Administration. – jul 1992.

[39] Glover (F.), Taillard (E.) et de Werra (D.). – *A User's Guide to Tabu Search.* – Rapport technique n° ORWP 92101, Département de Mathématiques, Ecole Polytechnique Féderal de Lausane, Switzerland, 1992.

[40] Golden (B.). – *Vehicle Routing Problems: Formulations and Heuristic Solution Techniques.* – Rapport technique n° 113, M.I.T. Operations Research Center, august 1975.

[41] Golden (B.) et Assad (A.). – *Vehicle Routing: Methods and Studies.* – Rapport technique, North Holland, 1988.

[42] Hadjiconstantinou (E.) et Christofides (N.). – An optimal procedure for solving basic vehicle routing problems. presented at 35th Annual Conference of the Canadian Operational Research Society, Halifax, Canada. – 1993.

[43] Haimovich (M.) et Kan (A. H. G. R.). – Bounds and heuriatics for capacited routing problems. *Mathematics of Operations Research*, vol. 10, 1985, pp. 527–542.

[44] Hansen (P.). – The steepest ascend mildest descent heuristic for combinatorial programming. – 1986. Presented at the Congress on Numerical Methods in Combinatorial Optim. Capri, Italy.

[45] Hays (R.). – *The Delivery Problem.* – Rapport technique n° 106, Carnegie Institute of Technology, Departement of Management Science, 1967.

[46] Held (M.) et Karp (R.). – The traveling-salesman problem and minimum spanning trees. *Operations Research*, vol. 18, 1970, pp. 1138–1162.

[47] Hertz (A.) et de Werra (D.). – Using tabu search techniques for graph coloring. *Computing*, vol. 39, 1987, pp. 345–351.

[48] Kearney (A. T.). – *Measuring and Improving Productivity in Physical Distribution.* – Rapport technique, Oak, Brook, IL., National Council of Physical Distribution Management, 1984.

[49] Kirpatrick (S.), Gelatt (C. D. J.) et Vecchi (M. P.). – Optimisation by simulated annealing. *Science*, 1983, pp. 671–680.

[50] Laguna (M.), Kelly (J. P.), Gonzalez-Valarde (J. L.) et Glover (F.). – *Tabu Search for Multilevel Generalized Assignment Problem.* – Rapport technique, Graduate School of Business and Adiministration, University of Colorado at Boulder, 1991.

[51] Laporte (G.). – The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, vol. 59, 1992, pp. 345–358.

[52] Laporte (G.), Mercure (H.) et Nobert (Y.). – An exact algorithm for the assymetrical capacited vehicle routing problem. *Networks*, vol. 16, 1986, pp. 33–46.

[53] Laporte (G.) et Nobert (Y.). – Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics*, vol. 31, 1987, pp. 147–184.

[54] Laporte (G.), Nobert (Y.) et Desrochers (M.). – Optimal routing under capacity and distance restrictions. *Operations Research*, vol. 33, 1985, pp. 1050–1073.

[55] Lasdon (L. S.). – *Optimisation Theory for Large Systems.* – The Macmillan Company, London, 1970.

[56] Lenstra (J. K.) et Kan (A. H. G. R.). – Some simple applications of the travelling salesman problem. *Operations Research Quarterly*, vol. 26, 1975, pp. 717–734.

[57] Lenstra (J. K.) et Kan (A. H. G. R.). – A complexity of vehicle routing and scheduling problems. *Networks*, vol. 11, 1981, pp. 221–227.

[58] Lin (S.). – Computer solutions of the tsp. *Bell Systems Technical Journal*, vol. 14, 1965, p. 2245.

[59] Lin (S.) et Kernighan (B.). – An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, vol. 21, 1973, pp. 498–516.

[60] Little (J. D. C.), Murty (K.), Sweeney (D.) et Kare (C.). – An algorithm for the traveling salesman problem. *Operations Research*, vol. 11, n° 6, 1963, pp. 972–989.

[61] Magnanti (T. L.). – Combinatorial optimization and vehicle fleet planning: Prespectives and prospects. *Networks*, vol. 11, 1981, pp. 179–213.

[62] Martello (S.) et Toth (P.). – *Knapsack Problems. Algorithms and Computer Implementation*, pp. 189–220. – (S. Martello, P. Toth), Wiley, Chichester, 1990.

[63] Miller (C. E.), Tucker (A. W.) et Zemlin (R. A.). – Integer programming formulations and traveling salesman problems. *Journal of the Association for Computing Machinery*, vol. 7, 1960, pp. 326–329.

[64] Minoux (M.). – *Problèmes de Grandes Dimensions in Programation Mathématique, Tome 2.* – Dunod, Paris, 1983.

[65] Mole (R. H.) et Jameson (S. R.). – A sequential route-building algorithm employing a generalised savings criterion. *Operational Research Quarterly*, vol. 27, 1976, pp. 503–511.

[66] Mulvey (J.). – Pivot strategies for primal simplex network codes. *Journal of the Association for Computing Machinery*, vol. 25, n° 2, 1978, pp. 266–270.

[67] Nelson (M. D.), Nygard (K. E.), Griffin (J. H.) et Shreve (W. E.). – Implementation techniques for the vehicle routing problem. *Computers and Operations Research*, vol. 12, 1985, pp. 273–283.

[68] Orloff (C.). – Routing a fleet of m vehicles to/from a central facility. *Networks*, vol. 4, n° 2, 1974, pp. 147–162.

[69] Osman (I. H.). – Metrastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research. Forthcoming*, 1993.

[70] P. Jog (J. S.) et Gucht (D. V.). – The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem. Inproceedings on the third international conference on Genetics Algorithms. Arlington, pp. 110–115. – 1989.

[71] Paessens (H.). – The savings algorithm for the vehicle routing problem. *European Journal and Operations Research*, vol. 34, 1988, pp. 336–344.

[72] Parker (G.), Deane (R.) et Holmes (R.). – On the use of a vehicle routing algorithm for the parallel processor problem with sequence dependent changeove cost. *AIIE Transations*, vol. 9, 1977, pp. 115–160.

[73] Psaraftis (H.). – K-interchange procedures for the local search in a precedence-constrained routing. *European Journal of Operational Research*, vol. 13, 1983, pp. 391–402.

[74] Pureza (V. M.) et França (P. M.). – *Vehicle Routing Problems via Tabu Search Methaheuristic.* – Rapport technique n° CRT-747, Centre de Recherche sur les Transports, Université de Montréal, 1991.

[75] Russell (R. A.). – An effective heuristic for the m-tour traveling salesman problem with some side constraintes. *Operations Research*, vol. 25, 1977, pp. 517–524.

[76] Sanso (B.), Soumis (F.) et Gendreau (M.). – Routing models and applications in telecommunications networks. – 1988. Présenté au congrés EURO IX TIMS XXVIII, Paris.

[77] Semmet (F.) et Taillard (E.). – Solving real-life vehicle routing problems efficiently using taboo search. *Annals of Operations Research. Forthcoming*, 1992.

[78] Skorin-Kapov (J.). – Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, vol. 2, 1990, pp. 33–45.

[79] Sveska (J.) et Huckfeldt (V.). – Computational experience with an m-salesman traveling salesman algorithm. *Management Science*, vol. 19, n° 7, 1973, pp. 790–799.

[80] Taillard (E.). – Robust tabu search for the quadratic assignment problem. *Parallel Computing*, vol. 17, 1991, pp. 443–455.

[81] Taillard (E.). – *Parallel Iterative Search Methods for the Vehicle Routing Problems.* – Rapport technique n° ORWP 92/03, Ecole Polytechnique Féderale de Lausane, Switzerland, 1992.

[82] Toth (P.). – Heuristic algorithms for the vehicle routing. – 1984. Presented at the Workshop on Routing Problems, Hamburg.

[83] Yellow (P.). – A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, vol. 21, 1970, pp. 281–283.

Annexe A

Les meilleures solutions obtenues avec TABOU_CHAINE (distances réelles)

Dans la colone "Temps" n'est montré que les temps de voyages. Pour obtenir les la durée totale de la tournée il faut additioner les temps de services dans les cas où ils sont appliqués. Chaque ligne représente la sequence de clients à être visités en chaque tournée.

Tournée											n'	D=160 Charge	Temps
6	14	25	24	43	7	23	48	27			9	152	98.45
8	26	31	28	3	36	35	20	22	1	32	11	149	118.52
38	9	30	34	50	16	21	29	2	11		10	159	99.33
46	5	49	10	39	33	45	15	44	37	12	11	160	99.25
47	4	17	42	19	40	41	13	18			9	157	109.06
											50		524.61

Tableau A.1 : Solution du problème 1

Tournée										n'	D=140 Charge	Temps
2	62	22	28	61	21	74				7	138	89.34
12	58	11	53	35	7	26				7	139	74.12
19	14	59	66	65	38					6	140	97.51
29	5	37	20	70	60	71	69	36	47	10	136	103.47
40	39	9	32	44	3					6	134	63.22
46	8	54	13	57	15	48	30			8	135	85.13
49	24	18	50	25	55	31	10	72		9	140	135.48
67	34	52	27	45	4					6	136	39.91
73	1	43	42	64	41	56	23			8	128	106.39
75	68	6	33	63	16	51	17			8	138	56.35
										75		850.92

Tableau A.2 : Solution du problème 2

Tournée														n'	D=200 Charge	Temps
13	97	87	42	44	38	14	43	15	57	2	58			12	167	113.32
26	4	39	67	25	55	54	24	29	68	80	12			12	191	120.82
52	7	82	48	47	36	49	64	11	19	62	88			12	194	122.57
53	40	73	74	22	41	23	56	75	72	21				11	152	84.44
69	31	10	63	90	32	20	51	9	81	33	50	28		13	186	106.25
76	77	3	79	78	34	35	71	65	66	30	70	1	27	14	200	122.36
89	60	5	84	61	16	86	17	45	46	8	83	18		13	169	114.56
94	95	92	37	98	100	91	85	93	59	99	96	6		13	199	58.26
														100		842.58

Tableau A.3 : Solution du problème 3

Tournée	n'	D=200 Charge	Temps
5 125 106 73 117 89 39 75 105 54 10 76	12	193	110.64
11 100 2 83 131 20 59 101 51 32	10	175	69.00
12 144 37 52 15 108 103	7	114	46.70
17 92 65 93 42 64 19 94 88 40 136 41 66 111	14	200	124.35
23 69 7 61 114 99 43 86 97 24 96 58 14 68	14	199	128.44
38 62 9 130 50 118 29 129 53 127 16 126 78	13	197	63.30
46 102 6 57 132 98 112 48 138 81 27 77	12	189	75.40
47 55 134 13 67 95 25 133 110 18 139	11	197	77.84
49 104 30 34 74 79 21 128 84 35 85 36 115 121 3	15	200	134.51
56 146 149 4 109 143 135 141 150 87 148 142 147 145	14	193	60.48
63 137 44 107 45 91 72 33 124 122 123 71 90	13	190	73.18
119 1 120 60 8 26 113 140 82 31 116 28 70 80 22	15	188	108.35
	150		1072.19

Tableau A.4 : Solution du problème 4

Tournée	n'	D=200 Charge	Temps
9 177 130 50 118 21 79 74 163 34 104 30 179	13	188	79.67
17 147 151 142 148 87 150 193 141 135 143 109 149 4	14	198	59.77
45 91 72 33 171 124 122 197 155 54 10 76	12	198	79.92
46 77 167 102 27 162 1 119 32 51 176 11	12	153	60.59
56 146 145 137 44 172 92 42 93 65 107 178 15 52 37	15	183	88.72
68 133 199 14 58 174 25 95 67 13 173 134 55	13	193	87.50
78 154 38 184 62 195 49 198 71 180 90	11	200	61.53
83 131 20 59 116 188 70 28 160 80	10	200	96.21
96 24 168 97 86 43 99 114 181 113 26 140 82 31 22 120	16	198	151.75
100 2 157 185 129 158 53 127 16 196 126	11	189	56.78
101 3 161 121 115 36 85 35 164 84 128 169 29	13	198	114.14
123 125 106 186 73 117 89 39 187 170 75 105 165	13	196	111.79
132 98 23 69 7 175 61 159 112 48	10	195	81.71
139 166 110 18 47 183 144 63	8	180	48.45
153 6 194 57 182 138 189 60 8 81 190	11	200	69.40
156 12 152 108 5 103	6	127	26.17
191 41 94 19 192 64 88 40 136 66 111	11	190	103.35
	199		1377.46

Tableau A.5 : Solution du problème 5

Tournée										n'	D=160 Charge	T=200 t=10 Temps
2	29	20	35	36	3	32				7	115	98.16
5	49	10	39	33	45	15	44	37	12	10	155	99.12
6	23	7	43	24	25	14				7	120	91.62
18	13	41	40	19	42	17	4	47		9	157	109.04
27	48	8	26	31	28	22	1			8	102	80.41
46	38	9	30	34	21	50	16	11		9	128	82.89
										50		561.24

Tableau A.6 : Solution du problème 6

Tournée								n'	D=140 Charge	T=160 t=10 Temps
4	29	5	36	47	48	75		7	135	68.00
11	53	19	8	34	67			6	139	73.00
16	49	24	18	50	44	3		7	114	87.00
17	32	55	25	9	39	12		7	130	88.00
35	14	59	66	65	38			6	135	93.00
40	72	31	10	58	7	26		7	139	90.00
46	52	27	13	54	57	15	45	8	134	80.00
51	63	23	56	41	43	73		7	94	86.00
62	22	64	42	1	33	6		7	133	90.00
68	2	28	61	69	21	30		7	138	86.00
74	71	60	70	20	37			6	73	97.00
								75		938

Tableau A.7 : Solution du problème 7

Tournée												n'	D=200 Charge	T=230 t=10 Temps
1	51	66	71	65	35	9	81	33	50			10	154	116.00
6	96	85	91	44	14	38	86	16	61	99		11	186	97.00
7	19	11	63	64	49	36	47	48	82			10	167	124.00
12	68	80	24	29	34	78	79	3	77	76	28	12	169	90.00
18	83	8	46	45	17	84	5	60	89			10	102	90.00
26	54	55	25	4	39	67	23	56	72	21		11	189	112.00
40	73	74	75	22	41	57	15	43	42	87	13	12	143	95.00
52	88	62	10	90	32	20	30	70	31	69	27	12	163	93.00
53	58	2	97	92	37	100	98	93	59	95	94	12	185	66.00
												100		883

Tableau A.8 : Solution du problème 8

Tournée												n'	D=200 Charge	T=200 t=10 Temps
1	120	80	28	116	70	22	51	32				9	111	83.00
5	10	54	75	105	30	104	9	62	38	78		11	151	71.00
16	127	21	79	74	34	130	50	118				9	127	71.00
27	48	7	114	99	69	23	57	6				9	143	105.00
37	44	107	65	93	42	92	137	147	17	63	12	12	158	79.00
46	102	77										3	30	25.00
47	134	67	13	136	40	88	64	109	144			10	190	99.00
71	122	125	106	73	117	89	39	49	76			10	194	95.00
108	52	15	45	91	72	33	124	123	90	103		11	131	62.00
119	101	3	121	115	36	85	35	59				9	114	105.00
126	53	129	29	128	84	20	131	83	2	100	11	12	199	79.00
132	98	24	97	43	86	96	58	14	68			10	148	98.00
135	111	66	41	94	19	141	150	148	87	142	145	12	185	74.00
138	112	61	26	113	140	82	31	8	60	81		11	160	88.00
139	18	110	133	95	25	55	143	149	4	146	56	12	194	70.00
												150		1204

Tableau A.9 : Solution du problème 9

Tournée													n'	D=200 Charge	$T = 200$ $t = 10$ Temps
3	161	121	115	36	85	35	20	59					9	163	107.00
11	176	51	32	119	162	81	190	27	167	77			11	192	48.00
12	63	183	56	47	139	166	68	153	6	102	46		12	193	58.00
22	101	70	188	116	28	160	80	120	1				10	149	90.00
64	88	40	136	13	67	173	134	55					9	142	102.00
76	49	198	10	122	123	71	180	90	152	108			11	180	59.00
78	126	196	185	129	29	53	158	127	16	184	62		12	191	61.00
98	24	168	97	86	43	99	114						8	141	114.00
100	2	157	83	131	164	84	128	169	79	21			11	175	85.00
132	14	58	174	96	95	25	199	133	110	18			11	178	83.00
137	44	172	92	42	93	65	107	178	15	52	37		12	149	79.00
138	26	113	181	31	82	140	8	189	60				10	143	91.00
144	146	149	4	109	143	148	87	151	142	147	145	17	13	179	52.00
150	193	141	192	19	94	191	41	66	111	135			11	189	75.00
154	38	5	103	156									5	99	27.00
155	54	39	187	89	117	170	75	105	165				10	162	94.00
171	124	197	106	186	73	125	33	72	91	45			11	176	86.00
194	57	182	23	69	7	175	61	159	112	48			11	193	78.00
195	179	30	104	34	163	74	130	50	118	177	9		12	192	75.00
													199		1464

Tableau A.10 : Solution du problème 10

Tournée																n'	D=200 Charge	Temps
2	1	3	4	5	6	7	9	10	11	15	14	13	12	8	118	16	199	133.00
53	55	58	56	60	63	66	64	62	61	65	59	57	54	52	100	16	199	213.00
67	69	70	71	74	72	75	78	80	79	77	76	73	68	98		15	200	145.00
87	92	89	91	90	114	18	108	83	113	117	84	85	112	81	119	16	191	63.00
88	82	111	86	95	93	96	94	97	115	116	103	104	107	99	101			
102	106	105	120													20	191	78.00
109	21	20	23	26	28	29	32	35	36	34	33	30	31	27	24			
22	25	19	16	17												21	197	208.00
110	40	43	45	48	51	50	49	46	47	44	41	42	39	38	37	16	198	199.00
																120		1039

Tableau A.11 : Solution du problème 11

Tournée														n'	D=200 Charge	Temps
5	7	8	11	9	6	4	2	1	75					10	160	55.00
13	17	18	19	15	16	14	12	10						9	200	96.00
23	26	28	30	29	27	25	24	22	21	20				11	170	52.00
34	36	39	38	37	35	31	33	32						9	200	97.00
43	42	41	40	44	45	46	48	51	50	52	49	47		13	160	65.00
59	60	58	56	53	54	57	55							8	200	101.00
66	68	64	61	72	80	79	77	73	70	71	76	78	81	14	200	137.00
67	65	63	74	62	69									6	150	45.00
90	87	86	83	82	84	85	88	89	91					10	170	77.00
98	96	95	94	92	93	97	100	99	3					10	200	98.00
														100		823

Tableau A.12 : Solution du problème 12

Tournée														n'	D=200 Charge	T=720 t=50 Temps
2	1	3	4	5	37	38	39	42	40					10	140	216.00
6	7	9	10	11	15	14	13	12	8	108	118			12	146	117.00
21	20	23	26	29	28	25	19	16	17					10	117	188.00
32	35	36	34	33	30	31	27	24	22					10	67	182.00
43	45	48	51	50	49	46	47	44	41					10	116	189.00
59	65	61	62	64	66	63	60	58	55					10	138	205.00
67	69	70	71	74	75	72	73	68	98	110				11	154	134.00
76	77	78	79	80	56	53	57	54	52					10	107	195.00
96	93	94	97	115	109	114	18	90	91	89	92	87		13	166	56.00
119	81	117	83	113	84	112	85	86	111	82	88			12	116	45.00
120	105	106	107	104	103	116	100	99	101	102	95			12	108	49.00
														120		1576

Tableau A.13 : Solution du problème 13

Tournée										n'	D=200 Charge	T=1040 t=90 Temps
3	99	100	97	93	92	94	95	96	98	10	200	98.00
5	7	10	8	9	6	4	2	1	75	10	160	57.00
13	17	18	19	15	16	14	12	11		9	200	97.00
23	26	28	30	29	27	25	24	22	21	10	160	50.00
34	36	39	38	37	35	31	33	32		9	200	97.00
41	40	44	45	48	51	50	52	49	20	10	100	64.00
47	46	42	43							4	70	44.00
59	60	58	56	53	54	57	55			8	200	101.00
65	63	62	74	72	61	64	68	66	69	10	190	59.00
67	80	79	77	73	70	71	76	78	81	10	160	128.00
90	87	86	83	82	84	85	88	89	91	10	170	77.00
										100		872

Tableau A.14 : Solution du problème 14

Annexe B

Graphiques

Dans les graphiques ci-dessous, on répresente les problèmes sur l'axe des abcisses et l'ecart à la meilleure solution connue sur l'axe des ordonnés.

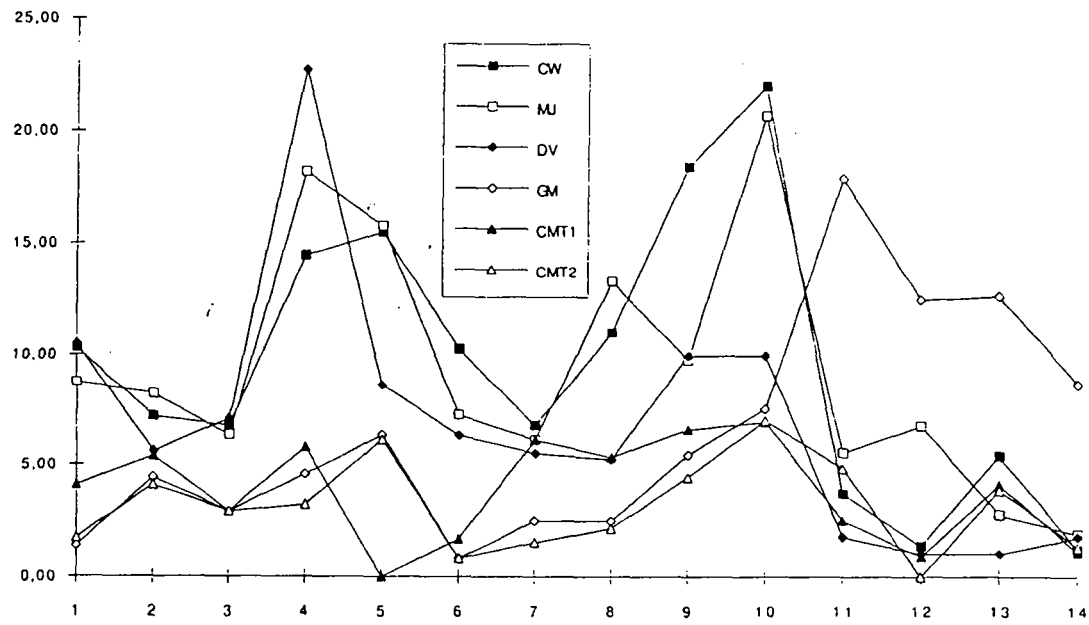


Figure B.1 : Comparaison de GM avec d'autres heuristiques "classiques" suivant la structure du problème.

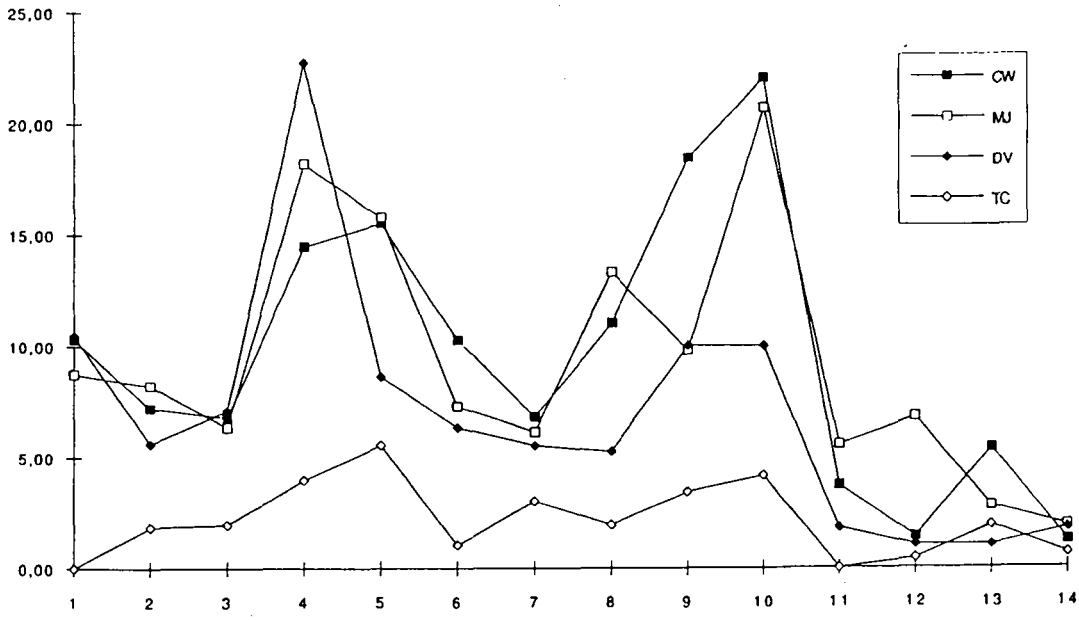


Figure B.2 : Les meilleures résultats de TABOU_CHAINE pour les premières 12 problèmes.

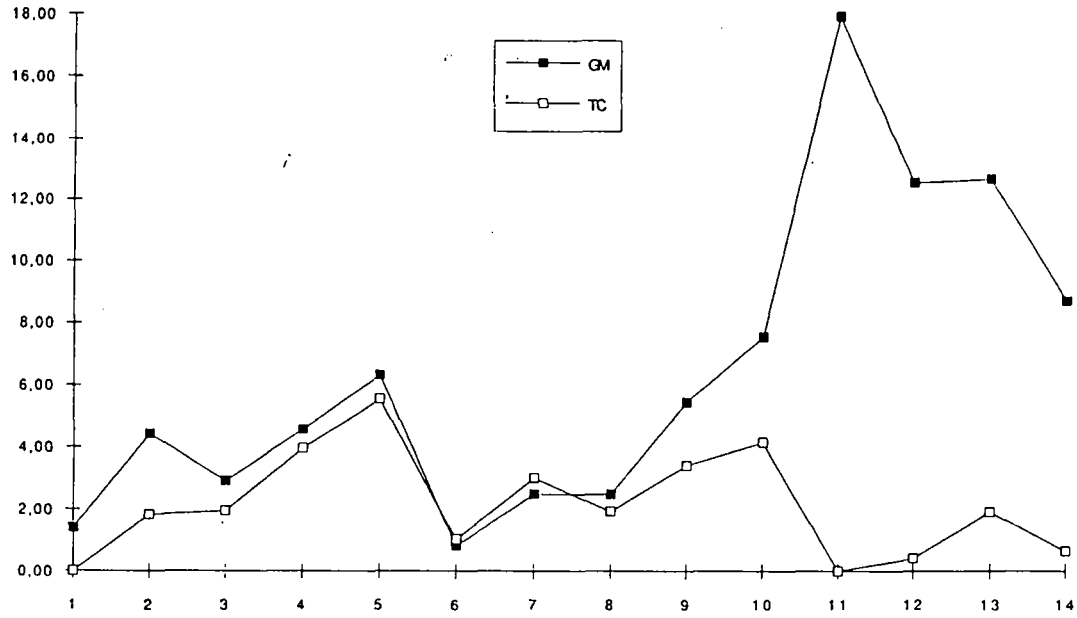


Figure B.3 : TABOU_CHAINE donne les meilleurs résultats pour les problèmes structurés.

Dans les graphiques suivants sur l'axe des abcisses sont répresentés les differentes heuristiques. Sur l'axe des ordonnés est répresente l'ecart à la valeur de la solution donné par TABOU_CHAINE.

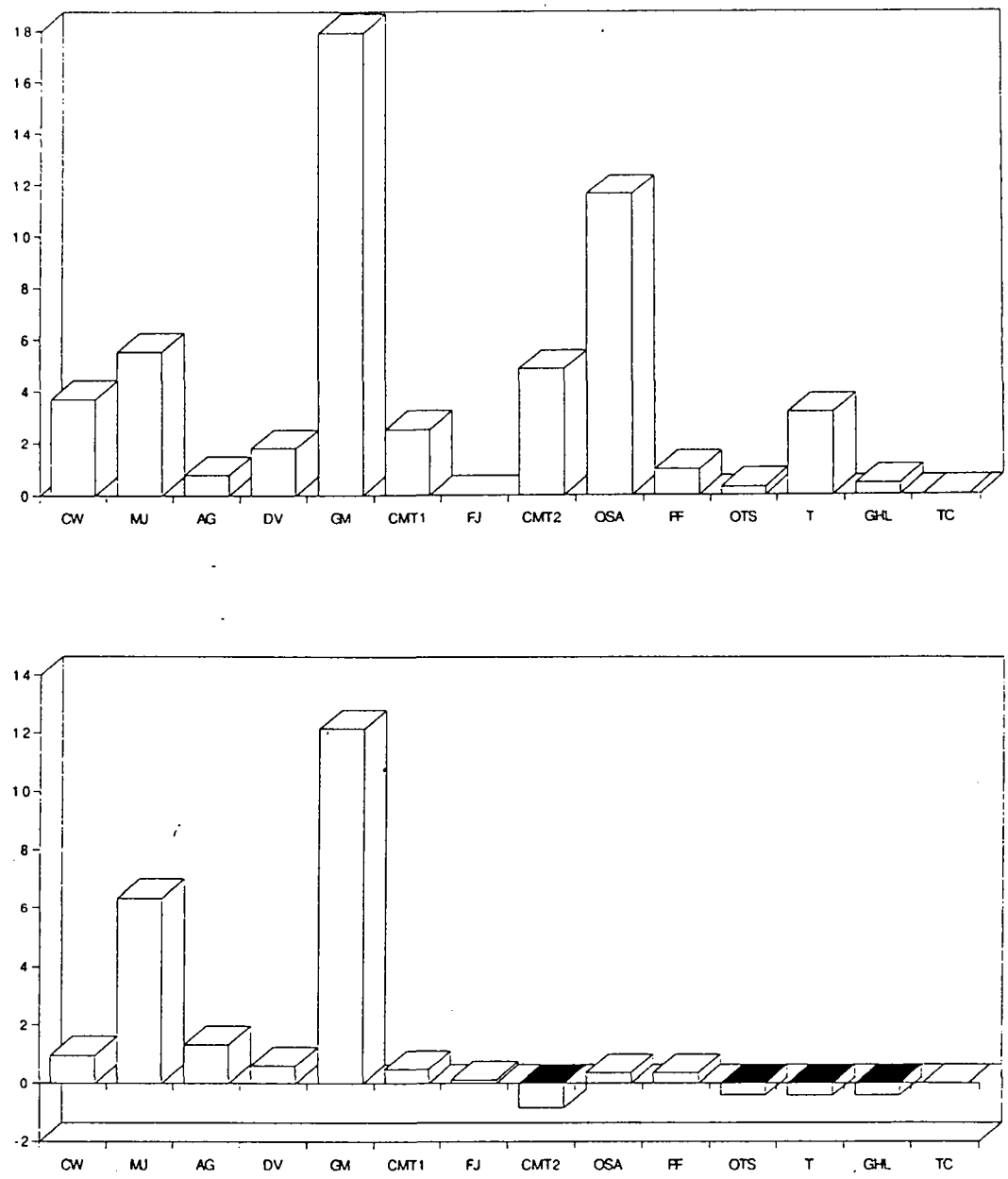


Figure B.4 : Exemples de compétitivité entre TABOU_CHAINE et d'autres algorithmes de Recuit Simulé et Tabou
(resp. le cas des problèmes 11 et 12).

Table des figures

Table des tableaux

Unité de Recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399