PROBLÈMES DE CHEMINEMENTS OPTIMAUX DANS LES RÉSEAUX AVEC CONTRAINTES ASSOCIÉES AUX ARCS

THÈSE N° 2103 (2000)

PRÉSENTÉE AU DÉPARTEMENT DE MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES TECHNIQUES

PAR

Michel MITTAZ

Ingénieur mathématicien diplômé EPF
de nationalité suisse et originaire de Chermignon (VS)

acceptée sur proposition du jury:

Prof. A. Hertz, directeur de thèse
Prof. L. Gambardella, rapporteur
Prof. G. Laporte, rapporteur
Prof. F.-L. Perret, rapporteur
Prof. D. de Werra, rapporteur

Lausanne, EPFL
1999

Remerciements

La réalisation d'un travail de thèse n'est pas une tâche qui s'effectue en solitaire. C'est une ascension que se fait en cordée et chaque membre y joue un rôle important.

Pour commencer il y a les guides, les professeurs Dominique de Werra et Alain Hertz. Ils m'ont fait découvrir et apprécier la recherche opérationnelle, m'ont fait partager leur enthousiasme et m'ont toujours prodigué des conseils avisés. A leur nom il faut ajouter celui du professeur Gilbert Laporte avec qui j'ai eu la chance de pouvoir collaborer et ceux des membres du jury qui se sont penchés sur mon travail et l'ont évalué.

Il y a mes collègues de la Rose, embarqués dans la même ascension, ou l'ayant déjà accomplie. Ils ont contribué à l'excellente ambiance qui régna au sein du groupe, et se sont toujours montrés disponibles lorsque je les sollicitais.

Il y a toutes les personnes qui, lorsque mon sac devenait un peu lourd, n'ont pas hésité à prendre avec elles une partie de son contenu. Je pense à Ariane, Anne-Chantal et à mon papa qui ont consacré de longues heures à la relecture de ce travail et dont les remarques pertinentes ont grandement contribué à en améliorer la qualité. Je pense également aux membres de ma famille grâce au soutien desquels cette ascension m'est apparue plus facile.

Je remercie du fond du coeur tous les membres de cette cordée.

Résumé

Dans cette thèse, nous considérons deux problèmes de cheminements optimaux dans les réseaux. Ces deux problèmes sont NP-durs et sont fréquemment utilisés pour modéliser des problèmes pratiques de taille importante.

Nous traitons tout d'abord le problème de tournées sur les arcs avec capacité (PTAC). Ce problème consiste à servir un ensemble de clients, représentés par les arcs d'un réseau, à l'aide de véhicules de capacité limitée. Le but est de minimiser la longueur totale des tournées accomplies par les véhicules. Nous présentons deux heuristiques basées sur des méthodes de recherche locale que nous avons développées pour résoudre le PTAC. La première est une méthode tabou. Elle fait appel à une procédure de post-optimisation originale. En se basant sur les principes utilisés dans cette procédure, nous avons défini une classe de voisinages que nous employons dans une méthode de descente à voisinage variable. Nos heuristiques se comparent favorablement à d'autres types de méthodes. Elles ont été initialement élaborées pour traiter le cas non orienté du PTAC. Nous décrivons comment les modifier pour les appliquer au PTAC orienté. Nous proposons également une adaptation au cas orienté d'une borne inférieure développée à l'origine pour le PTAC non orienté. Cette borne permet d'évaluer la qualité des solutions produites par nos algorithmes dans le cas orienté du PTAC. Ces derniers fournissent des solutions dont la valeur est, en moyenne, proche de celle de bornes inférieures.
Un aspect plus pratique du PTAC est abordé en étudiant un problème réel de confection d'horaires de bateaux. Nous décrivons ce problème en termes de PTAC avec fenêtres de temps. Le processus de modélisation est présenté en détail et les premiers résultats obtenus sont analysés. Nous avons trouvé des solutions nécessitant moins de bateaux que dans l'horaire actuel. Ces solutions ne sont cependant pas encore totalement utilisables en pratique. Quelques pistes sont suggérées pour les améliorer.

Le deuxième problème abordé dans cette thèse est le problème de plus courts chemins avec capacité (PPCCC). Une méthode tabou pour la résolution du PPCCC est présentée ainsi que des procédures de post-optimisation. Ces procédures sont utilisées dans notre méthode tabou. Nous avons considéré deux variantes de cette dernière. La première utilise les procédures de post-optimisation de manière intensive. La seconde, plus rapide, les employe moins fréquemment. Comparées à une approche gloutonne, les deux variantes obtiennent des solutions qui, en moyenne, sont de meilleure qualité.
Le PPCCC nous a été initialement proposé par EDF (Electricité de France) dans le but d'optimiser les itinéraires empruntés par des câbles dans une centrale nucléaire. Il peut aussi être vu comme un sous-problème du "Bandwidth Packing Problem" (BWP) apparaissant dans le domaine des télécommunications.

Abstract

In this thesis we consider two optimal routing problems in networks. Both are NP-hard and are often used to modelize real life problems of large size.

We first present the capacitated arc routing problem (CARP). In this problem, we have to serve a set of customers represented by the arcs of a network with vehicles of restricted capacity. The aim is to design a set of vehicle routes of least total cost. We describe two heuristic methods for the CARP based on local search techniques. The first one is a Tabu Search heuristic using an original post-optimization procedure. Principles appearing in this procedure are used to define a class of neighborhoods we combine in a Variable Neighborhood Descent method. Our algorithms compare favorably with some other methods. They were initially developed to deal with the undirected CARP. We describe how to transform them so that they can be applied to the directed CARP. We also propose an adaptation to the directed case of a lower bound originally developed for the undirected CARP. This lower bound is used to evaluate the quality of the solutions produced by our algorithms for the directed CARP. Our methods obtain solutions whose value is, on average, close to known lower bounds.
We use the CARP to tackle a real life problem of boat routing and scheduling. This problem can be viewed as a directed CARP with time windows. We describe in detail its modeling process and we analyze our first results. We obtain solutions that require less boats than the present schedule. Nevertheless, these solutions are not completely feasible in practice. We suggest some ways to improve them.

The second problem treated in this thesis is the shortest capacitated paths problem (SCPP). We present a Tabu Search heuristic and some post-optimization procedures for the SCPP. These procedures are called in our Tabu Search algorithm. Two versions of this algorithm are considered. In the first one, post-optimization procedures are intensively used. In the second one, they are less frequently applied. This second version is faster. Compared with a greedy approach, the two versions of our Tabu Search heuristic obtain solutions which are, on average, of better quality.
The SCPP was proposed to us by EDF (Electricité de France) in order to minimize the total length of the paths followed by cables in a nuclear power-station. It can also be viewed as a subproblem of the bandwidth packing problem (BWP) appearing in telecommunications.

Table des matières

Introduction générale

Les problèmes liés au transport et à l'acheminement de biens apparaissent dans tous les secteurs de l'économie tant privée que publique. La notion de biens doit être prise ici dans un sens très large. Ce terme comprend certes des marchandises ou des personnes qu'il faut transporter, mais également des services qui doivent être assurés auprès de clients, des travaux d'entretien qui doivent être réalisés sur certaines infrastructures, des informations qu'il faut transmettre, de l'énergie qu'il s'agit de distribuer, etc. Le développement des méthodes de production en flux tendu, l'extension rapide des réseaux informatiques et de téléphonie mobile, font que ces problèmes occupent une place de plus en plus importante. Dans un monde où le souci de rentabilité est sans cesse croissant, une bonne gestion des problèmes liés au transport de biens permet aux entreprises et aux collectivités publiques de réaliser des économies appréciables. Ces problèmes jouent également un rôle important dans le domaine des mathématiques appliquées et dans celui de la recherche opérationnelle en particulier. Ils ont en effet inspiré de nombreux chercheurs et ont ainsi contribué au développement d'outils et de modèles mathématiques dont le champ d'application dépasse souvent le cadre initial des problèmes à partir desquels ils ont été élaborés.

Deux types de problèmes sont souvent distingués dans le domaine de l'optimisation combinatoire. Il y a d'une part les problèmes qui peuvent être résolus de manière exacte. Il s'agit de problèmes pour lesquels il existe des algorithmes capables de trouver une solution optimale en des temps de calcul qui n'explosent pas lorsque la taille du problème augmente. Pour la deuxième catégorie de problèmes, nous ne disposons pas de telles méthodes de résolution. La structure ainsi que le nombre important de solutions qui existent pour ce type de problèmes font qu'il est impossible, à l'heure actuelle, de déterminer de façon certaine et systématique la solution optimale de ces derniers dès que leur taille devient un peu importante. Pour traiter de tels problèmes, il faut recourir à des méthodes appelées heuristiques. Une heuristique est un algorithme qui a pour but de déterminer une approximation aussi bonne que possible de l'optimum d'un problème. Il en existe de toutes sortes. Au cours des années, des heuristiques de plus en plus performantes ont été mises au point. Les plus récentes sont très générales et peuvent de ce fait s'appliquer à de très nombreux problèmes. La difficulté consiste alors à développer des ingrédients qui permettent de les adapter efficacement au problème particulier qui est traité.

Dans le cadre de cette thèse, nous allons nous intéresser à deux problèmes liés au transport de biens. Ces problèmes ont ceci de commun qu'ils consistent tous deux à déterminer des itinéraires de longueur totale minimum par lesquels doivent transiter les biens transportés. Dans les deux cas, un certain nombre de contraintes doivent être satisfaites parmi lesquelles des contraintes de capacité. Pour des tailles importantes, il n'existe pas de mé-

thode exacte pour résoudre ces problèmes. L'utilisation de méthodes heuristiques devient alors inévitable. Nous verrons dans ce travail comment nous avons adapté à ces deux problèmes des heuristiques basées sur des méthodes de recherche locale.

Le premier problème que nous allons aborder dans cette thèse est un problème de tournées de véhicules, où les clients qu'il faut servir sont représentés par les arcs (ou les arêtes) d'un réseau. Une telle modélisation est très souvent utilisée pour traiter des problèmes de distribution de courrier [Roy89], d'entretien de réseaux routiers [Egl94], de ramassage d'ordures [Bel74]. Elle est également employée pour résoudre certains problèmes industriels comme des problèmes de découpe de pièces [Man84] et des problèmes de placement d'éléments électroniques sur des circuits imprimés [Bal88]. Les clients correspondent alors généralement à des routes qu'il faut entretenir, des rues qu'il faut desservir, des trajets qui doivent obligatoirement être effectués par des véhicules, des mouvements imposés que doit accomplir un bras robotisé, etc. Nous allons étudier le cas où une demande est associée à chaque client et où les véhicules chargés d'approvisionner les clients sont de capacité limitée. Ce problème est appelé problème de tournées sur les arcs avec capacité (PTAC). Il appartient à une famille de problèmes nommés problèmes de tournées sur les arcs (PTA) parce que les clients sont modélisés à l'aide des arcs d'un réseau.
Nous présenterons une revue des différents problèmes de tournées sur les arcs, et les principales techniques utilisées pour les résoudre. Par la suite, nous nous attacherons plus particulièrement au PTAC qui généralise bon nombre de problèmes de tournées sur les arcs. Dans un premier temps, nous aborderons ce problème sous un angle plutôt théorique en présentant les algorithmes que nous avons développés ainsi que d'autres outils liés à sa résolution. Un aspect plus pratique sera ensuite traité. Nous verrons comment le PTAC peut servir de base à la modélisation d'un problème réel de confection d'horaires de bateaux.

Le deuxième problème dont il est question dans ce travail consiste à déterminer les chemins que doivent emprunter des câbles afin de relier les divers équipements d'une centrale nucléaire. Le réseau dans lequel doivent passer les câbles est composé d'éléments dont le diamètre limite le nombre et le type de câbles qui peuvent les traverser. Ce problème nous a été proposé par Electricité de France (EDF). Il entre dans le cadre très général des problèmes liés au transport de biens que nous avons décrits au début de cette introduction. Dans ce cas précis, un bien peut être vu comme une extrémité d'un câble qui est transportée jusqu'à l'équipement qu'elle doit connecter. Nous traiterons brièvement de la complexité de ce problème et présenterons un certain nombre d'outils que nous avons développés pour le résoudre. Nous verrons qu'il peut également être utilisé pour traiter un problème apparaissant dans le domaine des télécommunications.

Il n'est pas toujours aisé de décrire des concepts mathématiques de manière précise sans qu'apparaissent parfois certaines lourdeurs de style. Nous espérons que cela ne nuira pas trop à la clarté de ce travail et que les lecteurs qui s'y plongeront auront néanmoins du plaisir à le parcourir.

Définitions et notations

Nous allons présenter quelques définitions ayant trait à la théorie des graphes et qui seront utilisées tout au long de ce travail. Quelques notations seront également introduites.

Nous utiliserons généralement la notation $G = (V, E \cup A)$ pour désigner le graphe dans lequel seront définis les problèmes que nous allons traiter. L'ensemble $V = \{v_1, v_2, ..., v_n\}$ représente les sommets de G (ces sommets seront parfois numérotés de 0 à n). E est l'ensemble des arêtes du graphe et A représente l'ensemble des arcs. Un arc ou une arête reliant le sommet v_i au sommet v_j sera noté (v_i, v_j). Nous préciserons chaque fois qu'il sera possible de le faire si cette notation correspond à un arc ou à une arête. Lorsque des coûts seront associés aux arcs et aux arêtes de G, nous noterons c_{ij} le coût de l'arc ou de l'arête (v_i, v_j) de $E \cup A$. Pour désigner le coût d'un arc ou d'une arête, nous utiliserons également la notion de longueur. Ces deux termes seront employés de façon équivalente. Sauf précision contraire, tous les coûts associés aux arcs et aux arêtes de G seront supposés positifs ou nuls.

Un graphe $G = (V, E \cup A)$ est dit **non orienté** lorsque $A = \emptyset$. On utilisera la notation $G = (V, E)$ pour représenter un graphe non orienté.

Un graphe $G = (V, E \cup A)$ est dit **orienté** lorsque $E = \emptyset$. Un graphe orienté sera noté $G = (V, A)$.

Un graphe $G = (V, E \cup A)$ est dit **mixte** lorsque $E \neq \emptyset$ et $A \neq \emptyset$.

Une **chaîne** est une suite de sommets: $(v_{i_1}, v_{i_2}, ..., v_{i_t})$ telle que $(v_{i_j}, v_{i_{j+1}})$ appartient à $E \cup A$ ou $(v_{i_{j+1}}, v_{i_j})$ appartient à $E \cup A$, $1 \leq j \leq t-1$.

Un **chemin** est une chaîne particulière: $(v_{i_1}, v_{i_2}, ..., v_{i_t})$ où tous les couples $(v_{i_j}, v_{i_{j+1}})$ appartiennent à $E \cup A$, $1 \leq j \leq t-1$.

Un **cycle** est une chaîne $(v_{i_1}, v_{i_2}, ..., v_{i_t})$ où v_{i_1} est égal à v_{i_t}.

Un **circuit** est un chemin $(v_{i_1}, v_{i_2}, ..., v_{i_t})$ où v_{i_1} est égal à v_{i_t}.

Un **cycle simple** (resp. circuit simple) est un cycle (resp. circuit) ne contenant pas deux fois le même arc ou la même arête.

Un graphe $G = (V, E \cup A)$ est **connexe** si pour toute paire de sommets v_i, v_j de V il existe une chaîne reliant v_i à v_j.

Un graphe $G = (V, E \cup A)$ (où $A \neq \emptyset$) est **fortement connexe** si pour toute paire de sommets de v_i, v_j de V il existe un chemin reliant v_i à v_j et un chemin reliant v_j à v_i.

Un **arbre** est un graphe non orienté connexe sans cycle.

Un **arborescence** est un graphe orienté, connexe, sans cycle et possédant au moins une **racine**, c'est-à-dire un sommet à partir duquel tous les autres sommets peuvent être atteints.

Une **anti-arborescence** est un graphe orienté, connexe, sans cycle et possédant au moins une **anti-racine**, c'est-à-dire un sommet pouvant être atteint à partir de n'importe quel autre sommet.

Un **arbre maximal** dans un graphe $G = (V, E)$ est un arbre possédant le même ensemble de sommets que G.

Une **arborescence maximale** (resp. anti-arborescence maximale) dans un graphe $G = (V, A)$ est une arborescence (resp. anti-arborescence) possédant le même ensemble de sommets que G.

Le graphe $G' = (V', E' \cup A')$ où $V' \subseteq V$ et dont l'ensemble des arcs et des arêtes est formé des arcs et des arêtes de G ayant leurs deux extrémités dans V' est appelé **sous-graphe** de G engendré par V'.

Le graphe $G' = (V, E' \cup A')$ où $E' \cup A' \subseteq E \cup A$ est appelé **graphe partiel** de G engendré par $E' \cup A'$.

Le **degré** d'un sommet v_i dans le graphe $G = (V, E \cup A)$, noté $d_G(i)$, est le nombre d'arcs et d'arêtes de G incidents à v_i.

Le **degré entrant** d'un sommet v_i dans le graphe $G = (V, E \cup A)$, noté $d_G^-(i)$, est le nombre d'arcs de G arrivant dans v_i.

Le **degré sortant** d'un sommet v_i dans le graphe $G = (V, E \cup A)$, noté $d_G^+(i)$, est le nombre d'arcs de G quittant v_i.

Un graphe $G = (V, E \cup A)$ est un **graphe pair** si tous les sommets v_i de V sont de degré pair.

Un graphe $G = (V, E \cup A)$ (où $A \neq \emptyset$) est dit **symétrique** si pour tout sommet v_i de V on a $d_G^-(i) = d_G^+(i)$.

Remarque: Certains auteurs utilisent le terme de graphe **pseudo-symétrique** pour désigner un graphe où pour tout sommet v_i de V on a $d_G^-(i) = d_G^+(i)$. Ils emploient alors le terme graphe symétrique pour

parler d'un graphe dans lequel pour chaque arc $(v_i, v_j) \in A$, il existe un arc $(v_j, v_i) \in A$ (voir [Ber83]).

Un graphe $G = (V, E \cup A)$ est dit **équilibré** si pour tout sous-ensemble $X \subseteq V$ la différence entre le nombre d'arcs orientés de X vers $V \setminus X$ et celui des arcs orientés de $V \setminus X$ vers X est inférieure ou égale au nombre d'arêtes reliant X à $V \setminus X$.

Un **cycle eulérien** dans un graphe non orienté $G = (V, E)$ est un cycle passant exactement une fois par chaque arête de E. Lorsque G est orienté ou mixte, on appelle **circuit eulérien** le circuit traversant exactement une fois chaque arc ou arête de $E \cup A$.

On appelle **graphe eulérien** un graphe $G = (V, E \cup A)$ connexe qui possède un circuit ou un cycle eulérien. Les conditions nécessaires et suffisantes pour qu'un graphe connexe soit eulérien sont les suivantes:

> Un graphe connexe, non orienté est eulérien si et seulement si il est pair [Eul1736].
>
> Un graphe $G = (V, A)$ connexe, orienté est eulérien si et seulement si $d_G^-(i) = d_G^+(i)$ pour tout sommet $v_i \in V$.
>
> Un graphe $G = (V, E \cup A)$ connexe, mixte est eulérien si et seulement si il est pair et équilibré [For62].

Soit un graphe $G = (V, E \cup A)$ et soit $R \subseteq E \cup A$. On appelle **circuit (ou cycle) couvrant** R un circuit (ou cycle) passant au moins une fois par chaque arête ou arc de R. Lorsque les éléments de R sont traversés exactement une fois dans un circuit (ou cycle), on dit alors que celui-ci **couvre exactement** R.

Un **couplage** dans un graphe non orienté $G = (V, E)$ est un ensemble d'arêtes de G deux à deux non adjacentes. Un **couplage parfait** dans G est un couplage dont les arêtes sont incidentes à tous les sommets de G. Le **poids d'un couplage** dans G est la somme des coûts des arêtes de G appartenant au couplage. Un **couplage parfait de coût minimum** dans G est un couplage parfait dans G dont le poids est minimum.

Chapitre 1

Problèmes de tournées sur les arcs: revue des principales heuristiques

1.1 Introduction

La distribution et la collecte de biens sont des activités qui apparaissent fréquemment au sein de certaines entreprises et collectivités. Il s'agit généralement d'acheminer des marchandises d'un dépôt vers des succursales, d'apporter des commandes à des clients, d'entretenir des réseaux routiers, de distribuer du courrier, de ramasser des ordures, etc. Ces opérations sont souvent très coûteuses et de substantielles économies peuvent être réalisées en réduisant la longueur des trajets à effectuer.
Suivant la nature du service fourni par les véhicules et en fonction de la répartition géographique des clients, deux grands types de modélisation se dégagent. La première consiste à représenter les clients à l'aide des sommets (ou noeuds) d'un réseau. Les problèmes basés sur cette modélisation sont appelés problèmes de tournées sur les sommets (PTS) (en anglais node routing problems). Une telle modélisation est bien adaptée lorsque les clients sont ponctuels et localisés de façon distincte. Les PTS ont fait et font encore l'objet d'études très abondantes. De nombreuses variantes de PTS existent, allant du problème de base qu'est le problème du voyageur de commerce (PVC) à des problèmes beaucoup plus contraints comme les problèmes de tournées sur les sommets avec fenêtres de temps (PTSFT). Un autre type de modélisation associe les clients aux arêtes ou aux arcs d'un réseau. On parle alors de problèmes de tournées sur les arcs (PTA) (en anglais ARP's pour Arc Routing Problems). Moins présents dans la littérature que les PTS, ils apparaissent lorsque les clients se trouvent distribués le long de rues, ou lorsque les rues elles-mêmes nécessitent un service.

Tout comme pour les PTS, il existe différentes variantes de PTA. La plupart d'entre elles sont NP-dures et les problèmes pratiques qu'elles modélisent sont généralement de grande taille. C'est pourquoi rares sont les méthodes exactes qui ont été développées pour résoudre ce type de problèmes. Citons néanmoins Hirabayashi, Saruwatari et Nishida [Hir92] qui ont développé un algorithme exact pour le problème de tournées sur les arcs avec capacité (PTAC) (en anglais CARP pour Capacitated Arc Routing Problem), Christofides, Campos, Corberán et Mota [Chr81], Corberán et Sanchis [Cor94], Letchford [Let96] ainsi que Ghiani et Laporte [Ghi2000], qui ont traité le problème du postier rural

(PPR) (en anglais RPP pour Rural Postman Problem). Les méthodes heuristiques ont fait l'objet, quant à elles, de nombreuses études. C'est de ces méthodes dont il est question dans ce chapitre. Nous allons présenter une revue des principales heuristiques développées pour la résolution des différentes variantes de PTA. Cette revue n'est bien sûr pas exhaustive, le but n'étant pas ici de recenser toutes les méthodes existantes mais plutôt de donner un aperçu aussi complet que possible des principales techniques de résolution utilisées.

1.2 Les problèmes sans contrainte de capacité

1.2.1 Le problème du postier chinois (PPC)

Le problème du postier chinois a été introduit pour la première fois par le mathématicien chinois Meigu Guan en 1962 [Gua62]. Etant donné un graphe $G = (V, E \cup A)$, il s'agit de déterminer un circuit de longueur totale minimum traversant au moins une fois chaque arc et arête du graphe.
Le PPC peut être résolu polynomialement dans les graphes non orientés, les graphes orientés et dans les graphes mixtes pairs. Dans les autres cas, le PPC est un problème NP-dur.

Bien que ce chapitre soit consacré aux méthodes heuristiques, nous mentionnerons brièvement les méthodes de résolution exactes utilisées pour traiter les cas du PPC qui sont polynomiaux, car ces méthodes apparaissent fréquemment dans des heuristiques servant à résoudre des problèmes de tournées sur les arcs plus difficiles.

1.2.1.1 Le PPC dans les graphes non orientés

Soit $G = (V, E)$ un graphe connexe, non orienté et dont le coût associé à chaque arête est identique quel que soit le sens dans lequel cette arête est parcourue (nous traiterons à la section 1.2.1.4 le cas où cette dernière hypothèse n'est pas remplie).

Si G est pair, le problème du postier chinois revient à construire un cycle eulérien dans G. Des algorithmes polynomiaux existent qui permettent de construire un tel cycle. L'algorithme que nous présentons est décrit dans [Edm73], sa complexité est de l'ordre de $O(|E|)$. D'autres méthodes sont présentées dans [Fle90].

Algorithme: CYCLE_EULERIEN

ENTREE: Un graphe non orienté $G = (V, E)$ connexe pair.

SORTIE: Un cycle eulérien T dans G.

Etape 1. Déterminer un cycle simple T dans G. Si T couvre exactement E, STOP.

Etape 2. Choisir un sommet v appartenant à T et incident à une arête ne figurant pas dans T. Construire un deuxième cycle simple T' contenant v et tel que T et T' n'aient pas d'arête commune.

Etape 3. Fusionner T et T' pour former un cycle T''. Cette fusion se fait en partant du sommet v, en parcourant l'ensemble du cycle T puis étant revenu au sommet v en visitant l'ensemble du cycle T' pour finalement terminer en v.

Etape 4. Poser $T := T''$.
Si T couvre exactement E STOP, sinon retourner à l'étape 2.

Dans le cas où G n'est pas pair, il faut dupliquer un certain nombre d'arêtes de G de façon à ce que le nouveau graphe G' ainsi obtenu soit pair. Il est alors possible de construire un cycle eulérien dans G'. Le choix des arêtes à dupliquer se fait de manière à ce que ce cycle eulérien soit de longueur minimum. Ces arêtes peuvent être déterminées en résolvant un problème de couplage parfait de coût minimum dans un graphe auxiliaire complet ayant pour sommets les sommets de degré impair dans G.

L'algorithme présenté ci-dessous est similaire à ceux figurant dans [Edm73] et [Orl74].

Algorithme: PPC_NON_ORIENTE

ENTREE: Un graphe non orienté connexe $G = (V, E)$.

SORTIE: Un cycle T solution optimale du PPC dans G.

Etape 1. Si G est pair, appliquer CYCLE_EULERIEN sur G. Le cycle T ainsi obtenu est la solution optimale de notre problème. STOP.

Etape 2. Déterminer l'ensemble V_0 des sommets de degré impair dans G. Construire le graphe complet G_0 dont l'ensemble des sommets est V_0. Le coût de chaque arête (v_i, v_j) de G_0 est égal à la longueur de la plus courte chaîne SC_{ij} entre v_i et v_j dans G.

Etape 3. Trouver un couplage parfait de coût minimum dans le graphe G_0.

Etape 4. Poser $G' = G$. Pour chaque arête (v_i, v_j) du couplage optimal, ajouter dans G' toutes les arêtes appartenant à la plus courte chaîne SC_{ij} reliant v_i et v_j dans G.

Etape 5. Déterminer un cycle eulérien T en appliquant CYCLE_EULERIEN sur G'. T est la solution optimale de notre problème. STOP.

1.2.1.2 Le PPC dans les graphes orientés

Le problème du postier chinois dans un graphe $G = (V, A)$ orienté peut être résolu polynomialement. Nous supposerons tout au long de cette section que G est fortement connexe. Si G ne remplit pas cette dernière condition, le problème n'admet pas de solution. Comme pour le PPC dans les graphes non orientés, deux cas peuvent être considérés, suivant que G est symétrique ou ne l'est pas.

Si G est symétrique, le problème du postier chinois revient à trouver un circuit eulérien dans G. Pour cela nous pouvons utiliser l'algorithme décrit dans [Aar51] et [Edm73]. Cet algorithme est présenté à la page suivante.

Algorithme: CIRCUIT_EULERIEN

ENTREE: Un graphe orienté $G = (V, A)$ fortement connexe et symétrique.

SORTIE: Un circuit eulérien T dans G.

Etape 1. Construire une anti-arborescence maximale dans G dont l'anti-racine est un sommet $v \in V$.

Etape 2. Ordonner et numéroter les arcs sortant de v de façon arbitraire. Ordonner et numéroter les arcs sortant de chaque sommet appartenant à $V \setminus v$ de telle sorte que le dernier arc numéroté fasse partie de l'anti-arborescence.

Etape 3. Construire un circuit T partant de v et utilisant toujours, pour quitter un sommet, l'arc non encore parcouru ayant le plus petit numéro.

Lorsque G n'est pas symétrique, des copies de certains arcs doivent être ajoutées à G de manière à obtenir un graphe orienté G' qui soit symétrique. Les arcs à copier peuvent être déterminés en résolvant un problème de transport comme dans [Bel74] ou dans [Orl74]. L'algorithme a alors une complexité de l'ordre de $O(|A||V|^2)$. Edmonds et Johnson [Edm73] déterminent les arcs à copier en résolvant un problème de flot de coût minimum. Dans ce cas la complexité de l'algorithme est de l'ordre de $O(|V|^3)$.

Une toute autre approche est proposée par Lin et Zhao [Lin88] pour résoudre le problème du postier chinois dans le cas orienté. Partant d'une formulation du problème en terme de programmation linéaire en nombres entiers, et utilisant le théorème des écarts complémentaires, ces deux auteurs définissent un ensemble de conditions d'optimalité. Toute solution satisfaisant cet ensemble de conditions est alors une solution optimale du PPC. L'algorithme qu'ils proposent consiste à générer de manière itérative une solution satisfaisant ces conditions. Sa complexité est de l'ordre de $O((|A| - |V| + 1)|V|^2)$, ce qui, lorsque le graphe traité est de faible densité, est meilleur que la complexité des deux algorithmes mentionnés dans le paragraphe précédent.

L'algorithme que nous décrivons à la page suivante est similaire à celui présenté dans [Bel74] et dans [Orl74].

Algorithme: PPC_ORIENTE

ENTREE: Un graphe orienté $G = (V, A)$ fortement connexe.

SORTIE: Un circuit T solution optimale du PPC dans G.

Etape 1. Si G est symétrique, appliquer CIRCUIT_EULERIEN sur G. Le circuit T ainsi obtenu est une solution optimale de notre problème. STOP.

Etape 2. Pour tout sommet $v_i \in V$, calculer $f_i = d_G^-(i) - d_G^+(i)$.
Poser $I = \{v_i \in V | f_i > 0\}$ et $J = \{v_j \in V | f_j < 0\}$.

Etape 3. Construire le réseau $G_0 = (V_0, A_0)$ où $V_0 = I \cup J$ et $A_0 = (I \times J)$. Le coût d'un arc $(v_i, v_j) \in I \times J$ est égal à la longueur du plus court chemin SP_{ij} reliant v_i à v_j dans G. Résoudre le problème de transport défini dans G_0 où chaque sommet $v_i \in I$ a une disponibilité valant f_i et chaque sommet $v_j \in J$ possède une demande valant $|f_j|$.

Etape 4. Soit x_{ij} le nombre d'unités transportées dans G_0 entre les sommets v_i et v_j dans la solution optimale du problème de transport.
Poser $G' = G$. Pour chaque paire $(v_i, v_j) \in A_0$, ajouter à G' x_{ij} copies des arcs appartenant au plus court chemin SP_{ij} reliant v_i à v_j dans G.

Etape 5. Construire un circuit eulérien T dans le graphe G'. T est la solution optimale du PPC dans G. STOP.

1.2.1.3 Le PPC dans les graphes mixtes

Le problème du postier chinois dans un graphe mixte $G = (V, E \cup A)$ est un problème NP-dur [Pap76]. Toutefois, lorsque G est pair, le problème peut se résoudre polynomialement [Edm73], [Min79], [Fre79]. Nous supposons comme pour le cas orienté que G est fortement connexe. Une méthode exacte résolvant le PPC lorsque G est pair sera tout d'abord décrite. Des méthodes heuristiques seront ensuite présentées.

Lorsque G est pair, deux cas peuvent être considérés selon que G est équilibré ou non.

Si G est équilibré, la résolution du PPC consiste à trouver un circuit eulérien dans G. Une première méthode pour déterminer un tel circuit revient à orienter certaines arêtes de G de façon à obtenir un graphe mixte G' symétrique. Cela se fait en résolvant un problème de flot compatible dans un graphe auxiliaire [For62]. Les arêtes de G' sont ensuite orientées pour obtenir un graphe G'' orienté et symétrique. En appliquant CIRCUIT_EULERIEN à G'', on obtient un circuit eulérien dans G'' qui est également un circuit eulérien dans G et qui est une solution optimale du PPC. Cette méthode est présentée à la page suivante. Une autre technique pour déterminer un circuit eulérien dans G consiste à adapter l'algorithme CIRCUIT_EULERIEN aux graphes mixtes [Edm73].

Algorithme: EULERIEN_MIXTE

ENTREE: Un graphe mixte $G = (V, E \cup A)$ fortement connexe pair et équilibré.

SORTIE: Un circuit eulérien T dans G.

Etape 1. *(Construction d'un graphe G' symétrique)*
Construire un graphe $H = (V, A \cup A')$ en remplaçant chaque arête de E par une paire d'arcs d'orientation opposée. Attribuer à chaque arc de $A \cup A'$ une borne supérieure de flot de 1. Attribuer à chaque arc de A une borne inférieure de flot valant 1 et à chaque arc de A' une borne inférieure de flot nulle.
Déterminer un flot compatible dans H. Soit x_{ij} la valeur du flot associée à l'arc (v_i, v_j).
Poser $G' = G$ et orienter chaque arête (v_i, v_j) de E de v_i vers v_j lorsque $x_{ij} = 1$ et $x_{ji} = 0$.

Etape 2. *(Construction d'un graphe orienté G'' symétrique)*
Poser $G'' = G'$. Si G'' est un graphe orienté aller à l'étape 3.
Construire un cycle eulérien sur chacune des composantes connexes du graphe partiel induit par les arêtes de G''. Donner une orientation à chacun des cycles construits et diriger chaque arête de G'' en fonction de cette orientation.

Etape 3. *(Construction d'un circuit eulérien)*
Construire un circuit eulérien T dans G'' en utilisant CIRCUIT_EULERIEN. T est un circuit eulérien dans G et une solution optimale du PPC.

Lorsque le graphe G est pair, mais pas équilibré, le PPC peut également se résoudre polynomialement. En orientant ses arêtes, en dupliquant certains arcs, le graphe G est transformé en un graphe orienté symétrique et pair. La solution optimale du PPC est alors obtenue en appliquant CIRCUIT_EULERIEN sur ce dernier graphe. L'orientation finale des arêtes de G ainsi que le choix des arcs à dupliquer sont déterminés en résolvant un problème de flot compatible de coût minimum dans un graphe auxiliaire. Cette méthode proposée initialement par Edmonds et Johnson [Edm73] est également présentée dans [Min78], [Min79] et [Fre79].

Algorithme: PPC_MIXTE_PAIR

ENTREE: Un graphe $G = (V, E \cup A)$ mixte, pair et fortement connexe.

SORTIE: Un circuit T solution optimale du PPC dans G.

Etape 1. Construire un graphe orienté $G_1 = (V, A \cup A_1)$ en orientant de façon arbitraire les arêtes de G.
Calculer pour chaque sommet v_i de G_1 la valeur $f_i = d^-_{G_1}(i) - d^+_{G_1}(i)$.
Si tous les f_i sont nuls, poser $G_3 = G_1$ et aller à l'étape 3.

Etape 2. Construire le graphe $G_2 = (V, A \cup A_1 \cup A_2 \cup A_3)$ tel qu'à chaque arête (v_i, v_j) de E correspondent dans $A_1 \cup A_2$ deux arcs de direction opposée et de coût c_{ij} et tel qu'à chaque arc (v_i, v_j) de A_1 soit associé un arc de coût nul (v_j, v_i) dans A_3.
Attribuer à chaque arc de $A \cup A_1 \cup A_2$ une capacité infinie et à chaque arc de A_3 une capacité de 1.
Définir une disponibilité de valeur $\frac{f_i}{2}$ sur chaque sommet v_i tel que $f_i > 0$ et une demande de $\frac{-f_i}{2}$ sur tout sommet v_i tel que $f_i < 0$.
Trouver un flot de coût minimum dans G_2 satisfaisant les demandes.
Soit x_{ij} la valeur du flot traversant un arc de $A \cup A_1 \cup A_2 \cup A_3$.
Poser $G_3 = G$. Pour chaque arc $(v_i, v_j) \in A_3$ orienter l'arête (v_i, v_j) de G_3 de v_i vers v_j si $x_{ij} = 1$ et de v_j vers v_i dans le cas contraire.
Augmenter G_3 en ajoutant $2 \cdot x_{ij}$ copies de chaque arc (v_i, v_j) appartenant à $A \cup A_1 \cup A_2$.
G_3 est maintenant un graphe orienté pair et symétrique.

Etape 3. Construire un circuit eulérien T dans G_3 en utilisant CIRCUIT_EULERIEN. T est une solution optimale du PPC dans G.

Remarque:

L'algorithme ci-dessus correspond à celui présenté par Minieka dans [Min79] et dans [Min78]. Nous avons toutefois défini le graphe G_2 dans lequel un flot compatible de coût minimum est recherché de manière légèrement différente.
Minieka résout ce problème de flot dans un graphe $G'_2 = (V, A \cup A_1 \cup A_2 \cup A_3)$ dans lequel les demandes associées aux sommets v_i tels que $f_i < 0$ valent $-f_i$, les disponibilités attribuées aux sommets v_i tels que $f_i > 0$ valent f_i et la capacité des arcs de A_3 vaut 2. On peut montrer qu'il existe toujours un flot compatible de coût minimum dans G'_2 traversant chaque arc un nombre pair de fois. Etant donné un tel flot dans G'_2, une arête (v_i, v_j) est orientée de v_i vers v_j si le flot x_{ij} traversant l'arc (v_i, v_j) de A_3 vaut 2. L'orientation est inverse lorsque x_{ij} vaut 0. Un nombre x_{ij} de copies de chaque arc $(v_i, v_j) \in A \cup A_1 \cup A_2$ est également ajouté à G pour former G_3.
S'il existe toujours un flot compatible de coût minimum dans G'_2 dont les valeurs sont paires, il peut exister d'autres flots compatibles dans G'_2 de coût minimum également, mais traversant certains arcs un nombre impair de fois. Le flot x_{ij} transitant par un arc (v_i, v_j) de A_3 peut donc avoir une valeur de 1. L'arête (v_i, v_j) de G associée à cet arc

reste alors non orientée. Frederickson [Fre79] a mis en évidence le fait qu'avec un tel flot optimal dans G'_2, le graphe G_3 obtenu à la fin de l'étape 2 n'est plus forcément pair (cf. figure 1.8). Il a développé une procédure appelée EVENPARITY, utilisée après l'étape 2 et avant l'étape 3 afin de "corriger" G_3 lorsque celui-ci n'est pas pair.

Les figures ci-dessous illustrent le fonctionnement de l'algorithme PPC_MIXTE_PAIR.

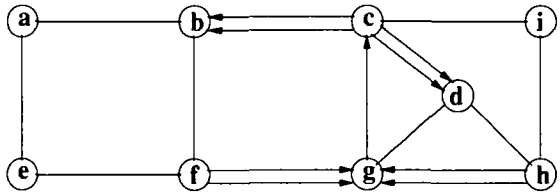


FIG. 1.1 – *Graphe initial G.*

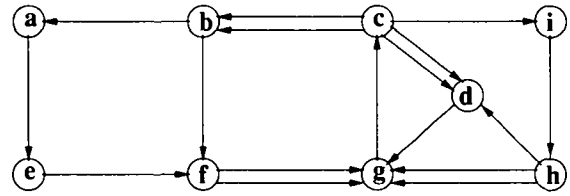


FIG. 1.2 – *Graphe G_1.*

La figure 1.1 représente le graphe de départ G. Ce graphe est pair et non équilibré. Toutes les arêtes et tous les arcs ont un coût valant 1. En orientant chacune des arêtes de G, on obtient le graphe G_1 de la figure 1.2.

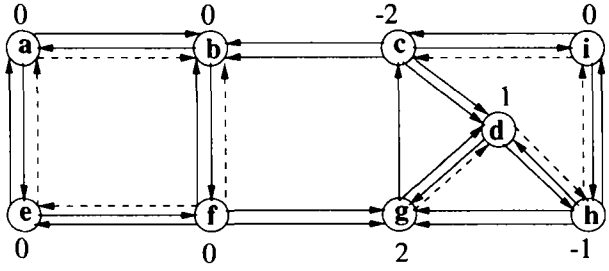


FIG. 1.3 – *Graphe auxiliaire G_2.*

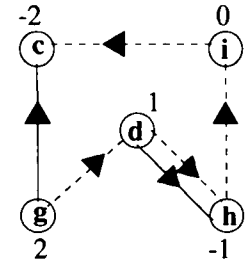


FIG. 1.4 – *Un flot optimal dans G_2.*

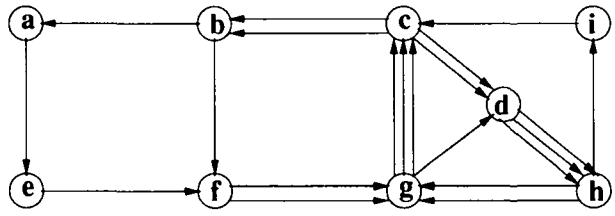


FIG. 1.5 – *Graphe G_3.*

La figure 1.3 représente le graphe G_2 dans lequel un flot compatible de coût minimum satisfaisant les demandes et les disponibilités est cherché. Les arcs appartenant à A_3 sont en pointillés. Ils ont une capacité de 1 et un coût nul. Les autres arcs ont un coût de 1 et une capacité infinie. Les nombres situés près des sommets correspondent aux demandes (s'ils sont négatifs) et aux disponibilités (s'ils sont positifs). Un flot compatible de coût minimum dans G_2 est représenté sur la figure 1.4. La figure 1.5 correspond au graphe G_3 construit à partir du flot optimal de la figure 1.4. Ce graphe est complètement orienté, et symétrique. Nous pouvons donc lui appliquer CIRCUIT_EULERIEN.

Nous allons illustrer le problème mis en évidence par Frederickson [Fre79] qui peut se produire lorsque le graphe auxiliaire G'_2 est utilisé à la place de G_2. L'exemple traité est celui de la figure 1.1.

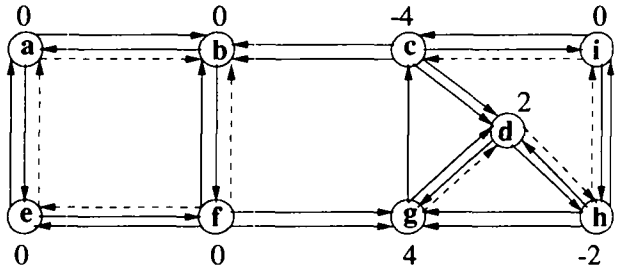


FIG. 1.6 – *Graphe auxiliaire G'_2.*

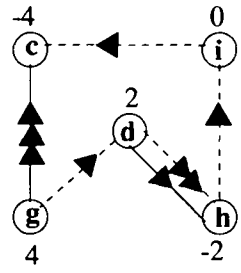


FIG. 1.7 – *Un flot optimal dans G'_2.*

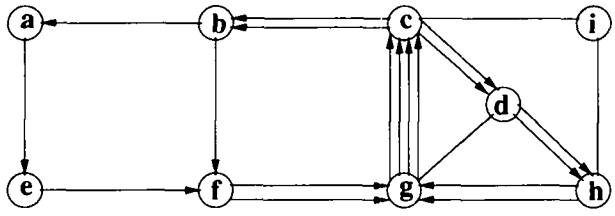


FIG. 1.8 – *Graphe G_3.*

La figure 1.6 représente le graphe G'_2. Les arcs en pointillés sont les arcs de A_3. Ils ont une capacité de 2 et un coût nul. Tous les autres arcs ont un coût valant 1 et une capacité infinie. Les demandes et les disponibilités ont doublé par rapport à celles du graphe G_2 de la figure 1.3. Un flot compatible de coût minimum dans G'_2 est représenté dans la figure 1.7. Des quantités impaires de flot traversent les arcs (i,c), (h,i) et (g,d) appartenant à A_3. Les arêtes de G associées à ces arcs restent alors non orientées. Le graphe G_3 correspondant au flot optimal de la figure 1.7 est un graphe mixte, équilibré, mais il n'est pas pair. Il n'existe donc pas de circuit eulérien dans ce graphe.

Nous allons à présent considérer le cas où le graphe G n'est pas pair. Dans ce cas, le problème du postier chinois est NP-dur [Pap76]. Deux méthodes heuristiques seront présentées. Elles ont été développées par Frederickson [Fre79] et sont basées sur des techniques semblables à celles utilisées lorsque G est pair.

Le premier algorithme consiste à transformer G en un graphe mixte pair G' et à utiliser PPC_MIXTE_PAIR pour déterminer un circuit optimal dans G'. Ce circuit n'est pas forcément optimal dans G.

Algorithme: PPC_MIXTE1

ENTREE: Un graphe $G = (V, E \cup A)$ mixte et fortement connexe.

SORTIE: Un circuit T solution (pas forcément optimale) du PPC dans G.

Etape 1. Déterminer l'ensemble V_0 des sommets de degré impair dans G.
Construire un graphe complet non orienté $G_0 = (V_0, E_0)$. Le coût d'une arête $(v_i, v_j) \in E_0$ est égal à la longueur de la plus courte chaîne SC_{ij} reliant v_i à v_j dans G. Lors du calcul de ces plus courtes chaînes, l'orientation des arcs de G n'est pas prise en compte (i.e. les arcs de G sont considérés comme des arêtes).
Déterminer un couplage parfait de coût minimum dans G_0.
Poser $G' = G$.
Pour chaque arête (v_i, v_j) du couplage optimal, ajouter à G' une copie de chaque arête et arc appartenant à la plus courte chaîne SC_{ij} reliant v_i à v_j dans G.

Etape 2. Construire une solution optimale T du PPC dans G' en utilisant PPC_MIXTE_PAIR.

Exemple:

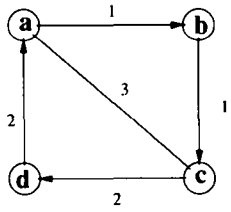


FIG. 1.9 – *Graphe initial G.*

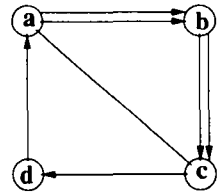


FIG. 1.10 – *Graphe pair G'.*

La figure 1.9 de l'exemple ci-dessus représente le graphe de départ G. Les nombres situés sur les arcs et les arêtes correspondent aux coûts de ces arcs et arêtes. Le graphe G' représenté sur la figure 1.10 est obtenu en dupliquant les arcs (a, b) et (b, c). La longueur d'un circuit eulérien dans G' vaut 11 et correspond à la valeur de la solution du PPC dans G fournie par PPC_MIXTE1.

La deuxième méthode revient à transformer G en un graphe G' mixte symétrique, puis à transformer G' en un graphe mixte G'' qui soit symétrique et pair. Les arêtes de G'' sont ensuite orientées de manière à ce que la parité et la symétrie soient conservées. CIRCUIT_EULERIEN est alors appliqué sur G'' pour obtenir un circuit optimal dans G''. Ce circuit n'est pas forcément optimal dans G.

Algorithme: PPC_MIXTE2

ENTREE: Un graphe $G = (V, E \cup A)$ mixte et fortement connexe.

SORTIE: Un circuit T solution (pas forcément optimale) du PPC dans G.

Etape 1. *(Construction d'un graphe $G' = (V, E' \cup A')$ symétrique)*
Construire un graphe orienté $G_1 = (V, A \cup A_1)$ en orientant de façon arbitraire les arêtes de G.
Calculer pour chaque sommet v_i de G_1 la valeur $f_i = d^-_{G_1}(i) - d^+_{G_1}(i)$.
Si tous les f_i sont nuls poser $G' = G_1$ et aller à l'étape 2.
Construire le graphe $G' = (V, A \cup A_1 \cup A_2 \cup A_3)$ tel qu'à chaque arête (v_i, v_j) de E correspondent dans $A_1 \cup A_2$ deux arcs de direction opposée et de coût c_{ij} et tel qu'à chaque arc (v_i, v_j) de A_1 soit associé un arc de coût nul (v_j, v_i) dans A_3.
Attribuer à chaque arc de $A \cup A_1 \cup A_2$ une capacité infinie et à chaque arc de A_3 une capacité de 2.
Définir une disponibilité de valeur f_i sur chaque sommet v_i tel que $f_i > 0$ et une demande de $-f_i$ sur tout sommet v_i tel que $f_i < 0$.
Trouver un flot compatible de coût minimum dans G' satisfaisant les demandes. Soit x_{ij} la valeur du flot traversant un arc de $A \cup A_1 \cup A_2 \cup A_3$.
Poser $G' = G$. Pour chaque arc $(v_i, v_j) \in A_3$ orienter l'arête (v_i, v_j) de G' de v_i vers v_j si $x_{ij} = 2$ et de v_j vers v_i lorsque $x_{ij} = 0$. Si $x_{ij} = 1$ laisser l'arête (v_i, v_j) non orientée.
Augmenter G' en ajoutant x_{ij} copies de chaque arc (v_i, v_j) appartenant à $A \cup A_1 \cup A_2$.

Etape 2. *(Construction d'un graphe $G'' = (V, E'' \cup A')$ symétrique et pair)*
Soit E' l'ensemble des arêtes qui ne sont pas orientées à la fin de la première étape. Dans le graphe partiel $H' = (V, E')$ de G' déterminer l'ensemble V_0 des sommets de degré impair.
Construire le graphe complet non orienté $G_0 = (V_0, E_0)$. Le coût d'une arête $(v_i, v_j) \in E_0$ est égal à la longueur de la plus courte chaîne SC_{ij} reliant v_i à v_j dans le graphe partiel $H = (V, E)$. Si une telle chaîne n'existe pas dans H, les sommets v_i et v_j sont reliés dans G_0 par une arête de coût infini.
Déterminer un couplage parfait de coût minimum dans G_0.
Poser $G'' = G'$.
Pour chaque arête (v_i, v_j) du couplage optimal, ajouter à G'' une copie de chaque arête appartenant à la plus courte chaîne SC_{ij} reliant v_i à v_j dans H.

Etape 3. *(Construction d'un circuit eulérien dans G'')*
Déterminer un circuit T solution optimale du PPC dans G'' en lui appliquant les étapes 2 et 3 de EULERIEN_MIXTE.

Exemple:

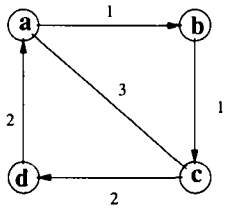


FIG. 1.11 -

Graphe initial G.

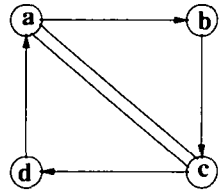


FIG. 1.12 -

Graphe symétrique et pair G''.

La figure 1.11 représente le graphe de départ. Ce graphe est identique à celui de la figure 1.9. En appliquant l'étape 2 de PPC_MIXTE2, nous obtenons le graphe G'' de la figure 1.12 (l'arête (a, c) est dupliquée). La longueur d'un circuit eulérien dans G'' vaut 12. Cette valeur est celle de la solution du PPC dans G fournie par PPC_MIXTE2.

Frederickson [Fre79] a montré que les algorithmes PPC_MIXTE1 et PPC_MIXTE2 fournissent toujours des solutions dont la valeur est inférieure ou égale à deux fois celle de l'optimum. Il existe des cas où cette borne est atteinte. En combinant ces deux méthodes, c'est-à-dire en appliquant successivement PPC_MIXTE1 et PPC_MIXTE2 à un problème donné et retenant la meilleure des deux solutions obtenues, Frederickson a montré que la borne obtenue est alors de $\frac{5}{3}$. A notre connaissance, aucun exemple pour lequel cette borne est atteinte n'a encore été produit.

1.2.1.4 Le PPC avec vent

Nous avons traité à la section 1.2.1.1 le problème du postier chinois dans un graphe $G = (V, E)$ connexe dont les coûts étaient symétriques (i.e. $c_{ij} = c_{ji} \; \forall \; (v_i, v_j) \in E$). Lorsque les coûts ne sont pas symétriques, on parle alors de problème du postier chinois avec vent (PPCV). Ce problème, introduit pour la première fois par Minieka, [Min79] correspond au cas où un postier circulant à vélo doit livrer son courrier un jour de fort vent. Sa vitesse de déplacement le long d'une route est alors plus ou moins rapide suivant qu'il roule contre ou avec le vent.

Contrairement au cas où les coûts sont symétriques, le PPCV est un problème NP-dur [Gua84]. Seuls quelques cas très particuliers peuvent être résolus polynomialement. Guan a par exemple montré que si tout cycle de G a la même longueur quel que soit son sens de parcours, alors le PPCV est polynomial.

Nous présentons à la page suivante une heuristique proposée par Guan [Gua84] pour le cas général du PPCV.

Algorithme: GUAN_PPCV

ENTREE: Un graphe $G = (V, E)$ connexe non orienté dont les coûts ne sont pas symétriques.

SORTIE: Un cycle T solution (pas forcément optimale) du PPCV dans G.

Etape 1. Considérer le graphe $G' = (V, E)$ dans lequel on associe à chaque arête (v_i, v_j) un coût $c'_{ij} = \frac{c_{ij}+c_{ji}}{2}$.

Etape 2. G' est un graphe connexe non orienté avec coûts symétriques. Résoudre le PPC dans G' en utilisant PPC_NON_ORIENTE décrit en 1.2.1.1. Soit T_0 le cycle solution optimale du PPC dans G'.

Etape 3. Décomposer arbitrairement T_0 en un ensemble de cycles $T_1, T_2, ..., T_r$ disjoints par les arêtes.
Orienter chacun des cycles T_i $(i = 1, ..., r)$ dans le sens de sa longueur minimum de manière à obtenir des circuits T'_i $(i = 1, ..., r)$.

Etape 4. Poser $T = \bigcup_{i=1}^{r} T'_i$. T est une solution (pas forcément optimale) du PPCV dans G.

L'algorithme ci-dessus fournit une solution optimale au PPCV lorsque tous les cycles de G ont la même longueur quel que soit leur sens de parcours. Dans ce cas, les étapes 3 et 4 sont inutiles et le cycle T_0 est une solution optimale du PPCV dans G.

Un autre cas dans lequel le PPCV est polynomial est celui où le graphe G est eulérien (i.e. lorsque G est pair). Win [Win89] a adapté l'algorithme PPC_MIXTE_PAIR au problème du postier chinois avec vent de manière à le résoudre de façon optimale lorsque G est pair. Son algorithme est présenté ci-dessous.

Algorithme: EULERIEN_PPCV

ENTREE: Un graphe non orienté $G = (V, E)$ pair, connexe et dont les coûts ne sont pas symétriques.

SORTIE: Un cycle T solution optimale du PPCV dans G.

Etape 1. Appliquer l'étape 1 de PPC_MIXTE2 moyennant les modifications suivantes:
les arêtes $(v_i, v_j) \in E$ sont orientées de v_i vers v_j si $c_{ij} < c_{ji}$ et de v_j vers v_i dans le cas contraire (plus d'orientation arbitraire).
Un coût $\frac{c_{ji}-c_{ij}}{2}$ (au lieu de 0 dans PPC_MIXTE2) est attribué à chaque arc (v_j, v_i) de A_3.

Etape 2. Appliquer l'étape 3 de PPC_MIXTE_PAIR de façon à obtenir un cycle T dans G. Ce cycle est une solution optimale du PPCV dans G.

Une heuristique traitant le cas général du PPCV a également été proposée par Win [Win89]. Elle consiste à transformer le graphe de départ G en un graphe eulérien puis à appliquer EULERIEN_PPCV sur ce graphe.

Algorithme: WIN_PPCV

ENTREE: Un graphe non orienté $G = (V, E)$ connexe et dont les coûts ne sont pas symétriques.

SORTIE: Un cycle T solution (pas forcément optimale) du PPCV dans G.

Etape 1. Si G est eulérien déterminer une solution optimale du PPCV dans G au moyen de EULERIEN_PPCV. STOP.

Etape 2. Construire le graphe $G' = (V, E)$ dont le coût c'_{ij} d'une arête (v_i, v_j) de E vaut $\frac{c_{ij}+c_{ji}}{2}$.

Etape 3. Déterminer l'ensemble V_0 des sommets de degré impair dans G'.
Construire le graphe complet $G_0 = (V_0, E_0)$ où le coût d'une arête $(v_i, v_j) \in E_0$ est égal à la longueur de la plus courte chaîne SC_{ij} entre v_i et v_j dans G'.
Trouver un couplage parfait de coût minimum dans G_0. Poser $G'' = G$.
Pour chaque arête (v_i, v_j) appartenant au couplage optimal, ajouter à G'' une copie de chaque arête appartenant à SC_{ij}.

Etape 4. Le graphe G'' est eulérien. Déterminer un cycle T, solution optimale du PPCV dans G'' en utilisant EULERIEN_PPCV. Ce cycle est une solution (pas forcément optimale) du PPCV dans G.

Win [Win89] a montré que cet algorithme produit toujours des solutions dont la valeur est inférieure ou égale à 2 fois celle de l'optimum. Il existe des exemples pour lesquels cette borne est atteinte.

En se basant sur les algorithmes GUAN_PPCV et WIN_PPCV, Pearn [Pea94] a proposé deux nouvelles heuristiques appelées "modified Guan's algorithm" et "reverse Win's algorithm" pour résoudre le problème du postier chinois avec vent.

1.2.2 Le problème du postier rural (PPR)

Le problème du postier rural (PPR) a été introduit par Orloff [Orl74]. Il s'agit là d'une généralisation du problème du postier chinois. Etant donné un graphe $G = (V, E \cup A)$ et un sous-ensemble R de $E \cup A$, le PPR consiste à déterminer un circuit de longueur minimum passant une fois au moins par chaque arête et arc de R. Les éléments de R seront appelés arcs et arêtes à desservir ou plus simplement clients. Les arêtes et arcs ne se trouvant pas dans R seront nommés arêtes et arcs facultatifs. En posant $R = E \cup A$ nous retrouvons le problème du postier chinois. L'ensemble des sommets de V incidents aux arcs et arêtes à desservir sera noté V_R. Nous noterons encore par $G_R = (V_R, R)$ le sous-graphe partiel de G induit par R.

Le PPR est NP-dur [Len76]. Cependant, lorsque G est non orienté ou orienté et si G_R est connexe, alors le PPR peut se résoudre polynomialement.

Comme nous l'avons fait pour le PPC, nous étudierons dans cette section les principales méthodes de résolution du PPR lorsque G est non orienté, orienté et mixte. Un problème très particulier appelé en anglais "Stacker Crane Problem" (SCP) sera également étudié. Ce problème tire son origine d'une application pratique dans laquelle une grue doit, partant d'une position initiale, effectuer un certain nombre de mouvements avant de retourner à sa position de départ. Finalement nous décrirons toute une série d'outils récemment développés et pouvant être appliqués à n'importe quelle solution du PPR.

1.2.2.1 Le PPR dans les graphes non orientés

Soit $G = (V, E)$ un graphe connexe non orienté, et soit $R \subseteq E$ l'ensemble des arêtes à desservir. Plutôt que de travailler directement dans G dont certains sommets peuvent ne pas être liés aux clients qu'il faut desservir, il est généralement préférable de travailler dans un graphe simplifié composé uniquement des sommets appartenant à V_R. Ce graphe, que nous noterons $G_S = (V_R, R \cup E_S)$ est obtenu à partir de G_R de la manière suivante:

1) Pour chaque paire de sommets $v_i, v_j \in V_R$ insérer dans E_S une arête (v_i, v_j) dont le coût est égal à la longueur de la plus courte chaîne reliant v_i à v_j dans G (les arêtes de R gardent le coût qu'elles ont dans G).

2) Enlever de E_S toutes les arêtes (v_i, v_j) pour lesquelles il existe un sommet $v_k \in V_R$ tel que: $c_{ij} = c_{ik} + c_{kj}$.

3) Enlever de E_S toutes les arêtes (v_i, v_j) parallèles et de coût égal à une arête de R.

A un cycle T_S solution admissible pour le PPR dans G_S correspond un cycle T solution admissible pour le PPR dans G. T est obtenu à partir de T_S en remplaçant dans T_S chaque arête non desservie par la plus courte chaîne qui lui est associée dans G.

L'exemple de la page suivante illustre la construction de G_S (figures 1.13 et 1.14) ainsi que la correspondance entre T_S et T (figures 1.15 et 1.16). Les nombres au-dessus des arêtes représentent leur coût, les arêtes dessinées en gras sont celles qui appartiennent à R.

Exemple:

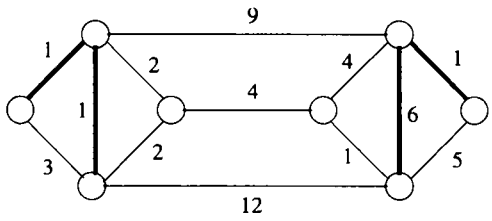


FIG. 1.13 – *Graphe initial G.*

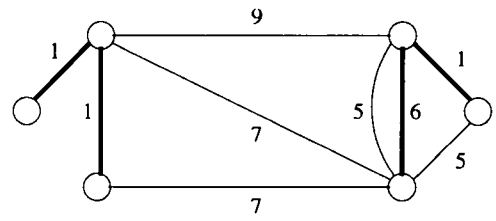


FIG. 1.14 – *Graphe simplifié G_S.*

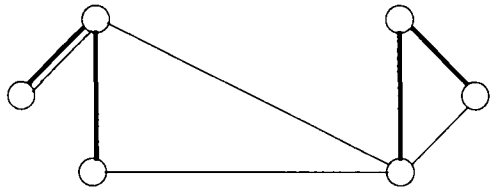


FIG. 1.15 –

Une solution T_S du PPR dans G_S.

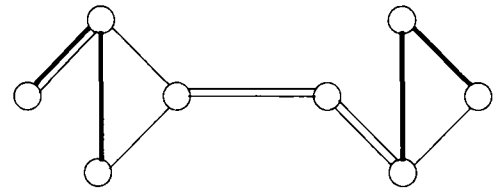


FIG. 1.16 –

La solution T du PPR dans G associée à T_S.

Dès à présent, nous supposerons que le graphe G sur lequel nous allons travailler est déjà simplifié (nous avons donc $V = V_R$).

Nous allons maintenant considérer le cas où le graphe G_R est connexe. Comme nous l'avons déjà mentionné, ce cas peut être résolu polynomialement. Nous présentons ci-dessous un algorithme qui fournit une solution optimale du PPR dans G lorsque G_R est connexe. Les ingrédients utilisés dans cet algorithme sont les mêmes que ceux qui apparaissent dans PPC_NON_ORIENTE.

Algorithme: PPR_NON_ORIENTE_CONNEXE

ENTREE: Un graphe non orienté $G = (V, E)$ connexe et un ensemble $R \subseteq E$ tel que $G_R = (V_R, R)$ est connexe.

SORTIE: Un cycle T solution optimale du PPR dans G.

Etape 1. Poser $G' = G_R$. Si G' est pair aller à l'étape 3.

Etape 2. Déterminer l'ensemble V_0 des sommets de degré impair dans G'.
Construire le graphe complet $G_0 = (V_0, E_0)$ où le coût d'une arête $(v_i, v_j) \in E_0$ est égal à la longueur de la plus courte chaîne SC_{ij} entre v_i et v_j dans G.
Trouver un couplage parfait de coût minimum dans G_0.
Pour chaque arête (v_i, v_j) appartenant au couplage optimal, ajouter à G' une copie de chaque arête appartenant à SC_{ij}.

Etape 3. Construire un cycle eulérien T dans G' en utilisant CYCLE_EULERIEN (cf. 1.2.1.1). T est une solution optimale du PPR dans G.

Lorsque G_R n'est pas connexe, le problème du postier rural dans un graphe non orienté est NP-dur. Frederickson [Fre79] a suggéré de résoudre le PPR en utilisant une heuristique basée sur des principes similaires à ceux figurant dans l'heuristique de Christofides pour le problème du voyageur de commerce [Chr76]. Une telle heuristique est décrite ci-dessous.

Algorithme: PPR_NON_ORIENTE

ENTREE: Un graphe non orienté $G = (V, E)$ connexe et un ensemble $R \subseteq E$.

SORTIE: Un cycle T solution (pas forcément optimale) du PPR dans G.

Etape 1. Soient $C_1, C_2, ..., C_c$ les différentes composantes connexes de G_R.
Poser $G' = G_R$.
Si $c = 1$ aller à l'étape 3.
Construire un graphe complet G'' dont les sommets sont $\{v''_1, v''_2, ..., v''_c\}$.
A chaque arête (v''_p, v''_q) de G'' est associé un coût égal à la longueur de la plus courte chaîne $SC_{C_pC_q}$ reliant un sommet de C_p à un sommet de C_q dans G.

Etape 2. Déterminer un arbre maximal de coût minimum dans G''.
Pour chaque arête (v''_p, v''_q) appartenant à l'arbre optimal, ajouter à G' toutes les arêtes de G appartenant à $SC_{C_pC_q}$.

Etape 3. Appliquer les étapes 2 et 3 de PPR_NON_ORIENTE_CONNEXE de façon à obtenir un cycle eulérien T dans G'. T est une solution (pas forcément optimale) du PPR dans G.

L'algorithme ci-dessus produit toujours une solution dont la valeur est inférieure où égale à $\frac{3}{2}$ fois celle de l'optimum [Fre79].

1.2.2.2 Le PPR dans les graphes orientés

Soit $G = (V, A)$ un graphe orienté, fortement connexe et soit $R \subseteq A$ l'ensemble des arcs à desservir. Comme dans le cas non orienté, nous allons résoudre le problème du postier rural en travaillant sur un graphe simplifié $G_S = (V_R, R \cup A_S)$ obtenu à partir de G_R. G_S se construit de la manière suivante:

1) Pour chaque paire de sommets $v_i, v_j \in V_R$, les arcs (v_i, v_j) et (v_j, v_i) sont insérés dans A_S. Le coût d'un arc $(v_i, v_j) \in A_S$ est égal à la longueur du plus court chemin reliant v_i à v_j dans G (les arcs de R gardent le coût qu'ils ont dans G).

2) Enlever de A_S tout arc (v_i, v_j) pour lequel il existe un sommet $v_k \in V_R$ tel que: $c_{ij} = c_{ik} + c_{kj}$.

3) Enlever de A_S tout arc (v_i, v_j) parallèle, de même direction et de coût égal à un arc de R.

A un circuit T_S solution du PPR dans G_S est associé un circuit T solution du PPR dans G. T est obtenu en remplaçant les arcs non desservis de T_S par le plus court chemin qui leur correspond dans G.

Christofides [Chr86] a observé que certains arcs de G_S appartenant à A_S peuvent être transférés dans R sans que la valeur d'une solution optimale du PPR ne soit modifiée. Pour qu'un arc (v_i, v_j) de A_S puisse passer dans R, au moins une des conditions suivantes doit être satisfaite:

i) (v_i, v_j) est le seul arc sortant du sommet v_i dans G_S;

ii) (v_i, v_j) est le seul arc entrant en v_j dans G_S;

iii) il existe une partition de l'ensemble F des composantes connexes de G_R en deux sous-ensembles F_1 et F_2 telle que (v_i, v_j) est le seul arc allant de F_1 vers F_2.

L'exemple ci-dessous illustre la construction de G_S (figures 1.17 et 1.18), puis le transfert de certains arcs de A_S dans R (figure 1.19). Les arcs en gras sont les arcs à desservir et les valeurs sur les arêtes correspondent à leur coût.

Exemple:

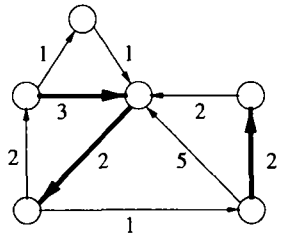


FIG. 1.17 -

Graphe initial G.

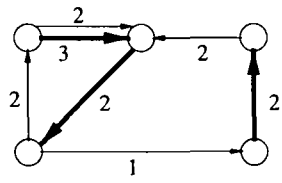


FIG. 1.18 -

Graphe simplifié G_S.

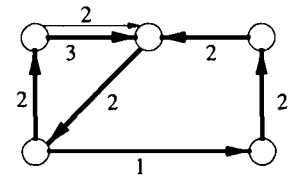


FIG. 1.19 -

Graphe G_S après transfert.

Nous pouvons remarquer que dans cet exemple, en effectuant le transfert de certains arcs de A_S dans R, le graphe $G_R = (V_R, R)$ devient connexe. Il est donc possible de résoudre polynomialement le problème ci-dessus, chose qui n'est pas évidente si nous considérons le graphe G sans simplification puisqu'alors G_R n'est pas connexe. A partir de maintenant, nous supposerons que le graphe G sur lequel nous travaillons est un graphe simplifié dont tous les arcs qu'il était possible de transférer dans R se trouvent effectivement dans R.

Comme dans le cas non orienté, nous allons traiter deux cas, celui où G_R est connexe et celui où il ne l'est pas. Lorsque G_R est connexe, le PPR peut se résoudre polynomialement à l'aide d'un algorithme très similaire à PPR_NON_ORIENTE_CONNEXE (cf. section 1.2.2.1). Cet algorithme est présenté à la page suivante.

Algorithme: PPR_ORIENTE_CONNEXE

ENTREE: Un graphe orienté $G = (V, A)$ fortement connexe et un ensemble $R \subseteq A$ tel que $G_R = (V, R)$ est connexe.

SORTIE: Un circuit T solution optimale du PPR dans G.

Etape 1. Poser $G' = G_R$. Si G' est symétrique aller à l'étape 5.

Etape 2. Calculer la valeur $f_i = d^-_{G'}(i) - d^+_{G'}(i)$ pour chaque sommet $v_i \in G'$. Soit $I = \{v_i \in V | f_i > 0\}$ et $J = \{v_j \in V | f_j < 0\}$.

Etape 3. Résoudre le problème de transport dans lequel chaque sommet $v_i \in I$ a une disponibilité de f_i, chaque sommet $v_j \in J$ possède une demande de $|f_j|$ et le coût d'un arc $(v_i, v_j) \in I \times J$ est égal à la longueur du plus court chemin SP_{ij} reliant v_i à v_j dans G.

Etape 4. Soit x_{ij} la quantité transportée sur l'arc (v_i, v_j) dans la solution optimale du problème de transport. Pour chaque arc $(v_i, v_j) \in I \times J$, ajouter à G' x_{ij} copies de chaque arc appartenant à SP_{ij}.

Etape 5. Déterminer un circuit eulérien T dans G' en utilisant CIRCUIT_EULERIEN (cf. 1.2.1.2). T est une solution optimale du PPR dans G.

Lorsque G_R n'est pas connexe, le PPR est NP-dur. Nous présentons ici deux approches qui ont été proposées pour la résolution du PPR.

La première approche, que nous appellerons approche S&C, consiste à partir du graphe G_R, à le transformer en un graphe symétrique G', puis à rendre G' fortement connexe. Cette approche a été proposée par Ball et Magazine [Bal88], leur algorithme est présenté à la page suivante.

La deuxième approche, que nous nommerons approche C&S, a, comme dans l'approche S&C, le graphe G_R comme point de départ. Ce graphe est tout d'abord rendu connexe, puis symétrique. Cette approche a été étudiée par plusieurs auteurs ([Bal88], [Chr86], etc.). Nous présenterons l'algorithme décrit dans [Chr86].

Algorithme: S&C_PPR_ORIENTE

ENTREE: Un graphe orienté $G = (V, A)$ fortement connexe et un ensemble $R \subseteq A$.

SORTIE: Un circuit T solution (pas forcément optimale) du PPR dans G.

Etape 1. Poser $G' = G_R$. Si G' est symétrique aller à l'étape 3.

Etape 2. Appliquer les étapes 2, 3 et 4 de PPR_ORIENTE_CONNEXE de façon à transformer G' en un graphe symétrique.

Etape 3. Si G' est connexe aller à l'étape 4.
Soient $C_1, ..., C_c$ les composantes connexes de G' et soit sp_{ij} la longueur d'un plus court chemin reliant v_i à v_j dans G.
Construire un graphe complet G'' non orienté dont les sommets sont $\{v''_1, v''_2, ..., v''_c\}$. Associer à chaque arête (v''_p, v''_q) de G'' un coût $c''_{pq} = min\{sp_{ij} + sp_{ji} | v_i \in C_p \text{ et } v_j \in C_q\}$.
Déterminer un arbre maximal de coût minimum dans G''.
Pour chaque arête (v''_p, v''_q) de l'arbre optimal, considérer les sommets v_i de C_p et v_j de C_q tels que $c''_{pq} = sp_{ij} + sp_{ji}$ et ajouter à G' tous les arcs se trouvant dans le plus court chemin reliant v_i à v_j et dans celui reliant v_j à v_i dans G.

Etape 4. Construire un circuit eulérien T dans G' en utilisant CIRCUIT_EULERIEN (cf. section 1.2.1.2). T est une solution du PPR dans G.

L'exemple ci-dessous illustre le fonctionnement de l'approche S&C que nous venons de décrire.

Exemple:

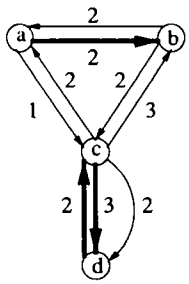


FIG. 1.20 – *Graphe initial G.*

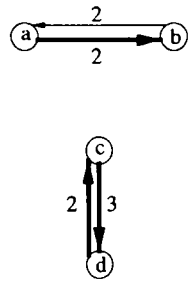


FIG. 1.21 – *Graphe symétrique à la fin de l'étape 2.*

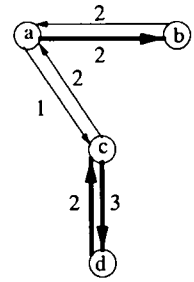


FIG. 1.22 – *Graphe symétrique et fortement connexe à la fin de l'étape 3.*

L'algorithme ci-dessous correspond à l'approche C&S proposée dans [Chr86].

Algorithme: C&S_PPR_ORIENTE

ENTREE: Un graphe orienté $G = (V, A)$ fortement connexe et un ensemble $R \subseteq A$.

SORTIE: Un circuit T solution (pas forcément optimale) du PPR dans G.

Etape 1. Poser $G' = G_R$. Si G' est connexe aller à l'étape 3.

Etape 2. Soient $C_1, ..., C_c$ les composantes connexes de G'.
Construire un graphe complet G'' orienté dont les sommets sont $\{v''_1, v''_2, ..., v''_c\}$. Associer à chaque arc (v''_p, v''_q) de G'' un coût $c''_{pq} = min\{c_{ij} | v_i \in C_p \text{ et } v_j \in C_q\}$.
Déterminer une arborescence maximale de coût minimum dans G'' dont la racine est un sommet choisi arbitrairement. (pour une telle construction voir par exemple [Edm67])
Pour chaque arc (v''_p, v''_q) de l'arborescence optimale, considérer les sommets v_i de C_p et v_j de C_q tels que $c''_{pq} = c_{ij}$ et ajouter à G' l'arc (v_i, v_j).

Etape 3. Appliquer les étapes 2, 3 et 4 de PPR_ORIENTE_CONNEXE pour transformer G' en un graphe symétrique.

Etape 4. Construire un circuit eulérien T dans G' en utilisant CIRCUIT_EULERIEN (cf. section 1.2.1.2). T est une solution du PPR dans G.

L'algorithme que nous venons de décrire peut être répété en choisissant à chaque fois un sommet différent de G'' comme racine de l'arborescence construite à l'étape 2. L'exemple qui suit illustre le fonctionnement de cet algorithme.

Exemple:

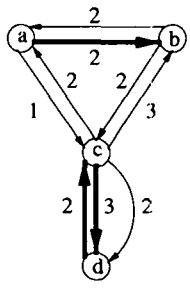


FIG. 1.23 -

Graphe initial G.

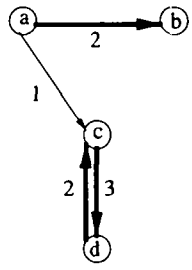


FIG. 1.24 -

Graphe connexe à la fin de l'étape 2.

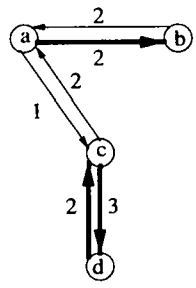


FIG. 1.25 -

Graphe symétrique et fortement connexe à la fin de l'étape 3.

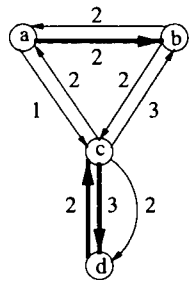


FIG. 1.26 -

Graphe initial G.

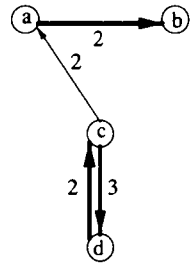


FIG. 1.27 -

Graphe connexe à la fin de l'étape 2.

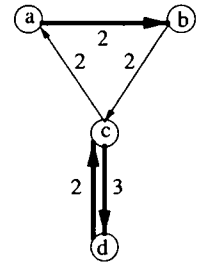


FIG. 1.28 -

Graphe symétrique et fortement connexe à la fin de l'étape 3.

Dans l'exemple ci-dessus, le graphe G'' ne comporte que deux sommets. Une première solution est obtenue (figure 1.25) en prenant pour racine de l'arborescence le sommet de G'' associé à la composante connexe de G_R composée de (a, b) (figure 1.24). Une meilleure solution (figure 1.28) est obtenue en utilisant comme racine de l'arborescence le sommet de G'' associé à la composante connexe de G_R contenant (c, d) et (d, c) (figure 1.27).

Comme mentionné dans [Chr86], toute solution de PPR peut être améliorée en remplaçant deux arcs facultatifs consécutifs (v_i, v_j) et (v_j, v_k) dans la solution par un arc (v_i, v_k) si $c_{ik} < c_{ij} + c_{jk}$. En appliquant ce procédé, nous pouvons par exemple passer de la solution de la figure 1.25 à celle de la figure 1.28 en remplaçant deux arcs facultatifs consécutifs de coût 2 et 1 par un arc de coût 2.

1.2.2.3 Le PPR dans les graphes mixtes

Considérons le graphe mixte $G = (V, E \cup A)$ fortement connexe et un ensemble $R \subseteq E \cup A$ d'arcs et d'arêtes à desservir. Nous pouvons supposer que chaque sommet de V est incident à un client et que chaque arête de E se trouve dans R. Si cela n'est pas le cas, le graphe initial G peut être simplifié de manière à satisfaire cette hypothèse. Cette simplification est similaire à celles effectuées dans les sections 1.2.2.1 et 1.2.2.2. Nous allons tout de même la décrire et l'illustrer par un exemple.

Le graphe simplifié $G_S = (V_R, R \cup A_S)$ est obtenu à partir de G_R de la manière suivante:

1) Insérer dans A_S deux arcs (v_i, v_j) et (v_j, v_i) pour chaque paire de sommets v_i, v_j de V_R. Le coût d'un arc (v_i, v_j) de A_S est égal à la longueur du plus court chemin reliant v_i à v_j dans G. (le coût des arcs et arêtes de R ne change pas).

2) Enlever de A_S tout arc (v_i, v_j) pour lequel il existe un sommet $v_k \in V_R$ tel que: $c_{ij} = c_{ik} + c_{kj}$.

3) Enlever de A_S tout arc (v_i, v_j) parallèle à une arête à desservir et ayant le même coût que celle-ci.

4) Enlever de A_S tout arc (v_i, v_j) parallèle à un arc à desservir et ayant même coût et même direction que celui-ci.

Nous remarquons qu'avec cette construction toutes les arêtes de G_S sont dans R. L'exemple ci-dessous illustre la construction du graphe simplifié.

Exemple:

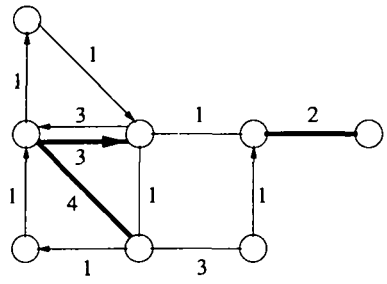


FIG. 1.29 – *Graphe de départ G.*

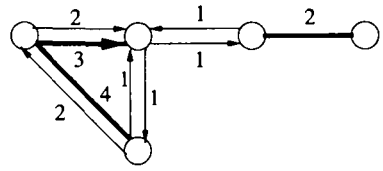


FIG. 1.30 – *Graphe simplifié G_S.*

A partir de maintenant, nous supposons que le graphe $G = (V, E \cup A)$ sur lequel nous travaillons est déjà un graphe simplifié. Cela signifie que $E \subset R$. Nous noterons par $A_R = A \cap R$ l'ensemble des arcs à desservir dans G.

Très peu d'articles traitent du PPR dans les graphes mixtes. Une des raisons est que ce problème peut être résolu en utilisant des méthodes semblables à celles développées pour les cas orienté et non orienté. En effet une solution peut être construite en donnant une orientation à chaque arête de R, en connectant les composantes connexes de G_R, en ajoutant quelques arcs pour obtenir un graphe symétrique et finalement en cherchant un circuit eulérien dans ce dernier graphe. Une stratégie de ce type a été proposée par Corberán et al. [Cor97]. L'idée est d'orienter les arêtes à desservir de telle sorte que le degré sortant de chaque sommet soit "presque" égal à son degré entrant.

Algorithme: PPR_MIXTE

ENTREE: Un graphe mixte $G = (V, E \cup A)$ fortement connexe et un ensemble $R \subseteq E \cup A$.

SORTIE: Un circuit T solution (pas forcément optimale) du PPR dans G.

Etape 1. *(Orientation des arêtes)*
Construire un graphe $G' = (V, A_E \cup A_R)$ en orientant et insérant dans A_E successivement chaque arête à desservir (v_i, v_j) de la manière suivante (initialement $A_E = \emptyset$):

Calculer $f'_i = d^-_{G'}(i) - d^+_{G'}(i)$ et $f'_j = d^-_{G'}(j) - d^+_{G'}(j)$.
Orienter (v_i, v_j) de v_i vers v_j si $f'_i \geq f'_j$ et de v_j vers v_i sinon.
Insérer l'arête orientée dans A_E.

Algorithme: PPR_MIXTE (SUITE)

Etape 2. *(Construction d'un graphe connexe G'.)*
Soient $C_1, ..., C_c$ les composantes connexes de G_R.
Soient $\delta_i^- = min\{c_{ji} | (v_j, v_i) \in E \cup A\}$ et $\delta_i^+ = min\{c_{ij} | (v_i, v_j) \in E \cup A\}$.
Attribuer un coût c'_{ij} à chaque arc (v_i, v_j) reliant deux composantes connexes de G_R tel que:

$$c'_{ij} = \begin{cases} c_{ij} + \delta_i^- + \delta_j^+ & \text{si } f'_i \leq 0 \text{ et } f'_j \geq 0 \\ c_{ij} - \delta_i^+ + \delta_j^+ & \text{si } f'_i > 0 \text{ et } f'_j \geq 0 \\ c_{ij} + \delta_i^- - \delta_j^- & \text{si } f'_i \leq 0 \text{ et } f'_j < 0 \\ c_{ij} - \delta_i^+ - \delta_j^- & \text{si } f'_i > 0 \text{ et } f'_j < 0 \end{cases}$$

Construire un graphe complet non orienté G'' dont l'ensemble des sommets est $\{v''_1, ..., v''_c\}$ et dont le coût d'une arête (v''_p, v''_q) est égal à $c''_{pq} = min\{c'_{ij} | v_i \in C_p \text{ et } v_j \in C_q, \text{ ou } v_i \in C_q \text{ et } v_j \in C_p\}$.
Déterminer un arbre maximal de coût minimum dans G'' et introduire les arcs correspondants dans G'. Si G' est symétrique aller à l'étape 5.

Etape 3. *(Transformation de G' en graphe symétrique)*
Construire un graphe $H = (V, A_E \cup A'_E \cup A''_E \cup A)$ tel que pour chaque arête $(v_i, v_j) \in E$, $A_E \cup A'_E$ contienne des arcs de direction opposée et de coût c_{ij}, et tel que A''_E contienne un arc (v_i, v_j) de coût nul pour tout arc (v_j, v_i) se trouvant dans A_E.
Attribuer une capacité infinie aux arcs de $A_E \cup A'_E \cup A$ et une capacité de 2 aux arcs de A''_E.
Calculer les valeurs $f'_i = d^-_{G'}(i) - d^+_{G'}(i)$ pour chaque sommet v_i de G'.
Attribuer une disponibilité de f'_i à chaque sommet v_i de H tel que $f'_i > 0$ et une demande de $|f'_i|$ à chaque sommet v_i tel que $f'_i < 0$.
Trouver un flot compatible de coût minimum dans H satisfaisant les demandes. Soit x_{ij} la valeur du flot optimum sur l'arc (v_i, v_j). Pour tout arc $(v_j, v_i) \in A''_E$ inverser l'orientation de $(v_i, v_j) \in G'$ si $x_{ji} = 2$, supprimer cette orientation si $x_{ji} = 1$, et la laisser inchangée si $x_{ji} = 0$.
Augmenter G' en ajoutant x_{ij} copies de chaque arc $(v_i, v_j) \in A_E \cup A'_E \cup A$.
Si G' est orienté aller à l'étape 5.

Etape 4. Soit E' l'ensemble des arêtes de G'. Déterminer l'ensemble V_0 des sommets de G' incidents à un nombre impair d'arêtes de E'. Construire un graphe complet non orienté $G_0 = (V_0, E_0)$. Soit sp_{ij} la longueur d'un plus court chemin utilisant les arcs de $A_E \cup A'_E \cup A$ et reliant v_i à v_j dans G'. Attribuer un coût $c^0_{ij} = min\{sp_{ij}, sp_{ji}\}$ à chaque arête $(v_i, v_j) \in E_0$.
Trouver un couplage parfait M de coût minimum dans G_0 et orienter chaque cycle induit par $E' \cup M$ de telle sorte que le circuit correspondant dans G soit de longueur minimum. Introduire ce circuit dans G'.

Etape 5. Déterminer un circuit eulérien T dans G' en utilisant CIRCUIT_EULERIEN. T correspond à une solution du PPR dans G.

Nous allons illustrer, étape par étape le fonctionnement de cet algorithme à l'aide de l'exemple ci-dessous.

Exemple:

Etape 1

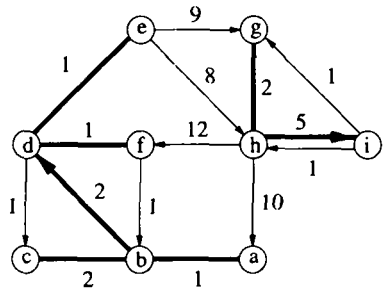


FIG. 1.31 – *Graphe de départ G.*

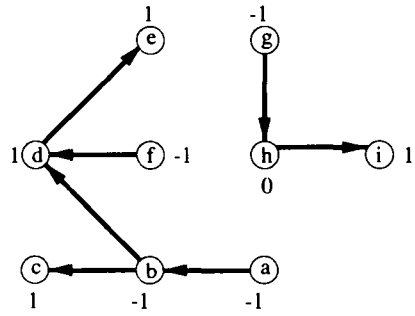


FIG. 1.32 – *Graphe G'.*

La figure 1.31 représente le graphe G initial. Le graphe G' obtenu après l'étape 1 est représenté dans la figure 1.32. Il est obtenu en orientant successivement les arêtes (a, b), (b, c), (d, e), (f, d) et (g, h). Les valeurs situées au-dessus des sommets de G' correspondent aux valeurs des f'_i à la fin de l'étape 1.

Etape 2

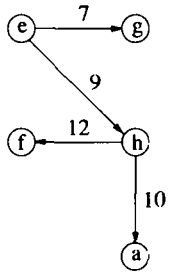


FIG. 1.33 –

Arcs connectant les composantes connexes de G_R.

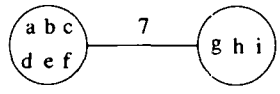


FIG. 1.34 –

Graphe complet G''.

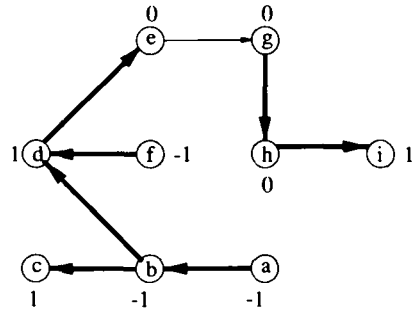


FIG. 1.35 –

Graphe G' obtenu à la fin de l'étape 2.

La figure 1.33 représente les arcs qui lient entre elles les différentes composantes connexes de G_R. Les valeurs situées sur ces arcs correspondent aux coûts c'_{ij} définis à l'étape 2. G'' est représenté par la figure 1.34. Ce graphe est composé de deux sommets uniquement. L'arbre maximal de coût minimum dans G'' correspond à l'arc (e, g) de G. C'est cet arc qui est rajouté à G' (voir figure 1.35). Les valeurs f'_e et f'_g sont mises à jour dans G'.

Le coût c'_{ij} d'un arc (v_i, v_j) prend en compte le fait qu'en connectant deux composantes connexes de G' à l'aide de (v_i, v_j), les degrés entrant et sortant des sommets v_i et v_j dans G' vont être modifiés. Si ces modifications accroissent la dissymétrie de G', le coût initial c_{ij} de (v_i, v_j) est pénalisé et c'_{ij} sera supérieur à c_{ij}. Dans le cas contraire, c'_{ij} est inférieur à c_{ij}. Ce deuxième cas se produit avec l'arc (e, g) de notre exemple. En utilisant cet arc pour connecter les deux composantes connexes de G' (figure 1.32), les sommets e et g deviennent symétriques dans G' (figure 1.35). Il faut donc sortir une fois de moins du sommet e d'où un gain d'au moins $\delta_e^+ = 1$ ($=c_{ed}$ dans G). De même, il faut entrer une fois de moins en g d'où un gain de $\delta_g^- = 1$ ($=c_{ig}$ dans G). Le fait d'utiliser l'arc (e, g) pour connecter les deux composantes de G' nous permet d'éviter de rajouter dans G' (à l'étape 3) deux arcs de coût au minimum 1. Nous avons ainsi $c'_{eg} = c_{eg} - 1 - 1 = 9 - 1 - 1 = 7$.

Etape 3 et Etape 4

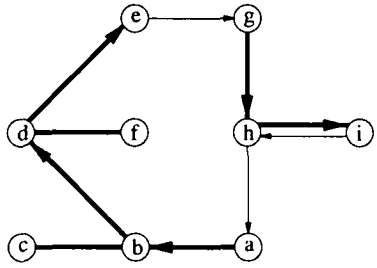


FIG. 1.36 -

Graphe symétrique G' à la fin de l'étape 3.

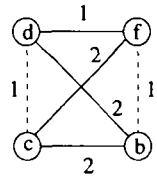


FIG. 1.37 -

Graphe complet non orienté G_0.

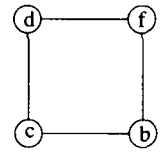


FIG. 1.38 -

$E' \cup M$.

La figure 1.36 représente le graphe G' obtenu à la fin de l'étape 3. Ce graphe contient encore deux arêtes à desservir qui ne sont pas orientées. La figure 1.37 représente le graphe G_0 dans lequel un couplage parfait M de coût minimum est recherché. Les arêtes de ce couplage sont en pointillés. La figure 1.38 représente l'union des arêtes de G' et de celles du couplage.

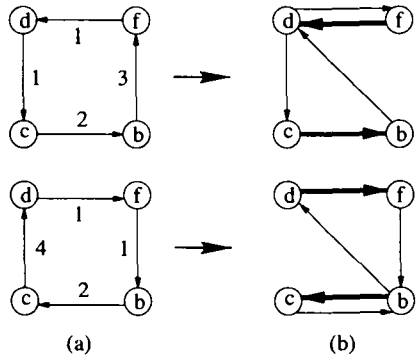


FIG. 1.39 -

Les deux orientations possibles du cycle induit par $E' \cup M$.

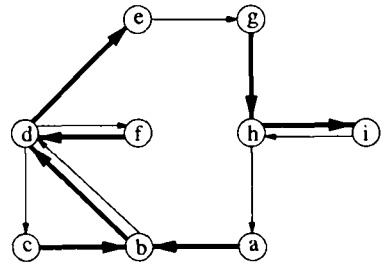


FIG. 1.40 -

Le graphe final G' symétrique connexe et orienté.

La figure 1.39 (a) représente les deux orientations possibles du cycle induit par $E' \cup M$ dans le graphe G_0. La figure 1.39 (b) représente les deux orientations correspondantes dans G. Le meilleur circuit issu de ces orientations est ajouté à G' pour obtenir un graphe symétrique connexe et orienté (figure 1.40).

Remarque:

Nous n'avons pas précisé l'ordre dans lequel les arêtes de E doivent être orientées à l'étape 1. Corberán et al. [Cor97] suggèrent d'utiliser un ordre particulier, ils proposent également quelques variations sur la construction de l'arbre maximal de coût minimum de l'étape 2. Nous n'avons pas présenté ici ces éléments mineurs afin de ne pas alourdir une description déjà fort longue de leur algorithme. Le lecteur désireux d'en savoir plus pourra se référer à [Cor97].

1.2.2.4 Le "Stacker Crane Problem" (SCP)

Le "Stacker Crane Problem" est défini dans un graphe $G = (V, E)$ non orienté. Soit $R \subseteq E$ l'ensemble des arêtes à desservir. La différence entre le SCP et le PPR réside dans le fait que dans le SCP la direction dans laquelle les arêtes de R sont desservies est imposée alors que pour le PPR aucun sens de service n'est prescrit.

Le SCP peut être vu comme un PPR dans un graphe mixte $G' = (V, E \cup A)$ où seuls les arcs sont à desservir (figure 1.42). Le graphe G' se construit en associant à chaque arête de $E \cap R$ un arc parallèle, de même coût que l'arête et dont la direction correspond au sens de service qui lui est imposé.
Le SCP peut également être vu comme un PPR dans un graphe orienté $G'' = (V, A'')$ (figure 1.43). Pour cela il suffit de remplacer les arêtes de E par deux arcs de direction opposée et de même coût que l'arête. Les arcs à desservir sont alors ceux qui correspondent aux arêtes de R et dont la direction est celle prescrite pour le service des arêtes.

Les figures ci-dessous illustrent la transformation d'un SCP (figure 1.41) en PPR mixte (figure 1.42) ou PPR orienté (figure 1.43). Les arêtes ou arcs à desservir sont en gras. Les flèches au-dessus des clients de G indiquent la direction imposée lors de leur service.

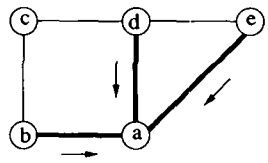


FIG. 1.41 – *Graphe de départ non orienté G.*

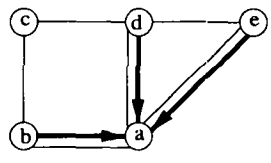


FIG. 1.42 – *Graphe mixte G'.*

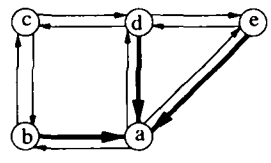


FIG. 1.43 – *Graphe orienté G''.*

Un SCP pouvant se transformer en PPR, les algorithmes décrits aux sections 1.2.2.2 et 1.2.2.3 peuvent être utilisés pour obtenir des solutions. Des méthodes spécifiques ont cependant été développées pour la résolution du SCP. Frederickson et al. [Fre78] ont proposé plusieurs algorithmes spécialement conçus pour le SCP. Avant de décrire ces heuristiques, nous allons tout d'abord voir comment simplifier le graphe initial G pour obtenir un graphe $G_S = (V_R, E_S \cup A)$ sur lequel nous allons travailler.

Le graphe G_S se construit à partir de G_R de la manière suivante:

1) Pour chaque paire de sommets $v_i, v_j \in V_R$ insérer dans E_S une arête (v_i, v_j) dont le coût est égal à la longueur d'une plus courte chaîne reliant v_i à v_j dans G.

2) Les arcs A correspondent aux arêtes de G_R orientées selon le sens de parcours imposé pour leur service.

A partir de maintenant, nous supposons que le graphe G à traiter est déjà un graphe simplifié. G est donc un graphe mixte et possède les propriétés suivantes:

i) Chaque arc doit être desservi et est parallèle à une arête de coût inférieur ou égal au sien.

ii) Chaque paire de sommets de G est reliée par une arête.

iii) Les coûts définis sur les arêtes satisfont l'inégalité triangulaire.

Les deux figures 1.44 et 1.45 illustrent la simplification pouvant être faite à partir du graphe de départ G.

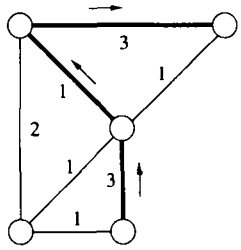


FIG. 1.44 -

Graphe initial G.

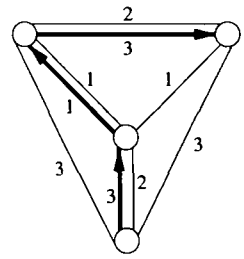


FIG. 1.45 -

Graphe simplifié G_S.

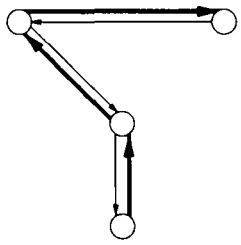


FIG. 1.46 -

Une tournée dans G_S.

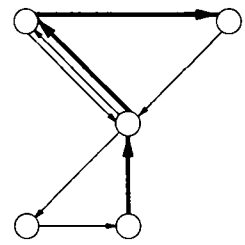


FIG. 1.47 -

La tournée correspondante dans G.

Le premier algorithme que nous présentons ici est très brièvement décrit dans [Fre78].

Algorithme: SCP_SIMPLE

ENTREE: Un graphe mixte $G = (V, E \cup A)$ où $A = R$ et $V = V_R$.

SORTIE: Un circuit T solution (pas forcément optimale) du SCP dans G.

Etape 1. Poser $G' = (V, A \cup A')$, où pour chaque arc (v_i, v_j) appartenant à A correspond un arc (v_j, v_i) dans A'.

Etape 2. Soient $C_1, ..., C_c$ les composantes connexes de G'.
Construire un graphe complet non orienté G'' dont l'ensemble des sommets est $\{v''_1, ..., v''_c\}$ et dont le coût associé à une arête (v''_p, v''_q) est égal au coût minimum d'une arête de G reliant C_p à C_q.
Trouver un arbre maximal de coût minimum dans G''. Soit S l'ensemble des arêtes de G correspondant aux arêtes de l'arbre optimal.
Pour chaque arête (v_i, v_j) de S ajouter dans G' deux arcs (v_i, v_j) et (v_j, v_i) de direction opposée.

Etape 3. Construire un circuit eulérien T dans G' en utilisant CIRCUIT_EULERIEN (voir 1.2.1.2). T est une solution du SCP dans G.

L'exemple ci-dessous illustre le fonctionnement de SCP_SIMPLE.

Exemple:

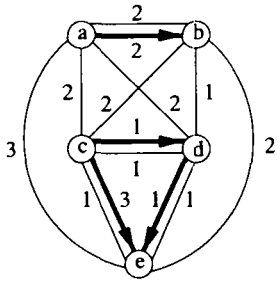


FIG. 1.48 -

Graphe initial G.

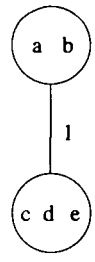


FIG. 1.49 -

Graphe complet non orienté G''.

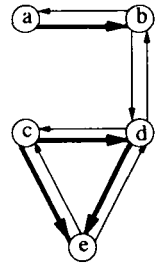


FIG. 1.50 -

Un circuit T couvrant A.

Les solutions fournies par SCP_SIMPLE ont une valeur qui est toujours inférieure ou égale à deux fois celle de la solution optimale. En effet, si nous notons C_A et C_S la longueur totale des arcs de A et des arêtes de S, SCP_SIMPLE trouve une solution dont le coût vaut $2C_A + 2C_S$. Soit C^* la valeur de la solution optimale. Nous avons $C_S \leq C^* - C_A$ et donc $2C_A + 2C_S \leq 2C_A + 2C^* - 2C_A = 2C^*$. Il existe des exemples pour lesquels cette borne sur la valeur des solutions de SCP_SIMPLE est atteinte.

Frederickson et al. ont proposé deux algorithmes plus élaborés dont l'utilisation combinée produit toujours des solutions de coût inférieur ou égal à $\frac{9}{5}$ fois celui de l'optimum. Nous allons décrire ces deux heuristiques. Leur nom correspond à celui utilisé dans [Fre78].

Algorithme: LARGE_ARCS

ENTREE: Un graphe mixte $G = (V, E \cup A)$ où $A = R$ et $V = V_R$.

SORTIE: Un circuit T solution (pas forcément optimale) du SCP dans G.

Etape 1. Soit P et O les ensembles contenant la pointe et l'origine des arcs de A. Construire un graphe biparti complet $B = (P \cup O, E_B)$ où chaque arête (v_i, v_j) de E_B a un coût identique à celui de l'arête (v_i, v_j) de E. Déterminer un couplage parfait M de coût minimum dans B. Construire un graphe $G' = (V, A \cup A_B)$ où A_B est obtenu en orientant les arêtes de M de P vers O.

Etape 2. Appliquer à G' les étapes 2 et 3 de SCP_SIMPLE de manière à déterminer un circuit T solution du SCP dans G.

Exemple:

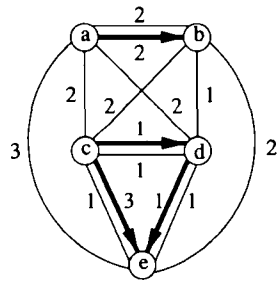


FIG. 1.51 -

Graphe initial G.

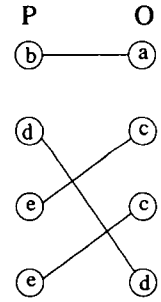


FIG. 1.52 -

Un couplage optimal dans B.

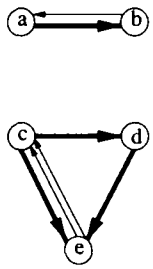


FIG. 1.53 -

Graphe G' à la fin de l'étape 1.

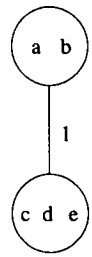


FIG. 1.54 -

Graphe complet non orienté G''.

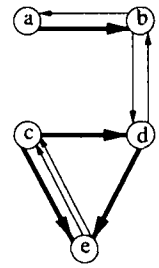


FIG. 1.55 -

Un circuit T couvrant A.

Etudions la qualité (par rapport à l'optimum) des solutions produites par LARGE_ARCS. Soit C_A, C_M et C_S la longueur totale des arcs de A, des arêtes de M et de celles de S, et soit C^* la valeur d'une solution optimale du SCP dans G. La solution obtenue en utilisant LARGE_ARCS vaut $C_A + C_M + 2C_S$. Les coûts C_M et C_S sont tous deux inférieurs ou égaux à $C^* - C_A$. Ainsi toute solution fournie par LARGE_ARCS aura une valeur inférieure ou égale à $3C^* - 2C_A$.

Nous présentons à présent le deuxième algorithme proposé dans [Fre78] pour la résolution du SCP.

Algorithme: SMALL_ARCS

ENTREE: Un graphe mixte $G = (V, E \cup A)$ où $A = R$ et $V = V_R$.

SORTIE: Un circuit T solution (pas forcément optimale) du SCP dans G.

Etape 0. Si les arcs de A ne sont pas disjoints par les sommets alors effectuer pour chaque sommet $v \in V$ adjacent à un ensemble $\{a_1, ..., a_k\}$ d'arcs $(k > 1)$ les opérations suivantes:

> Remplacer v par une clique K_v de taille k dont chaque arête a un coût nul.
> Remplacer chaque arc a_i $(i = 1, ..., k)$ par un arc incident au $i^{\text{ème}}$ sommet de K_v.
> Remplacer chaque arête de G incidente à v par une arête incidente à un sommet de K_v (n'importe lequel).

Tous les arcs sont maintenant disjoints par les sommets.

Etape 1. Construire un graphe complet non orienté G'' où chaque sommet correspond à un arc de A. Le coût d'une arête (v''_x, v''_y) de G'' est égal au coût minimum d'une arête reliant une extrémité de l'arc a_x à une extrémité de l'arc a_y dans G. Soit sc_{xy} la longueur d'une plus courte chaîne entre v''_x et v''_y dans G''.

Etape 2. Déterminer un arbre maximal de coût minimum S dans G'' en utilisant la fonction de distance sc.
Déterminer un couplage parfait M de coût minimum entre les sommets de degré impair dans S (toujours en utilisant la fonction de distance sc).
Poser $G' = (V, A)$ et ajouter à G' toutes les arêtes de G induites par $S \cup M$.

Etape 3. Soit A_{imp} le sous-ensemble de A dont les deux extrémités ont un degré impair dans G' et soit $A_{pair} = A \setminus A_{imp}$.
Pour chaque arc $(v_i, v_j) \in A_{imp}$ ajouter un arc (v_j, v_i) dans G'.
Remplacer chaque clique K_v définie à l'étape 0 par le sommet v correspondant.

Algorithme: SMALL_ARCS (SUITE)

Etape 4. Trouver un cycle eulérien T' dans G' (la direction des arcs est ignorée) en utilisant CYCLE_EULERIEN (voir 1.2.1.1).
Choisir une orientation de T' telle que la longueur totale des arcs de A_{pair} traversés en sens inverse soit inférieure ou égale à la moitié de la longueur totale des arcs de A_{pair}.
Orienter les arêtes de G' en fonction de l'orientation choisie pour T'.
Si un arc $(v_i, v_j) \in A_{pair}$ est traversé dans la mauvaise direction, ajouter deux arcs (v_i, v_j) et (v_j, v_i) à G'.

Etape 5. Déterminer un circuit eulérien T dans G' en utilisant CIRCUIT_EULERIEN. T est une solution du SCP dans G.

L'exemple ci-dessous traite le même problème que celui représenté aux figures 1.48 et 1.51.

Exemple:

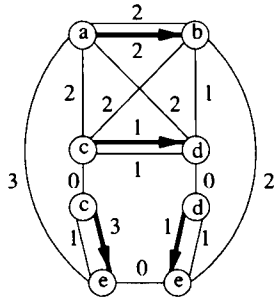


FIG. 1.56 -

Graphe G transformé à la fin de l'étape 0.

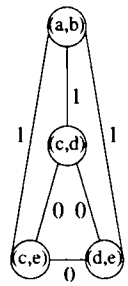


FIG. 1.57 -

Graphe complet G''.

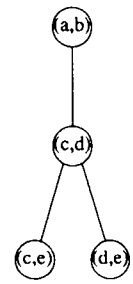


FIG. 1.58 -

Arbre S dans le graphe G''.

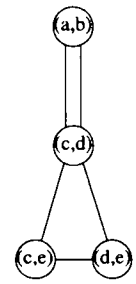


FIG. 1.59 -

S ∪ M dans le graphe G''.

Les trois figures ci-dessus illustrent les étapes 0, 1 et 2 de SMALL_ARCS. La figure 1.56 représente le graphe G obtenu à la fin de l'étape 0 après l'avoir transformé de manière à ce que les arcs de A soient disjoints par les sommets. Le graphe G'' obtenu à la fin de l'étape 1 est décrit dans la figure 1.57. Le nombre situé près d'une arête correspond à la longueur de la plus courte chaîne reliant ses extrémités dans G''. Cette longueur peut être différente du coût de l'arête. Le coût de l'arête reliant les sommets de G'' représentant les arêtes (a, b) et (c, e) du graphe G vaut 2. La longueur de la plus courte chaîne reliant ces deux sommets dans G'' vaut 1. L'arbre maximal de coût minium est représenté à la figure 1.58. Tous les sommets de cet arbre sont de degré impair. La figure 1.59 décrit la façon dont ces sommets sont couplés entre eux.

Les trois figures qui suivent illustrent les étapes 3, 4 et 5 de SMALL_ARCS.

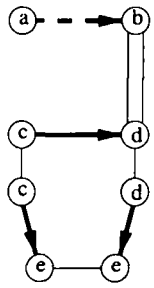


FIG. 1.60 -

Graphe G' après l'étape 2.

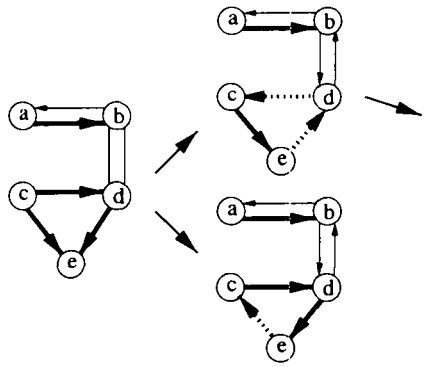


FIG. 1.61 -

Graphe G' après l'étape 3 et les deux orientations de T'.

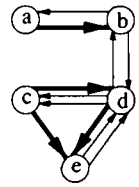


FIG. 1.62 -

Un circuit T solution du SCP dans G.

L'arc en traitillés de la figure 1.60 représente l'unique arc appartenant à A_{imp}. Les arcs en pointillés de la figure 1.61 correspondent aux arcs de A_{pair} traversés dans le mauvais sens lorsque T' est orienté. La meilleure orientation de T' est celle qui est représentée en haut dans la figure 1.61. En effet, la longueur totale des arcs de A_{pair} traversés dans la fausse direction est de 2 alors que la longueur totale de tous les arcs de A_{pair} vaut 5.

Comme nous l'avons fait pour LARGE_ARCS, nous allons étudier la qualité de la valeur des solutions fournies par SMALL_ARCS par rapport à celle de l'optimum. Soit C_A, C_{imp}, C_{pair}, C_M et C_S la longueur totale des arcs de A, de A_{imp}, de A_{pair}, des arêtes de M et de celles de S. Soit également C^* la valeur d'une solution optimale du SCP dans G. Le coût d'une solution produite par SMALL_ARCS est inférieur ou égal à $C_A + C_S + C_M + C_{imp} + 2\frac{1}{2}C_{pair}$. D'après Christofides [Chr76], nous avons que $C_S + C_M \leq \frac{3}{2}(C^* - C_A)$. Comme $C_{imp} + C_{pair} = C_A$, nous avons donc que la valeur de la solution donnée par SMALL_ARCS est toujours inférieure ou égale à $\frac{1}{2}C_A + \frac{3}{2}C^*$.

En comparant cette borne avec celle obtenue pour LARGE_ARCS ($= 3C^* - 2C_A$), nous constatons que SMALL_ARCS est plus efficace lorsque C_A est faible par rapport à C^* et que dans le cas contraire c'est LARGE_ARCS qu'il est préférable d'utiliser. Lorsque ces deux algorithmes sont combinés, c'est-à-dire lorsque nous retenons la meilleure des deux solutions fournies par SMALL_ARCS et LARGE_ARCS, nous sommes assurés que la valeur de cette solution est toujours inférieure ou égale à $\frac{9}{5}C^*$. En effet, en utilisant SMALL_ARCS si $C_A \leq \frac{3}{5}C^*$, nous obtenons une borne valant $\frac{1}{2}(\frac{3}{5}C^*) + \frac{3}{2}C^* = \frac{9}{5}C^*$. Lorsque $C_A \geq \frac{3}{5}C^*$, en utilisant LARGE_ARCS, nous obtenons une borne dont la valeur est $3C^* - 2\frac{3}{5}C^* = \frac{9}{5}C^*$. A notre connaissance il n'existe aucun exemple pour lequel cette borne est atteinte.

Pour en finir avec le SCP, notons encore qu'il peut être vu comme un cas particulier du problème du voyageur de commerce (PVC) dans un graphe orienté. A partir du graphe simplifié $G = (V, E \cup A)$ nous pouvons construire un graphe complet orienté G_{PVC} en associant à chaque arc (v_i, v_j) de A un sommet v'_{ij}. Le coût d'un arc (v'_{ij}, v'_{kh}) de G_{PVC}

est égal à la longueur d'une plus courte chaîne reliant v_j à v_k dans G. Toute solution du PVC dans G_{PVC} de valeur C_{PVC} peut se transformer en une solution du SCP dans G dont la valeur est $C_{PVC} + C_A$. Il est donc possible d'adapter les heuristiques développées pour le PVC à la résolution du SCP. Lukka et Salminen [Luk87] ont par exemple développé une heuristique pour le SCP basée sur l'insertion d'arcs qui est une adaptation de l'heuristique d'insertion pour le PVC proposée par Rosenkrantz et al. dans [Ros77].

L'exemple de la page suivante illustre la transformation du SCP dans G (figure 1.63) en un problème de voyageur de commerce dans le graphe orienté G_{PVC} (figure 1.64). A la tournée optimale du PVC dans G_{PVC} (figure 1.65) qui consiste à partir de (a, b) et à se rendre successivement à (d, e), (c, e), (c, d) avant de revenir à (a, b) est associée une solution optimale du SCP dans G (figure 1.66). Cette solution consiste à desservir l'arc (a, b), puis de b à rejoindre d en utilisant la plus courte chaîne possible dans G. Une fois en d, l'arc (d, e) est desservi. Il faut alors se rendre en c (toujours en utilisant la plus courte chaîne) avant de pouvoir desservir (c, e). Nous nous trouvons alors au sommet e à partir duquel il faut regagner c pour servir l'arc (c, d) et finalement on retourne au sommet a en utilisant l'arête (d, a) de G.

Exemple:

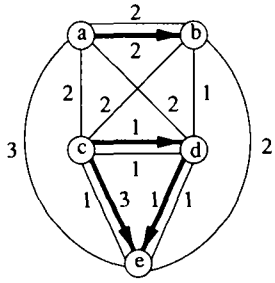


FIG. 1.63 -

Graphe initial G.

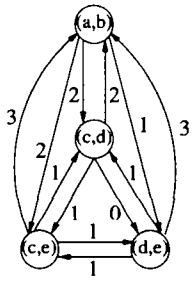


FIG. 1.64 -

Graphe G_{PVC}.

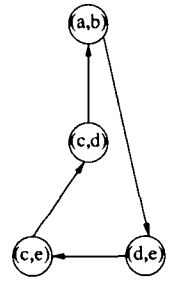


FIG. 1.65 -

Solution optimale du PVC dans G_{PVC}.

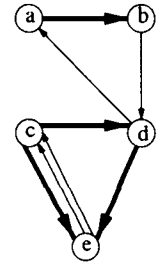


FIG. 1.66 -

Solution optimale du SCP dans G.

1.2.2.5 Autres outils algorithmiques de base

La plupart des algorithmes que nous avons présentés dans les sections 1.2.2.1 à 1.2.2.4 utilisent essentiellement les mêmes ingrédients de base. Un problème de couplage parfait de coût minimum, un problème de flot ou un problème de transport est généralement utilisé pour rendre le graphe initial G pair ou symétrique alors que la recherche d'un arbre (ou d'une arborescence) maximal de coût minimum sert le plus souvent à connecter entre elles les différentes composantes connexes du graphe G_R. De nouveaux outils sont apparus récemment qui peuvent s'appliquer aux problèmes de tournées sur les arcs sans contrainte de capacité [Her99]. Nous allons décrire ces outils en nous plaçant dans le cas non orienté. La plupart d'entre eux peuvent facilement être adaptés aux cas orienté et mixte.

Rappelons que nous travaillons dans un graphe $G = (V, E)$ non orienté, que R est l'ensemble des arêtes à desservir dans G et que le graphe $G_R = (V_R, R)$ est le sous-graphe

partiel de G induit par les sommets adjacents aux arêtes de R. Soit T une tournée et soit $R_T \subseteq R$ l'ensemble des arêtes desservies dans T. Une chaîne inutile dans T est une chaîne de T composée uniquement d'arêtes non desservies dans T. Deux types d'arêtes non desservies peuvent apparaître dans une tournée: les arêtes n'appartenant pas à R_T (arêtes facultatives) et les arêtes appartenant à R_T mais qui sont desservies ailleurs dans la tournée (ces arêtes apparaissent alors plusieurs fois dans T).

Plusieurs auteurs ont observé que lorsqu'une tournée contient une chaîne P constituée uniquement d'arêtes non traitées, il est possible de réduire la longueur de la tournée en remplaçant P par une plus courte chaîne reliant ses extrémités (voir par exemple [Fre78] et [Chr86]). Cette idée a été étendue par Hertz et al. [Her99] qui ont proposé une procédure qui tente de réduire la longueur d'une tournée donnée T en raccourcissant les chaînes inutiles de T. Le résultat de cette procédure est une tournée desservant les arêtes de R_T selon un ordre pouvant différer de celui utilisé dans T et dont la longueur totale est inférieure ou égale à celle de T. Cette procédure est décrite ci-dessous.

Algorithme: RACCOURCIR

ENTREE: Un cycle T dans $G = (V, E)$ desservant un ensemble $R_T \subseteq R$ d'arêtes.

SORTIE: Un cycle desservant les arêtes de R_T de coût au pire égal à celui de T.

Etape 1. Choisir une orientation de T et un sommet v_i de T. On considère v_i comme étant le premier sommet de T et on suppose que le service d'une arête de R_T se fait lors de sa dernière apparition dans T.

Etape 2. Déterminer le sommet v_j de T origine de la première arête desservie dans T rencontrée en partant de v_i. La chaîne P reliant v_i à v_j dans T est une chaîne inutile.

Etape 3. Soit Q la chaîne reliant v_j à v_i dans T ($Q = T \setminus P$).
Si toutes les arêtes de Q entrant dans un sommet identique à v_j sont des arêtes desservies, aller à l'étape 4.
Soit (v_k, v_j) une arête non desservie dans Q. Considérer le cycle C=$(v_j, ..., v_k, v_j)$, renverser son orientation dans T et aller à l'étape 2.

Etape 4. Si la longueur de la plus courte chaîne SC_{ij} reliant v_i à v_j dans G est inférieure à celle de P, remplacer P par SC_{ij}.

Etape 5. Répéter les étapes 2 à 4 en considérant les deux orientations possibles de T et tous les sommets de départ v_i de T jusqu'à ce qu'aucune amélioration ne puisse plus être obtenue.

Exemple:

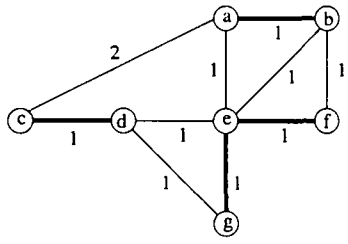


FIG. 1.67 -

Graphe initial G.

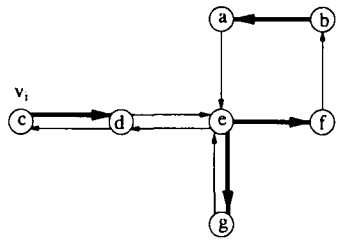


FIG. 1.68 -

Une tournée T dans G.

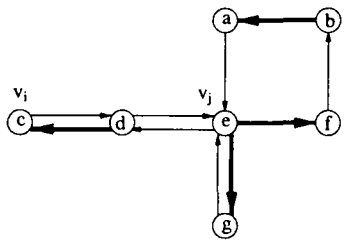


FIG. 1.69 -

Situation au début de l'étape 3.

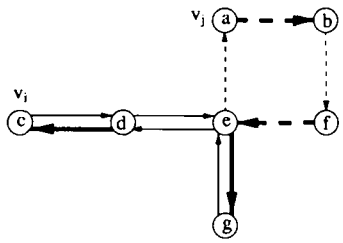


FIG. 1.70 -

Situation au début de l'étape 4.

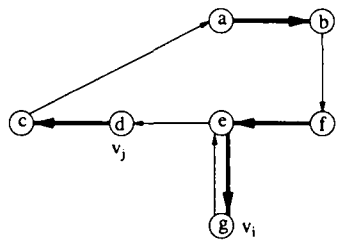


FIG. 1.71 -

Une tournée raccourcie dans G.

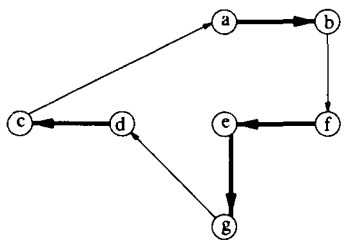


FIG. 1.72 -

Tournée finale fournie par RACCOURCIR.

L'exemple ci-dessus illustre le fonctionnement de la procédure RACCOURCIR. Le problème traité est un PPR. La figure 1.67 représente le graphe de départ. Une tournée $T = (c, d, e, f, b, a, e, g, e, d, c)$ desservant les arêtes de R est décrite dans la figure 1.68. Le sommet de départ v_i est le sommet c. Le service de l'arête (c, d) est déplacé en fin de tournée sur l'arête (d, c) qui devient une arête desservie (figure 1.69). Cela permet de rallonger la chaîne inutile partant de v_i. Nous avons alors $P = (c, d, e)$ et $Q = (e, f, b, a, e, g, e, d, c)$. L'arête (a, e) de Q est une arête non desservie entrant dans $v_j = e$. L'orientation du cycle $C = (e, f, b, a, e)$ est alors renversée (en traitillés sur la figure 1.70) ce qui permet d'étendre P jusqu'au sommet a. Nous avons donc $P = (c, d, e, a)$ et $Q = (a, b, f, e, g, e, d, c)$. La chaîne P est ensuite remplacée par l'arête (c, a) pour obtenir la tournée de la figure 1.71. Une nouvelle pair de sommets v_i, v_j est trouvée dans cette tournée qui une fois raccourcie donne la tournée finale de la figure 1.72.

La seconde procédure dont il est question dans cette section permet, étant donné une tournée T desservant un ensemble $R_T \subseteq R$ d'arêtes et une arête $(v_i, v_j) \in R_T$, de supprimer le service de l'arête (v_i, v_j) de T. La nouvelle tournée T' ainsi obtenue dessert alors les arêtes de $R_T \setminus (v_i, v_j)$. Cette procédure est décrite dans [Her99].

Algorithme: SUPPRIMER_CLIENT

ENTREE: Un cycle T dans $G = (V, E)$ desservant un ensemble $R_T \subseteq R$ d'arêtes et une arête $(v_i, v_j) \in R_T$.

SORTIE: Un cycle T' desservant les arêtes de $R_T \setminus (v_i, v_j)$.

Etape 1. Poser $R_T = R_T \setminus \{(v_i, v_j)\}$.

Etape 2. Appliquer RACCOURCIR à T pour obtenir une tournée T'.

Exemple:

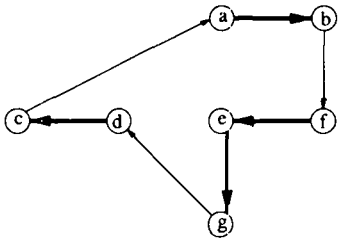


FIG. 1.73 -

Une tournée T couvrant $R_T = R$ dans G.

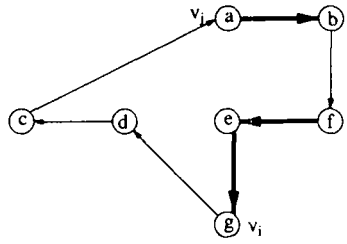


FIG. 1.74 -

L'arête (c, d) est ôtée de R.

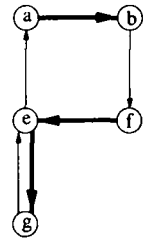


FIG. 1.75 -

Tournée finale T'.

Le problème traité dans l'exemple ci-dessus est identique à celui représenté par la figure 1.67. La figure 1.73 correspond à une tournée T desservant toutes les arêtes de R. En supprimant de R l'arête (c, d), la chaîne dans T d'extrémités $v_i = g$ et $v_j = a$ est une *chaîne inutile* (figure 1.74). La procédure RACCOURCIR remplace cette chaîne par la plus courte chaîne reliant v_i à v_j dans G pour former une nouvelle tournée T' desservant les arêtes de $R \setminus (c, d)$ (figure 1.75).

Hertz et al. ont également développé la procédure inverse de SUPPRIMER_CLIENT qui permet, étant donné une tournée T desservant les arêtes de $R_T \subset R$ et une arête $(v_i, v_j) \notin R_T$, d'insérer dans T l'arête (v_i, v_j) de R afin qu'elle puisse y être desservie. Dans la description de l'algorithme qui va suivre, SC_{ij} représentera une plus courte chaîne entre v_i et v_j dans G. La longueur de cette chaîne sera notée sc_{ij}.

Algorithme: AJOUTER_CLIENT

ENTREE: Un cycle T dans $G = (V, E)$ desservant un ensemble $R_T \subset R$ d'arêtes et une arête de R, $(v_i, v_j) \notin R_T$.

SORTIE: Un cycle T' desservant les arêtes de $R_T \cup (v_i, v_j)$.

Etape 1. Si aucun des sommets v_i et v_j ne sont présents dans T, trouver le sommet $v_k \in T$ minimisant la valeur $sc_{ki} + sc_{jk}$ et ajouter à T le cycle $SC_{ki} \cup \{(v_i, v_j)\} \cup SC_{jk}$.
Sinon, si seulement un des sommets v_i ou v_j se trouve dans T (disons v_i), ou si les deux sommets v_i et v_j sont dans T mais pas consécutivement, ajouter à T le cycle (v_i, v_j, v_i).

Etape 2. Poser $R_T = R_T \cup (v_i, v_j)$ et appliquer RACCOURCIR à T pour obtenir une tournée T' au moins aussi courte que T.

Exemple:

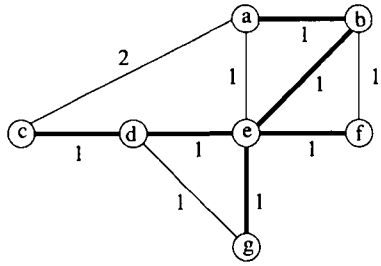


FIG. 1.76 -

Graphe initial G.

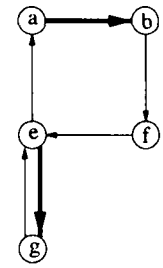


FIG. 1.77 -

Une cycle couvrant $R_T = \{(a, b), (e, g)\}$.

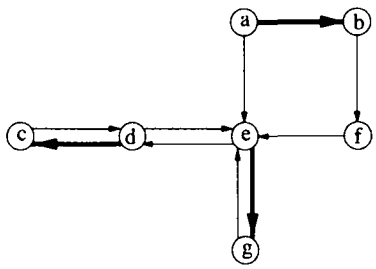


FIG. 1.78 -

(c, d) *est ajoutée à* R_T.

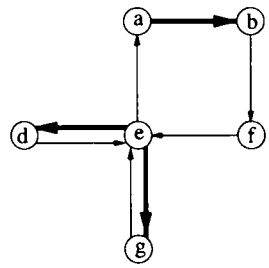


FIG. 1.79 -

(d, e) *est ajoutée à* R_T.

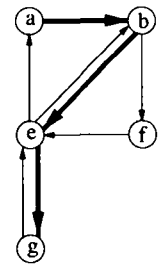


FIG. 1.80 -

(b, e) *est ajoutée à* R_T.

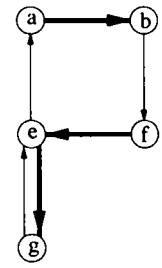


FIG. 1.81 -

(e, f) *est ajoutée à* R_T.

L'exemple de la page précédente illustre les 4 types d'insertions possibles d'une arête $(v_i, v_j) \notin R_T$ dans T:

1) si v_i et v_j n'appartiennent pas à T (figure 1.78)

2) si v_i est dans T mais pas v_j (figure 1.79)

3) si v_i et v_j sont dans T mais pas consécutivement (figure 1.80)

4) si v_i et v_j sont dans T consécutivement (figure 1.81)

Hertz et al. proposent d'utiliser AJOUTER_CLIENT comme procédure de base d'une méthode constructive permettant d'obtenir une solution du PPR dans $G = (V, E)$. Cette méthode est décrite ci-dessous.

Algorithme: PPR_CONSTRUCTIF

ENTREE: Un graphe non orienté $G = (V, E)$ et un sous-ensemble $R \subseteq E$.

SORTIE: Un cycle T solution du PPR dans G.

Etape 1. Choisir une arête $(v_i, v_j) \in R$. Poser $T = (v_i, v_j, v_i)$ et $R_T = \{(v_i, v_j)\}$.

Etape 2. Si $R = R_T$ STOP. Sinon choisir une arête $(v_i, v_j) \in R \setminus R_T$ et construire un cycle T' couvrant $R_T \cup \{(v_i, v_j)\}$ en utilisant AJOUTER_CLIENT. Poser $R_T = R_T \cup \{(v_i, v_j)\}$, $T = T'$ et répéter l'étape 2.

Des méthodes de post-optimisation sont également décrites dans [Her99] qui utilisent RACCOURCIR, AJOUTER_CLIENT et SUPPRIMER_CLIENT. La première d'entre elles est analogue à l'algorithme "Unstringing-Stringing" (US) utilisé pour le PVC dans [Gen92]. Etant donné une tournée T desservant les arêtes de $R_T \subseteq R$, cette procédure enlève de T et rajoute dans T successivement toutes les arêtes appartenant à R_T.

Algorithme: SUPPRIMER&AJOUTER_CLIENT

ENTREE: Un cycle T dans $G = (V, E)$ desservant un ensemble $R_T \subseteq R$ d'arêtes.

SORTIE: Un cycle desservant les arêtes de R_T de coût au pire égal à celui de T.

Etape 1. Choisir une arête (v_i, v_j) desservie dans T et construire un cycle T' desservant les arêtes de $R_T \setminus \{(v_i, v_j)\}$ en utilisant SUPPRIMER_CLIENT. Construire un cycle T'' desservant les arêtes de R_T en appliquant AJOUTER_CLIENT à T'.

Etape 2. Si la tournée T'' est plus courte que T, poser $T = T''$.

Etape 3. Répéter les étapes 1 et 2 avec toutes les arêtes de R_T jusqu'à ce qu'aucune amélioration de T ne puisse plus être obtenue.

La deuxième procédure de post-optimisation proposée par Hertz et al. est similaire à la procédure 2-opt décrite pour le PVC dans [Cro58].

Algorithme: 2-OPT_PPR

ENTREE: Un cycle T dans $G = (V, E)$ desservant un ensemble $R_T \subseteq R$ d'arêtes.

SORTIE: Un cycle desservant les arêtes de R_T de coût au pire égal à celui de T.

Etape 1. Choisir une orientation de T et sélectionner deux arêtes (v_i, v_j) et (v_k, v_h) dans T.
Remplacer (v_i, v_j) et (v_k, v_h) par respectivement deux plus courtes chaînes SC_{ik} et SC_{jh} dans G. Renverser l'orientation de la chaîne reliant v_j à v_k dans T.
Soit T_1 la nouvelle tournée obtenue et soit R_{T_1} l'ensemble des clients desservis par T_1.
Déterminer une tournée T_2 desservant les arêtes de R_{T_1} au moins aussi courte que T_1 en appliquant RACCOURCIR à T_1.

Etape 2. Si (v_i, v_j) et/ou (v_k, v_h) sont des arêtes de R_T non couvertes par T_2, construire une tournée T_3 desservant les clients de R_T en utilisant AJOUTER_CLIENT.
Si T_3 est plus courte que T, poser $T = T_3$.

Etape 3. Répéter les étapes 1 et 2 avec les deux orientations possibles pour T et avec toutes les paires d'arêtes (v_i, v_j) et (v_k, v_h) de T jusqu'à ce qu'aucune amélioration de T ne puisse plus être obtenue.

Malgré son apparente simplicité, la procédure 2-OPT_PPR appliquée à une tournée T peut passer par des étapes intermédiaires assez compliquées avant l'obtention d'une tournée de valeur inférieure à celle de T. Un tel comportement est différent de ce qui se produit dans le cas du problème du voyageur de commerce où seules deux arêtes sont remplacées par deux autres arêtes. L'exemple ci-dessous illustre les différentes étapes de 2-OPT_PPR.

Exemple:

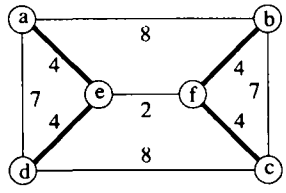


FIG. 1.82 -

Graphe initial G.

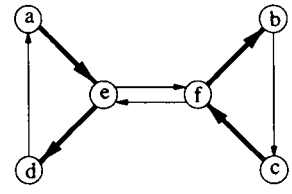


FIG. 1.83 -

Une orientation du cycle T solution du PPR dans G ($R_T = R$).

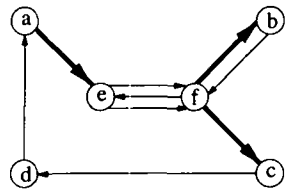


FIG. 1.84 -

Remplacement des arêtes (b, c) et (e, d). Tournée T_1.

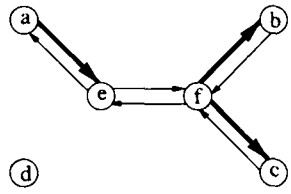


FIG. 1.85 -

Tournée T_1 partiellement raccourcie.

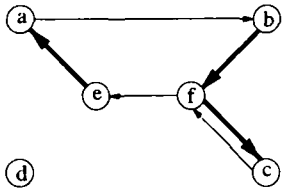


FIG. 1.86 -

Tournée T_2.

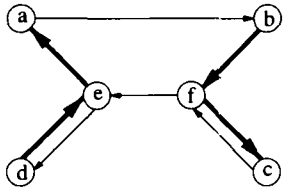


FIG. 1.87 -

L'arête (e, d) est ajoutée à T_2 pour obtenir la tournée T_3.

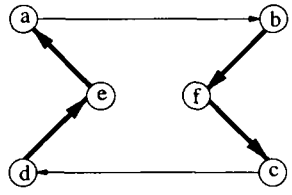


FIG. 1.88 -

Tournée T_3 raccourcie.

L'exemple ci-dessus traite un PPR. Le graphe de départ et le cycle T qui a été orienté sont représentés dans les figures 1.82 et 1.83. Les arêtes (b, c) et (e, d) sont ôtées de T, elles sont remplacées respectivement par les chaînes (b, f, e) et (c, d). L'orientation de la chaîne (c, f, e) de T est renversée pour obtenir la tournée T_1 (figure 1.84). Lors du passage de T_1 dans la procédure RACCOURCIR, le cycle (f, e, f) de T_1 est supprimé et la chaîne (c, d, a) de T_1 est remplacée par la chaîne (c, f, e, a) qui est plus courte (figure 1.85). Il reste encore une chaîne pouvant être raccourcie, il s'agit de (a, e, f, b). Elle est remplacée par la chaîne (a, b) pour finalement fournir la tournée T_2 de la figure 1.86. L'arête à desservir (e, d) ne se trouvant plus dans T_2 il faut la rajouter en utilisant AJOUTER_CLIENT. Nous obtenons ainsi la tournée T_3 (figure 1.87) qui une fois raccourcie nous donne la tournée de la figure 1.88.

Tous les outils que nous avons présentés dans cette section peuvent servir d'ingrédients dans l'adaptation de méta-heuristiques pour le PPR. Corberán et al. [Cor97] ont par exemple développé une méthode tabou pour résoudre le problème du postier rural dans les graphes mixtes. Etant donné une tournée T, une tournée voisine T' de T peut être obtenue en utilisant une procédure similaire à RACCOURCIR. Pour plus de détails le lecteur pourra se référer à [Cor97].
Nous verrons plus loin également que ces outils peuvent aussi être utiles lorsque qu'il s'agit de traiter des problèmes de tournées sur les arcs avec contraintes de capacité.

1.3 Le problème avec contrainte de capacité

Le problème de tournées sur les arcs avec contrainte de capacité (PTAC) (en anglais CARP pour Capacitated Arc Routing Problem) est une généralisation du problème du postier chinois et du problème du postier rural. Le PTAC est défini sur un graphe $G = (V, E \cup A)$ fortement connexe où $V = \{v_0, v_1, ..., v_n\}$ est l'ensemble des sommets, E l'ensemble des arêtes et A l'ensemble des arcs. Le sommet v_0 est un sommet particulier que nous appellerons le dépôt. A ce dépôt sont basés des véhicules de capacité W. La flotte de véhicules se trouvant au dépôt est supposée homogène c'est-à-dire que tous les véhicules ont la même capacité. Le nombre de véhicules de la flotte peut être fixé ou être considéré comme une variable de décision. Comme dans le cas du PPR, R représentera le sous-ensemble de $E \cup A$ des arcs et arêtes à desservir (clients). A chaque élément (v_i, v_j) de R est associé, en plus de son coût c_{ij}, une demande positive notée q_{ij}. Les demandes associées aux éléments de $(E \cup A) \setminus R$ sont nulles. Nous appellerons tournée admissible un cycle ou circuit partant du dépôt, revenant au dépôt et desservant des clients dont la demande totale n'excède pas W. Résoudre un PTAC revient à chercher un ensemble de tournées admissibles dont la longueur totale est minimum et telles que chaque élément de R est traversé au moins une fois par un véhicule. Si un arc ou une arête de R apparaît dans plusieurs tournées ou plusieurs fois dans la même tournée, il n'est desservi qu'une seule fois et par un seul véhicule.

Lorsque la capacité W des véhicules est suffisamment grande pour contenir toutes les demandes du graphe G, le PTAC devient alors un problème du postier rural. Le PPR étant un cas particulier du PTAC, le PTAC est donc NP-dur. Golden et Wong ont montré [Gol81] que même le problème consistant à trouver une $\frac{3}{2}$-approximation pour le PTAC est NP-dur. Nous allons décrire les principales méthodes heuristiques développées pour la résolution du PTAC. Elles seront regroupées en trois catégories: les méthodes constructives simples, les méthodes constructives à deux phases et les adaptations de méta-heuristiques.

Avant de débuter la description proprement dite des algorithmes, nous allons encore introduire une notation. Etant donné deux chaînes (pouvant être fermées) $P = (x, ..., y)$ et $P' = (y, ..., z)$ ayant un sommet commun y, nous noterons $P + P'$ l'union des arêtes de ces deux chaînes. Cette union forme une nouvelle chaîne (pouvant être fermée) $P'' = (x, ..., y, ..., z)$.

1.3.1 Méthodes constructives simples

Toutes les méthodes constructives simples que nous présentons dans cette section ont été initialement développées pour un cas particulier du PTAC et uniquement dans un graphe $G = (V, E)$ non orienté. Ce cas particulier du PTAC est le problème du postier chinois avec capacité (PPCC). Le PPCC est un problème de tournées sur les arcs avec capacité dans lequel tous les arcs et arêtes sont à desservir. Comme nous présentons ici la description originale de ces méthodes, le problème qui sera traité est donc un PPCC dans un graphe non orienté. Ces méthodes peuvent cependant s'adapter facilement à des cas plus généraux du PTAC.

1.3.1.1 Construct-Strike

L'algorithme "Construct-Strike" a été proposé par Christofides en 1973 [Chr73]. L'idée sur laquelle il repose est de construire des tournées admissibles qui, lorsqu'elles sont ôtées du graphe G, ne le déconnectent pas (un sommet isolé n'est dans ce cas pas considéré comme une composante connexe). Une fois qu'une telle tournée est construite, ses arêtes sont enlevées du graphe original et une nouvelle tournée admissible est recherchée. Lorsqu'aucune tournée admissible ne peut plus être trouvée et qu'il reste des arêtes à desservir, des arêtes artificielles facultatives sont ajoutées au graphe restant de manière à ce qu'une tournée admissible puisse à nouveau être construite. Ce procédé est répété jusqu'à ce que toutes les arêtes soient desservies.

Algorithme: CONSTRUCT-STRIKE

ENTREE: Un graphe non orienté $G = (V, E)$.

SORTIE: Une solution (pas forcément optimale) du PPCC dans G.

Etape 0. Poser $G' = G$.

Etape 1. Essayer de déterminer une tournée admissible qui ne déconnecte pas G' (les sommets isolés ne sont pas considérés comme des composantes connexes) lorsque ses arêtes sont supprimées de G'.
Si une telle tournée est trouvée, ôter ses arêtes de G' et répéter l'étape 1.

Etape 2. Supprimer les arêtes artificielles (s'il y en a) de G'. Si toutes les arêtes de G ont été couvertes ($G' = \emptyset$) stop.
Si le dépôt v_0 est de degré nul, ajouter une copie v_0' de v_0 dans G' et relier v_0 à v_0' par une arête artificielle et facultative de coût infini.
Si G' contient des sommets de degré impair aller à l'étape 3, sinon aller à l'étape 4.

Etape 3. Transformer G' en graphe pair en trouvant un couplage parfait de coût minimum entre les sommets de degré impair de G' (voir section 1.2.1.1) et en ajoutant à G' les arêtes correspondant à ce couplage.
Toutes les arêtes ajoutées à G' sont des arêtes artificielles facultatives.
Si une copie du dépôt a été faite à l'étape 2, fusionner v_0 et v_0' et aller à l'étape 1.

Etape 4. Ajouter à G' deux plus courtes chaînes composées d'arêtes artificielles facultatives entre v_0 et son plus proche voisin et retourner à l'étape 1.
Si une tournée admissible ne peut toujours pas être construite, en plus des deux plus courtes chaînes reliant le dépôt à son plus proche voisin, ajouter deux plus courtes chaînes reliant le dépôt à son deuxième plus proche voisin, etc., jusqu'à ce qu'une tournée admissible soit obtenue à l'étape 1.

Dans son article, Christofides ne propose aucune méthode pour construire les tournées admissibles de l'étape 1. Dans les exemples qu'il traite, ces tournées sont construites visuellement. L'exemple qui suit contient les différentes situations pouvant se rencontrer lors de l'utilisation de CONSTRUCT-STRIKE.

Exemple:

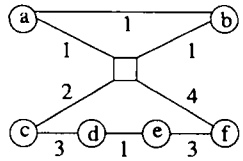


FIG. 1.89 -

Graphe initial G.

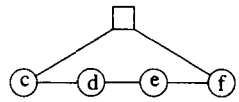


FIG. 1.90 -

La première tournée admissible (v_0, a, b, v_0) est ôtée de G'.

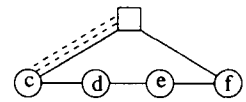


FIG. 1.91 -

Ajout de 2 plus courtes chaînes reliant v_0 à son plus proche voisin.

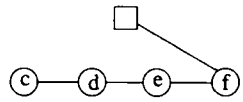


FIG. 1.92 -

La deuxième tournée admissible (v_0, c, v_0) est ôtée de G'.

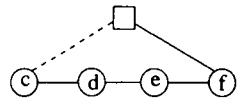


FIG. 1.93 -

Ajout des arêtes du couplage dans G'.

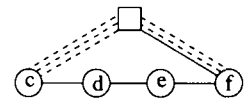


FIG. 1.94 -

Ajout de 2 plus courtes chaînes reliant v_0 à c et v_0 à f.

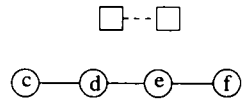


FIG. 1.95 -

La troisième tournée admissible (v_0, f, v_0) est ôtée de G' ainsi que toutes les arêtes artificielles. Une copie du dépôt est ajoutée.

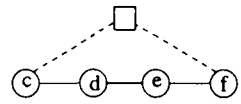


FIG. 1.96 -

Ajout des arêtes du couplage dans G'. Les deux copies du dépôt ont fusionné.

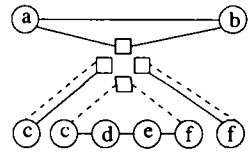


FIG. 1.97 -

Solution finale composée de 4 tournées admissibles.

Dans l'exemple ci-dessus, toutes les arêtes du graphe de la figure 1.89 sont à desservir et ont une demande de 1. La capacité des véhicules W vaut 3. Les valeurs situées sur les arêtes correspondent à leur coût. Les arêtes en traitillés sont les arêtes artificielles facultatives qui sont ajoutées à G' et le sommet carré représente le dépôt. La figure 1.90 représente le graphe G' après avoir enlevé la tournée admissible (v_0, a, b, v_0). Comme aucune tournée admissible ne peut plus être construite dans G', nous ajoutons deux plus courtes chaînes reliant le dépôt à son plus proche voisin, c'est-à-dire deux fois l'arête artificielle facultative (v_0, c) (figure 1.91). La tournée admissible (v_0, c, v_0) peut maintenant être supprimée de

G' qui est représenté dans la figure 1.92. Les deux sommets de degré impair dans G' sont couplés, G' est rendu pair en ajoutant l'arête (v_0, c) (figure 1.93). Pour pouvoir construire une tournée dans G', deux plus courtes chaînes sont ajoutées allant du dépôt à son plus proche voisin et du dépôt à son deuxième plus proche voisin (figure 1.94). La tournée admissible (v_0, f, v_0) est ensuite construite et ôtée de G'. Les arêtes artificielles sont enlevées de G' et comme le dépôt est maintenant de degré 0, il est dupliqué et relié à sa copie (figure 1.95). G' est à nouveau rendu pair (figure 1.96) et finalement la dernière tournée admissible (v_0, c, d, e, f, v_0) est construite. La figure 1.97 représente la solution fournie par CONSTRUCT-STRIKE. Cette solution est composée de 4 tournées et son coût total vaut 28.

1.3.1.2 Construct-Strike modifié

Pearn [Pea89] a proposé une version modifiée de l'algorithme CONSTRUCT-STRIKE de Christofides. Dans cette version, une tournée admissible peut être ôtée du graphe même si cela détruit sa connexité. Cet algorithme est décrit ci-dessous.

Algorithme: CONSTRUCT-STRIKE_MODIFIE

ENTREE: Un graphe non orienté $G = (V, E)$.

SORTIE: Une solution (pas forcément optimale) du PPCC dans G.

Etape 0. Poser $G' = G$ et $G'' = G$.

Etape 1. Choisir une arête (v_0, v_i) incidente au dépôt dans G'' et l'enlever de G''. Poser $P = (v_0, v_i)$ et $v_{fin} = v_i$ (v_{fin} est l'extrémité de P différente de v_0).

Etape 2. Pour chaque arête (v_{fin}, v_j) de G'' déterminer la chaîne de cardinalité minimum S_j dans G'' qui débute par (v_{fin}, v_j) et relie v_{fin} au dépôt. Supprimer de G'' toutes les arêtes (v_{fin}, v_j) telles que la demande totale couverte par $P + S_j$ est supérieure à W.
Si v_{fin} est isolé dans G'' alors poser $G'' = G'$ et aller à l'étape 4.
Sinon déterminer l'arête (v_{fin}, v_j) de G'' qui maximise la demande totale couverte par S_j. Poser $P = P + (v_{fin}, v_j)$ et $v_{fin} = v_j$.
Si $v_{fin} \neq v_0$ répéter l'étape 2. Sinon enlever de G' toutes les arêtes non artificielles de la tournée admissible P et poser $G'' = G'$.

Etape 3. Si toutes les arêtes de G ont été desservies (i.e $G' = \emptyset$) STOP.
Si toutes les arêtes de G'' peuvent être atteintes depuis le dépôt aller à l'étape 1.
Si v_0 est isolé dans G'' ajouter à G'' une plus courte chaîne composée d'arêtes artificielles facultatives entre v_0 et son plus proche voisin non isolé dans G''.

Algorithme: CONSTRUCT-STRIKE_MODIFIE (SUITE)

Etape 4. Si G'' est pair et connexe (avec peut-être des sommets isolés) aller à l'étape 5.
Sinon transformer G'' en graphe pair et connexe de la manière suivante:

> Résoudre un problème d'arbre maximal de coût minimum entre les composantes connexes de G''.
> Ajouter à G'' les arêtes correspondant à cet arbre.
> Résoudre un problème de couplage de coût minimum entre les sommets de degré impair dans G''.
> Ajouter à G'' les arêtes correspondant à ce couplage.

Toutes les arêtes ajoutées à G'' sont des arêtes artificielles facultatives.
Aller à l'étape 1.

Etape 5. Choisir une arête (v_0, v_i) incidente au dépôt dans G''et la supprimer de G''. Poser $P = (v_0, v_i)$ et $v_{fin} = v_i$.

Etape 6. Effacer de G'' toutes les arêtes (v_{fin}, v_j) telles que la demande couverte par $P + (v_{fin}, v_i)$ est plus grande que W. Si v_{fin} est isolé dans G'' aller à l'étape 7.
Choisir n'importe quelle arête (v_{fin}, v_j) de G'', l'effacer de G'', poser $P = P + (v_{fin}, v_j)$, poser $v_{fin} = v_j$ et répéter l'étape 6.

Etape 7. Ajouter à P la plus courte chaîne SC entre v_{fin} et v_0 dans G. $P + SC$ est une tournée admissible dans laquelle toutes les arêtes non artificielles de P sont desservies alors que les arêtes de SC et les arêtes artificielles de P sont seulement traversées.
Enlever de G' les arêtes non artificielles de P, poser $G'' = G'$ et aller à l'étape 3.

Le choix de l'arête incidente au dépôt qui est fait à l'étape 1 peut considérablement affecter la qualité de la solution finale. Pour cette raison, Pearn propose de générer un ensemble de solutions en choisissant à l'étape 1, pour chaque solution, une arête incidente au dépôt différente. La meilleure de ces solutions est alors la solution finale de l'algorithme. La complexité de CONSTRUCT-STRIKE_MODIFIE est de l'ordre de $O(|E||V|^4)$ alors que celle de l'heuristique de Christofides pour le PPCC est de l'ordre de $O(|E||V|^3)$.

L'exemple qui suit illustre le fonctionnement de la version modifiée du CONSTRUCT-STRIKE. Le graphe initial est le même que celui de la figure 1.89. Les arêtes en traitillés sont les arêtes artificielles facultatives. Toutes les arêtes sont à desservir, elles ont une demande de 1. La capacité des véhicules vaut 3.

En choisissant l'arête (v_0, a) à l'étape 1, une première tournée est construite à l'étape 2. Il s'agit de la tournée admissible (v_0, a, b, v_0). Elle est enlevée de G' (figure 1.99). Comme il n'est pas possible de construire une nouvelle tournée admissible au cours des étapes 1 et 2, et que le graphe G'' obtenu au début de l'étape 4 est pair et connexe (en

fait il est identique au graphe G' de la figure 1.99), la deuxième tournée admissible est construite aux étapes 5, 6 et 7. La chaîne $P = (v_0, f, e, d)$ (figure 1.100) est obtenue en choisissant (v_0, f) comme arête de départ (étape 5) et en effectuant l'étape 6. A l'étape 7, une plus courte chaîne reliant v_o à d est ajoutée à P pour obtenir la deuxième tournée (figure 1.101). Le graphe G' obtenu après avoir supprimé les arêtes de P n'est plus pair. Les deux sommets de degré impair de G'' v_0 et d sont alors couplés (figure 1.103). La troisième tournée (v_0, c, d, c, d, v_0) est ensuite construite. Nous obtenons finalement une solution composée de 3 tournées et de coût total égal à 26.

Exemple:

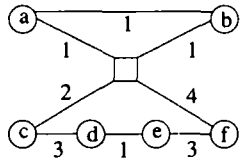


FIG. 1.98 -

Graphe initial G.

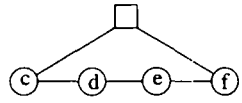


FIG. 1.99 -

La première tournée admissible (v_0, a, b, v_0) est ôtée de G'.

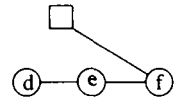


FIG. 1.100 -

Le chemin $P = (v_0, f, e, d)$ ne peut plus desservir d'arête.

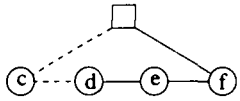


FIG. 1.101 -

Le chemin $SP = (d, c, v_0)$ est ajouté à P pour former la deuxième tournée.

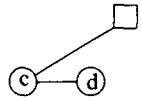


FIG. 1.102 -

Les arêtes desservies dans P sont ôtées de G'

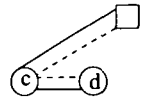


FIG. 1.103 -

Le graphe G' est rendu pair en couplant entre eux les sommets de degré impair.

Pearn utilise une règle particulière à l'étape 2 lors de la construction des tournées admissibles. Il choisit toujours l'arête maximisant la demande totale couverte par la chaîne de cardinalité minimum retournant au dépôt. D'autres règles peuvent être utilisées.

1.3.1.3 Path-Scanning

Proposé par Golden et al. [Gol83], le "Path Scanning" est une méthode heuristique qui consiste à construire les tournées admissibles les unes après les autres. La construction d'une tournée se fait arête par arête en ajoutant chaque fois l'arête à desservir satisfaisant le mieux à un critère fixé. Lorsqu'il n'est plus possible d'insérer un nouveau client dans la tournée, le retour au dépôt s'effectue en utilisant la plus courte chaîne possible.
Golden et al. proposent cinq critères permettant le choix du client à insérer dans une tournée. Etant donné une chaîne $P = (v_0, ..., v_i)$, l'arête à desservir (v_i, v_j) est choisie selon une des cinq règles qui suivent:

règle 1: choisir l'arête (v_i, v_j) qui minimise le coût c_{ij} par unité de demande restant à couvrir

règle 2: choisir l'arête (v_i, v_j) qui maximise le coût c_{ij} par unité de demande restant à couvrir

règle 3: choisir l'arête (v_i, v_j) minimisant la plus courte distance de v_j au dépôt

règle 4: choisir l'arête (v_i, v_j) maximisant la plus courte distance de v_j au dépôt

règle 5: utiliser la règle 4 si le véhicule couvre une demande inférieure ou égale à la moitié de sa capacité et la règle 3 sinon

Algorithme: PATH-SCANNING

ENTREE: Un graphe non orienté $G = (V, E)$.

SORTIE: Une solution (pas forcément optimale) du PPCC dans G.

Etape 0. Poser $G' = G$ et $G'' = G'$.

Etape 1. Si v_0 est isolé dans G'', ajouter à G'' une plus courte chaîne $(v_0, ..., v_i)$ composée d'arêtes artificielles facultatives et reliant v_0 à son plus proche sommet non isolé dans G''. Poser $P = (v_0, ..., v_i)$ et $v_{fin} = v_i$.
Sinon choisir une arête incidente au dépôt (v_0, v_i) dans G'' et l'enlever de G''. Poser $P = (v_0, v_i)$ et $v_{fin} = v_i$.

Etape 2. Supprimer de G'' toutes les arêtes (v_{fin}, v_j) telles que la demande totale couverte par $P + (v_{fin}, v_j)$ dépasse W.
Si v_{fin} est un sommet isolé dans G'' aller à l'étape 3.
Choisir une arête (v_{fin}, v_j) selon une des 5 règles ci-dessus, enlever cette arête de G'', poser $P = P + (v_{fin}, v_j)$, $v_{fin} = v_j$ et répéter l'étape 2.

Etape 3. Ajouter à P la plus courte chaîne SC reliant v_{fin} à v_0 dans G. $P + SC$ est une tournée admissible dans laquelle toutes les arêtes non artificielles de P sont desservies alors que toutes les arêtes de SC et les arêtes artificielles de P sont seulement traversées. Enlever de G' les arêtes non artificielles de P et poser $G'' = G'$.
Si toutes les arêtes de G sont desservies STOP, sinon aller à l'étape 1.

Une solution complète est générée avec chacune des cinq règles d'insertion ci-dessus. La meilleure des cinq solutions ainsi obtenues est la solution finale fournie par l'algorithme PATH-SCANNING. Pearn [Pea89] a proposé une version modifiée de cette méthode qui consiste, à chaque étape de l'algorithme, à choisir une des règles d'insertion selon une probabilité donnée. Plusieurs solutions sont ainsi générées et la meilleure est retenue.

1.3.1.4 Augment-Merge

L'algorithme Augment-Merge a été proposé pour la première fois par Golden et Wong dans [Gol81]. Dans cet article, il n'est que brièvement esquissé. Une version plus détaillée est décrite par Golden et al. dans [Gol83]. Cet algorithme est inspiré de celui développé par Clarke et Wright [Cla64] pour le problème de tournées sur les sommets avec capacité (PTSC) (en anglais VRP pour Vehicule Routing Problem). Partant d'un ensemble de tournées où chaque tournée ne dessert qu'une seule arête, le nombre de véhicules utilisés est progressivement réduit. Cette réduction s'effectue de deux façons. Le service de certaines arêtes est tout d'abord transféré d'une tournée vers une autre. De nouvelles tournées admissibles sont ensuite obtenues en fusionnant deux tournées.
Avant de présenter l'algorithme, nous allons décrire une façon d'effectuer la fusion de deux tournées admissibles. Etant donné une tournée admissible T, soit P_T une des deux chaînes de T de cardinalité maximale partant du dépôt et composé uniquement d'arêtes non desservies. Soit v_T la deuxième extrémité de P_T (la première étant v_0). Une fusion de deux tournées admissibles T et T' est obtenue en remplaçant $P_T + P_{T'}$ dans $T + T'$ par une plus courte chaîne SC reliant v_T à $v_{T'}$ dans G. Soit L_T, $L_{T'}$ et L la longueur totale de P_T, $P_{T'}$ et SC respectivement. Le gain induit par la fusion de T et T' est égal à $L_T + L_{T'} - L$. Les deux choix possibles de P_T dans T et de $P_{T'}$ dans T' engendrent 4 façons différentes de fusionner T et T'. La fusion provoquant le gain le plus important sera retenue. L'exemple ci-dessous illustre la procédure de fusion que nous venons de décrire.

Exemple:

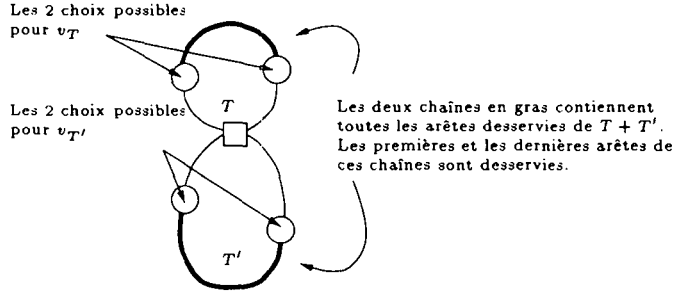


FIG. 1.104 -

Fusion des tournées T et T'.

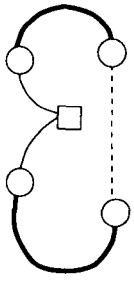


FIG. 1.105 -

Première fusion possible.

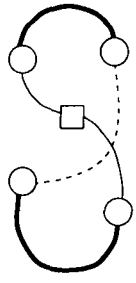


FIG. 1.106 -

Deuxième fusion possible.

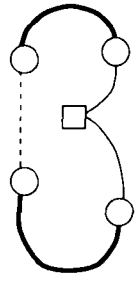


FIG. 1.107 -

Troisième fusion possible.

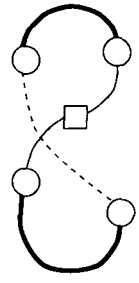


FIG. 1.108 -

Quatrième fusion possible.

Les figures 1.105 à 1.108 représentent les 4 façons de fusionner les deux cycles T et T'. La plus courte chaîne SC reliant v_T à $v_{T'}$ dans G est dessinée en traitillés.

Golden et al. [Gol83] n'imposent pas que P_T et $P_{T'}$ soient de cardinalité maximale et ne considèrent que les fusions où $v_T = v'_T$. Leur algorithme est décrit ci-dessous.

Algorithme: AUGMENT-MERGE

ENTREE: Un graphe non orienté $G = (V, E)$.

SORTIE: Une solution (pas forcément optimale) du PPCC dans G.

Etape 1. Pour chaque arête (v_i, v_j) dans G construire une tournée admissible T_{ij} en reliant v_0 à v_i à l'aide d'une plus courte chaîne (les arêtes de cette chaîne ne sont pas desservies), en desservant (v_i, v_j) puis en regagnant le dépôt par une plus courte chaîne (les arêtes de cette chaîne ne sont pas desservies).

Etape 2. Partant de la tournée T_{ij} la plus longue et tant que la capacité du véhicule le permet, changer le statut des arêtes de T_{ij} qui sont desservies dans des tournées plus courtes. Ces arêtes qui étaient traversées sans être desservies dans T_{ij} sont maintenant desservies dans T_{ij}, et les tournées dans lesquelles elles étaient desservies sont supprimées.

Etape 3. Evaluer toutes les fusions possibles de deux tournées qui n'entraînent pas un dépassement de capacité. Si le plus grand gain obtenu par une de ces fusions est positif, effectuer la fusion qui maximise ce gain et répéter l'étape 3, sinon stop.

1.3.1.5 Parallel-Insert

Une autre technique pour insérer des clients dans une tournée est utilisée par Chapleau et al. [Cha84]. Elle est inspirée de procédures utilisées pour le problème du voyageur de commerce. Deux stratégies d'insertion complémentaires l'une de l'autre sont utilisées:

1) étant donné une arête, déterminer la tournée existante dans laquelle il faut l'insérer de façon à minimiser le détour provoqué par l'insertion

2) étant donné une tournée déterminer quel client non desservi doit y être inséré

En plus des contraintes de capacité, Chapleau et al. imposent également une limite, L_{max}, sur la longueur de chaque tournée. Une version simplifiée de leur algorithme est décrite à la page suivante.

Algorithme: PARALLEL-INSERT

ENTREE: Un graphe non orienté $G = (V, E)$.

SORTIE: Une solution (pas forcément optimale) du PPCC dans G.

Etape 1. Déterminer l'arête (v_i, v_j) la plus éloignée du dépôt et créer une tournée admissible desservant (v_i, v_j) en allant de v_0 à v_i par une plus courte chaîne et en rentrant au dépôt depuis v_j également par une plus courte chaîne (les arêtes de ces deux plus courtes chaînes ne sont pas desservies).

Etape 2. *(Première stratégie d'insertion)*
Choisir l'arête non desservie (v_a, v_b) la plus éloignée du dépôt. Parmi les tournées existantes de capacité suffisante, déterminer celle qui minimise le détour engendré par l'insertion de (v_a, v_b) et dont la longueur totale après insertion de (v_a, v_b) ne dépasse pas L_{max}.
Si une telle tournée T a pu être trouvée, insérer (v_a, v_b) dans T et répéter l'étape 2.
Sinon, tant qu'un nombre de tournées m_{max} n'est pas égalé, créer une nouvelle tournée desservant (v_a, v_b) et répéter l'étape 2.
Déclarer une tournée T "fermée" lorsqu'aucune arête non desservie ne peut y être insérée sans entraîner un dépassement de capacité. Les autres tournées existantes sont "ouvertes".

Etape 3. *(Deuxième stratégie d'insertion)*
Choisir la route "ouverte" T desservant le moins de demandes. Soit E' l'ensemble des arêtes non desservies pouvant être ajoutées à T sans provoquer de dépassement de capacité. Sélectionner l'arête (v_a, v_b) de E' entraînant le plus petit détour.
Si une telle arête n'existe pas ou si après son insertion dans T la longueur de T excède L_{max}, déclarer la route T "fermée". Sinon insérer (v_a, v_b) dans T.
Répéter l'étape 3 jusqu'à ce que toutes les arêtes soient desservies ou jusqu'à ce que toutes les routes soient "fermées".

Etape 4. Si toutes les arêtes sont desservies, stop. Sinon choisir l'arête non traitée (v_a, v_b) la plus éloignée du dépôt, créer une nouvelle tournée desservant (v_a, v_b) et aller à l'étape 2.

Chapleau et al. [Cha84] ne décrivent pas comment insérer une arête dans une tournée. Une telle insertion peut se faire par exemple en utilisant la procédure AJOUTER_CLIENT décrite à la section 1.2.2.5. L'algorithme PARALLEL-INSERT a été appliqué à la résolution d'un problème de ramassage scolaire. Quelques ingrédients supplémentaires ont été introduits. Ils permettent d'améliorer la qualité des solutions obtenues. Nous en décrivons quelques-uns.

A l'étape 4, des procédures de post-optimisation sont appliquées à chacune des tour-

nées. Ces procédures sont du même type que celles présentées à la section 1.2.2.5 pour le PPR. Remarquons que ce genre de procédures peut s'appliquer à n'importe quelle tournée admissible, quel que soit l'algorithme ayant servi à la construction de ces tournées.

Afin de réduire le nombre de véhicules utilisés, Chapleau et al. [Cha84] proposent de relaxer (d'un certain pourcentage) la contrainte imposant une longueur maximum pour chaque tournée. En effet les tournées ayant une longueur supérieure à L_{max} peuvent voir leur longueur réduite en appliquant les procédures de post-optimisation. Une autre façon de réduire le nombre de véhicules utilisés est également proposée dans [Cha84]. Il s'agit de supprimer la tournée desservant le moins de demandes et d'insérer les clients desservis par cette tournée dans les autres tournées en utilisant les étapes 2, 3 et 4 de PARALLEL-INSERT.

Finalement, Chapleau et al. décrivent une stratégie ayant pour but d'éviter que les tournées ne fassent des "zigzags". Soit une tournée T et une arête (v_a, v_b) à insérer dans T. Si la longueur de T est inférieure à une certaine valeur, (v_a, v_b) peut être insérée n'importe où dans T. Sinon (v_a, v_b) peut seulement être insérée entre le dépôt et le dernier sommet incident à une arête desservie dans T.

1.3.1.6 Augment-Insert

L'algorithme dont il est question dans cette section combine les principes utilisés dans AUGMENT-MERGE et dans PARALLEL-INSERT. Il a été proposé par Pearn [Pea91]. Dans un premier temps, des tournées admissibles sont construites en utilisant une approche semblable à celle employée à l'étape 2 de AUGMENT-MERGE (section 1.3.1.4). Les arêtes qui ne sont pas traitées après cette première phase sont ensuite insérées séquentiellement dans des tournées en utilisant une règle similaire à la première stratégie d'insertion proposée par Chapleau et al. [Cha84] (section 1.3.1.5).

Algorithme: AUGMENT-INSERT

ENTREE: Un graphe non orienté $G = (V, E)$.

SORTIE: Une solution (pas forcément optimale) du PPCC dans G.

Etape 1. Soit SC_{ij} la plus courte chaîne entre les sommets v_i et v_j dans G, et soit sc_{ij} sa longueur. Pour chaque arête (v_i, v_j) de G définir $D_{ij} = sc_{0i} + sc_{j0}$. Poser $G' = G$.

Algorithme: AUGMENT_INSERT (SUITE)

Etape 2. Déterminer l'arête (v_i, v_j) dans G' qui a la plus grande valeur de D_{ij} et qui est contenue dans un cycle de G' dans lequel se trouve également le dépôt.
Si une telle arête n'existe pas aller à l'étape 3. Sinon déterminer le cycle de moindre coût T dans G' contenant (v_i, v_j) et le dépôt, et considérer T comme une tournée ne desservant que (v_i, v_j).
Aussi longtemps que la capacité W des véhicules n'est pas dépassée et en considérant les arêtes (v_a, v_b) selon l'ordre décroissant des D_{ab}, changer le statut des arêtes (v_a, v_b) de T qui sont traversées sans être desservies.
Soit v_p et v_q le premier et le dernier sommet de T incident à une arête desservie. Remplacer la chaîne reliant v_0 à v_p et la chaîne reliant v_q à v_0 par respectivement SC_{0p} et SC_{q0}. Enlever les arêtes desservies de G'.
Si toutes les arêtes de G sont desservies alors stop, sinon répéter l'étape 2.

Etape 3. Déterminer l'arête non desservie (v_i, v_j) qui a la plus grande valeur de D_{ij} et créer une tournée $T = SC_{0i} + (v_i, v_j) + SC_{j0}$ où (v_i, v_j) est la seule arête desservie.

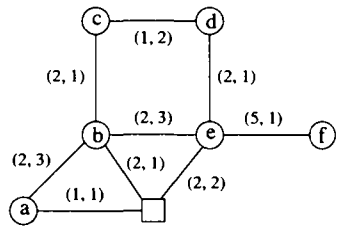
Etape 4. Soit E' l'ensemble des arêtes non desservies pouvant être ajoutées à T sans provoquer un dépassement de capacité, et soit (v_a, v_b) l'arête de E' dont la valeur D_{ab} associée est maximum.
Calculer le détour occasionné par l'insertion de (v_a, v_b) entre le dépôt et le premier (ou le dernier) sommet incident à une arête desservie dans T.
Si le coût de cette insertion est supérieur à une borne fixée B alors ôter (v_a, v_b) de E'. Sinon insérer (v_a, v_b) dans T.
Répéter l'étape 4 jusqu'à ce que E' soit vide ou que les deux arêtes incidentes au dépôt dans T soient desservies.

Etape 5. Enlever les arêtes desservies de G'.
Si $G' = \emptyset$ stop. Sinon aller à l'étape 3.

Pearn a proposé deux versions de l'algorithme AUGMENT-INSERT. La version ci-dessus est la version I. La version II utilise à l'étape 2 un cycle de moindre demande au lieu d'un cycle de moindre coût. La borne B employée à l'étape 4 est un paramètre de contrôle sur la longueur des tournées. En considérant les arêtes selon l'ordre décroissant des D_{ij}, les arêtes les plus éloignées du dépôt sont favorisées.

Nous allons illustrer cet algorithme à l'aide de l'exemple de la page suivante. Les nombres entre parenthèses situés au-dessus des arêtes (figure 1.109) représentent respectivement leur longueur et leur demande. Le dépôt est représenté par un sommet carré et les arêtes qui dans une tournée ne sont pas desservies sont dessinées en traitillés. La capacité des véhicules W est égale à 5, et on suppose que la borne B sur le coût d'insertion d'une arête est supérieure ou égale à 3.

Exemple:



(v_i, v_j)	D_{ij}
(e, f)	9
(c, d)	8
(b, c)	6
(d, e)	6
(b, e)	4
(a, b)	3
(v_0, b)	2
(v_0, e)	2
(v_0, a)	1

FIG. 1.109 -

Graphe original G et ordre décroissant des D_{ij}.

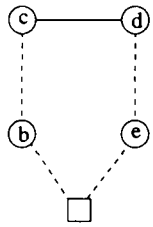


FIG. 1.110 -

Cycle T généré au début de l'étape 2.

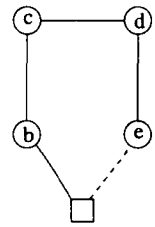


FIG. 1.111 -

Première tournée admissible.

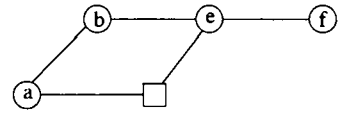


FIG. 1.112 -

Graphe restant G'.

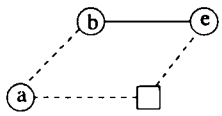


FIG. 1.113 -

Cycle T généré au début de l'étape 2.

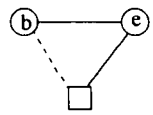


FIG. 1.114 -

Deuxième tournée admissible.

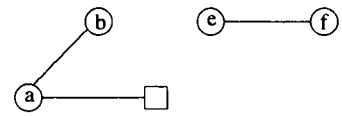


FIG. 1.115 -

Graphe restant G'.

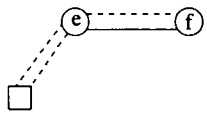


FIG. 1.116 -

Cycle T généré au début de l'étape 3.

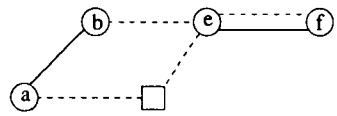


FIG. 1.117 -

Insertion de (a, b).

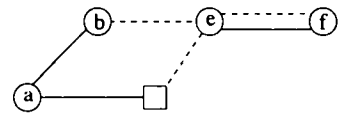


FIG. 1.118 -

Troisième tournée admissible.

L'arête qui maximise les D_{ij} est l'arête (e, f). Cette arête cependant n'appartient pas à un cycle de G' passant par le dépôt. Le cycle généré à l'étape 2 va donc desservir la deuxième

arête selon l'ordre décroissant des D_{ij} à savoir (c,d) (figure 1.110). Le statut des arêtes (b,c), (d,e) et (v_0,b) de ce cycle est changé, ces trois arêtes y sont maintenant desservies (figure 1.111). La demande desservie par cette tournée étant égale à W, aucune autre arête ne peut plus y être desservie. Les arêtes (c,d), (b,c), (d,e) et (v_0,b) sont enlevées de G' pour obtenir le graphe de la figure 1.112. L'étape 2 est répétée avec les arêtes qui ne sont pas encore desservies. L'arête (b,e) maximise D_{ij} parmi les arêtes restantes et appartient à un cycle de G' passant par le dépôt (figure 1.113). Le statut de (v_0,e) est modifié et la chaîne (v_0,a,b) est remplacée par la chaîne $SC_{0,b} = (v_0,b)$ (figure 1.114). Passant à l'étape 3, le cycle T représenté à la figure 1.116 est construit. L'arête (a,b) maximise les D_{ij} des arêtes non encore desservies. De plus le détour engendré par l'insertion de (a,b) dans T vaut $c_{0a} + c_{ab} + c_{be} - c_{0e} = 1 + 2 + 2 - 2 = 3$ ce qui est inférieur ou égal à B (nous avons supposé que $B \geq 3$). L'arête (a,b) est donc insérée dans T (figure 1.117). Finalement le statut de la dernière arête non traitée est modifié dans la nouvelle tournée obtenue pour fournir la troisième et dernière tournée admissible (figure 1.118).

1.3.2 Méthodes constructives à deux phases

Les méthodes constructives à deux phases peuvent être divisées en deux catégories. La première catégorie contient les méthodes qui consistent à construire une tournée englobant tous les clients puis à partager cette tournée en tournées admissibles. Cette stratégie sera appelée "Tournée & Groupes" (en anglais "route first-cluster second"). La deuxième catégorie englobe les algorithmes où tout d'abord des groupes de clients dont la demande ne dépasse pas W sont créés, et où ensuite pour chaque groupe, une tournée desservant les clients du groupe est construite. Nous appellerons cette stratégie "Groupes & Tournées" (en anglais "cluster first-route second"). La construction des tournées, dans une stratégie comme dans l'autre, se fait généralement en résolvant un problème de postier rural.

Toutes les méthodes présentées pouvant s'étendre moyennant quelques adaptations aux cas orienté et mixte, seul le cas non orienté sera traité. Le graphe sur lequel nous allons travailler est donc un graphe connexe $G = (V,E)$ non orienté. L'ensemble des arêtes à desservir sera noté R. Comme nous l'avons déjà fait, nous noterons par SC_{ij} la plus courte chaîne entre v_i et v_j dans G.

1.3.2.1 Stratégie "Tournée & Groupes"

Toutes les méthodes basées sur la stratégie "Tournée & Groupes" consistent tout d'abord à construire une tournée couvrant tous les clients. Pour cela, et suivant la nature de R, les algorithmes présentés aux sections 1.2.1.1 et 1.2.2.1 peuvent être utilisés. Nous allons décrire dans cette section uniquement les différentes techniques proposées pour partager cette tournée en tournées admissibles.

Un algorithme intéressant permettant de faire ce partage a été développé par Ulusoy [Ulu85]. Il s'inspire d'une méthode décrite par Beasley pour le problème de tournées sur les sommets avec capacité [Bea83]. Le graphe de départ G est transformé en graphe auxiliaire dans lequel un plus court chemin entre deux sommets est recherché. Cet algorithme est décrit à la page suivante.

Algorithme: PARTAGER_TOURNEE

ENTREE: Un graphe non orienté $G = (V, E)$, un ensemble $R \subseteq E$ et un cycle T couvrant R.

SORTIE: Une solution (pas forcément optimale) du PTAC dans G.

Etape 0. Si une arête est dans R, et qu'elle apparaît plusieurs fois dans T, elle est desservie la première fois qu'elle est traversée.

Etape 1. Renommer les sommets de V de telle sorte que le cycle T soit égal à $(v_0, v_1, v_2, ..., v_0)$ où v_0 est le dépôt. Soit r le plus grand indice d'un sommet incident à une arête desservie dans T.
Construire un graphe orienté $G' = (V', A')$ où $V' = \{v'_0, v'_1, ..., v'_r\}$ est l'ensemble des sommets et où un arc (v'_a, v'_b) appartient à A' si et seulement si la demande desservie sur la chaîne de T allant de v_a à v_b ne dépasse pas W.
Effacer les arcs (v_a, v_b) si $b > a + 1$ si les arêtes (v_a, v_{a+1}) ou (v_{b-1}, v_b) ne sont pas desservies dans T.
Définir le coût c'_{ab} de (v'_a, v'_b) dans G' de la manière suivante:

Si $b = a + 1$ et (v_a, v_{a+1}) n'est pas desservie dans T alors $c'_{ab} = 0$.

Si $b > a + 1$ ou (v_a, v_{a+1}) est desservie dans T alors il faut considérer la chaîne $P_{ab} = (v_a, ..., v_b)$ dans T. Si P_{ab} contient le dépôt alors poser $P_{ab} = P_{ab} + SC_{ba}$. Sinon poser $P_{ab} = P_{ab} + SC_{0a} + SC_{b0}$. Dans les deux cas P_{ab} est un cycle T_{ab} dans G et c'_{ab} est égale à la longueur totale de ce cycle.

Etape 2. Trouver un plus court chemin dans G' entre v'_0 et v'_r. Chaque arc apparaissant dans ce chemin correspond à une tournée admissible dans G.

L'exemple qui suit illustre le fonctionnement de PARTAGER_TOURNEE. Les arêtes en gras dans le graphe G représentent les arêtes à desservir. Ces arêtes ont une demande de 1. La capacité des véhicules vaut 3. Le nombre situé au-dessus de chaque arête correspond à son coût. Les arêtes en traitillés figurant dans les tournées sont les arêtes de la tournée qui sont traversées sans être desservies. Le dépôt est représenté par un carré.

Exemple:

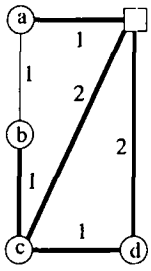


FIG. 1.119 -

Graphe initial G.

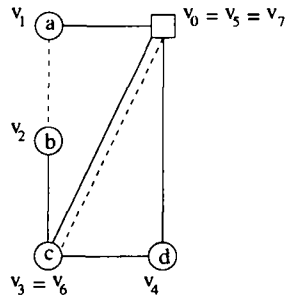


FIG. 1.120 -

Un cycle T couvrant R.

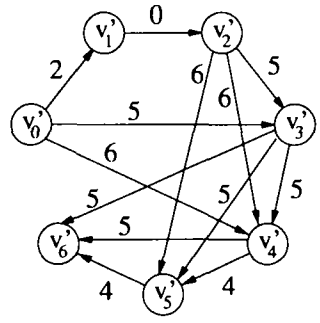


FIG. 1.121 -

Graphe auxiliaire G'.

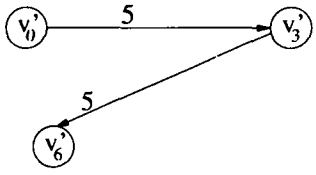


FIG. 1.122 -

Un plus court chemin dans G' entre v'_0 et v'_6.

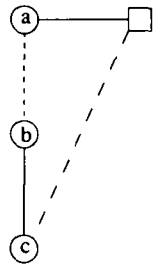


FIG. 1.123 -

Tournée admissible associée à (v'_0, v'_3).

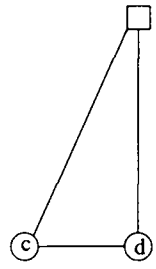


FIG. 1.124 -

Tournée admissible associée à (v'_3, v'_6).

Les figures 1.119 et 1.120 représentent le graphe de départ G et la tournée $T = (v_0, v_1, ..., v_7)$ contenant toutes les arêtes à desservir. Comme le plus grand indice d'un sommet de T incident à une arête desservie est 6, l'ensemble des sommets du graphe auxiliaire G' est le suivant: $\{v'_0, ..., v'_6\}$. Il n'y a pas d'arc (v'_0, v'_2), (v'_1, v'_3), (v'_1, v'_4), (v'_1, v'_5) et (v'_1, v'_6) dans G' car l'arête (v_1, v_2) dans T n'est pas desservie. Les arcs (v'_0, v'_5), (v'_0, v'_6) et (v'_2, v'_6) n'existent pas non plus car la chaîne qui leur est associée dans T dessert une demande supérieure à $W = 3$.
Le calcul des coûts des arcs de G' peut s'illustrer en regardant ce qui se passe pour l'arc (v'_4, v'_6). A cet arc correspond la chaîne dans T $P_{46} = (v_4, v_5, v_6)$. Cette chaîne contient le dépôt et son coût est de 4. La chaîne $SC_{64} = (v_6, v_4) = (c, d)$ est la plus courte chaîne reliant $v_6 = c$ à $v_4 = d$ dans G. Son coût vaut 1. Le coût dans G' de (v'_4, v'_6) est égal à la longueur de la chaîne $P_{46} + SC_{64} = (v_4, v_5, v_6, v_4)$. Nous avons donc $c'_{46} = 5$.
Le plus court chemin reliant v'_0 à v'_6 est représenté dans la figure 1.122. Les tournées associées aux arcs de ce chemin sont dessinées dans les figures 1.123 et 1.124.

Nous avons proposé dans [Her2000] une approche très différente pour partager une tournée couvrant R en tournées admissibles. Etant donné $T = (v_0, v_1, ..., v_t = v_0)$ une tournée

desservant une demande D supérieure à W, Hertz et al. commencent par déterminer un sommet v_r de T tel que la demande à couvrir sur la chaîne de T $(v_0, v_1, ..., v_r)$ ne dépasse pas W et celle située sur la chaîne $(v_r, ..., v_t)$ n'excède pas $W(\lceil \frac{D}{W} \rceil - 1)$. Une fois v_r déterminé, une tournée admissible est construite en ajoutant à $(v_0, ..., v_r)$ la plus courte chaîne SC_{r0} reliant v_r au dépôt. Soit $r' \geq r$ le plus petit indice tel que l'arête $(v_{r'}, v_{r'+1})$ soit desservie dans T. Le procédé que nous venons de décrire est appliqué sur la tournée T' obtenue à partir de T en remplaçant la chaîne $(v_0, v_1, ..., v_{r'})$ dans T par la chaîne $SC_{0r'}$. Les arêtes de cette dernière chaîne ne sont pas desservies dans T'. Cet algorithme est décrit de manière plus détaillée ci-dessous.

Algorithme: DECOUPER_TOURNEE

ENTREE: Un graphe non orienté $G = (V, E)$, un ensemble $R \subseteq E$ et un cycle T couvrant R.

SORTIE: Une solution (pas forcément optimale) du PTAC dans G.

Etape 0. Si une arête est dans R, et qu'elle apparaît plusieurs fois dans T, elle est desservie la première fois qu'elle est traversée.

Etape 1. Renommer les sommets de V de telle sorte que le cycle T soit égal à $(v_0, v_1, v_2, ..., v_t = v_0)$ où v_0 est le dépôt. Soit D la demande totale desservie dans T. Si $D \leq W$, STOP (T est une tournée admissible).

Etape 2. Déterminer le plus grand indice s tel que (v_{s-1}, v_s) soit desservie dans T et tel que la chaîne dans T $(v_0, v_1, ..., v_s)$ ne desserve pas une demande supérieure à W.
Déterminer le plus petit indice s' tel que $(v_{s'-1}, v_{s'})$ soit desservie dans T et tel que la demande totale desservie dans $(v_{s'}, ..., v_t = v_0)$ n'excède pas $W(\lceil \frac{D}{W} \rceil - 1)$.
Si $s' \geq s$ alors poser $r = s$ et aller à l'étape 3.
Sinon pour chaque indice $r \in [s', s]$ tel que (v_{r-1}, v_r) est desservie faire les opérations suivantes:

> Déterminer le plus petit indice $r' \geq r$ tel que $(v_{r'}, v_{r'+1})$ est desservie dans T.
> Poser $P_r = SC_{r0} + SC_{0r'}$. Calculer δ_r égale à la différence entre la longueur de P_r et celle de la chaîne $(v_r, ..., v_{r'})$ dans T.

Choisir l'indice $r \in [s', s]$ qui minimise δ_r.

Etape 3. Construire la tournée admissible $(v_0, ..., v_r) + SC_{r0}$. Les arêtes desservies dans cette tournée sont celles desservies dans la chaîne $(v_0, ..., v_r)$ de T et, si la capacité résiduelle du véhicule le permet, les arêtes de R non encore desservies rencontrées en parcourant SC_{r0}.
Remplacer dans T la chaîne $(v_0, ..., v_r)$ par la chaîne $SC_{0r'}$ dont les arêtes ne sont pas desservies et retourner à l'étape 1.

L'algorithme DECOUPER_TOURNEE est en fait une généralisation d'une procédure proposée par Greistorfer [Gre94]. Dans cette procédure, r est toujours choisi égal à s. L'exemple ci-dessous illustre le fonctionnement de DECOUPER_TOURNEE ainsi que l'amélioration que nous pouvons dans certains cas obtenir par rapport à la procédure proposée par Greistorfer. Comme dans l'exemple précédent, les arêtes en gras dans le graphe initial G représentent les clients. Ils ont tous une demande de 1. Toutes les arêtes ont un coût de 1, la capacité des véhicules est égale à 3. Les arêtes en traitillés dans les tournées sont les arêtes qui sont traversées sans être desservies.

Exemple:

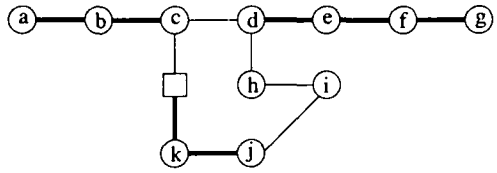


FIG. 1.125 -

Graphe initial G.

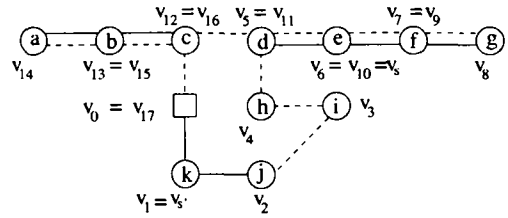


FIG. 1.126 -

Tournée T couvrant R.

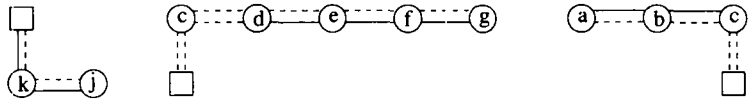


FIG. 1.127 -

Tournées admissibles fournies par DECOUPER_TOURNEE.

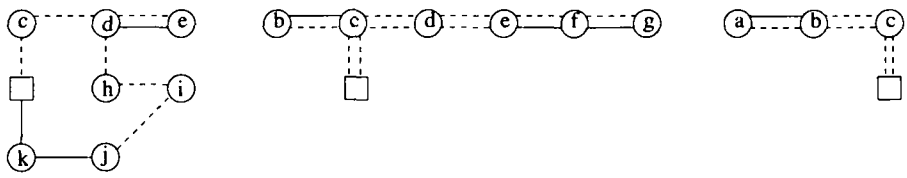


FIG. 1.128 -

Tournées admissibles fournies par l'algorithme de Greistorfer.

Le graphe initial G et la tournée T couvrant toutes les arêtes à desservir sont représentés dans les figures 1.125 et 1.126. La première tournée admissible doit au moins atteindre le sommet $v_{s'} = v_1$ avant de rentrer au dépôt, sinon au moins 4 véhicules devront être utilisés. Les contraintes de capacité imposent que cette première tournée ne desserve pas plus de trois arêtes, ce qui signifie que $v_s = v_6$. L'indice r est donc choisi dans l'intervalle $[1, 6]$. Trois choix de r sont possibles, $r = 1$, $r = 2$ et $r = 6$ puisqu'il faut également que

l'arête (v_{r-1}, v_r) soit desservie dans T. Les valeurs de δ_r ($r \in \{1, 2, 6\}$) sont les suivantes: $\delta_1 = 2, \delta_2 = 1, \delta_6 = 6$. Le r choisi à la fin de l'étape 2 de DECOUPER_TOURNEE vaut donc 2 (avec la procédure de Greistorfer r vaut 6). La première tournée admissible est donc $(v_0, v_1, v_2) + SC_{20} = (v_0, v_1, v_2, v_1, v_0)$. Les arêtes desservies dans cette tournée sont les arêtes (v_0, v_1) et (v_1, v_2). Pour obtenir la deuxième tournée admissible, le même procédé est répété sur la nouvelle tournée $T = (v_0, c, d, e, f, g, f, e, d, c, b, a, b, c, v_0)$. Finalement la longueur totale des tournées obtenues en utilisant DECOUPER_TOURNEE est de 20 alors que la solution obtenue avec la procédure de Greistorfer vaut 27.

Comme nous l'avons mentionné dans la description de DECOUPER_TOURNEE, il peut arriver que s' soit supérieur à s. Considérons la tournée $T = (v_0, v_1, v_2, v_0)$ où chaque arête est desservie et a une demande de 2. La capacité W des véhicules est elle fixée à 3. Un véhicule ne peut jamais servir plus d'une arête à la fois. Cela signifie que $v_s = v_1$. Comme $W(\lceil \frac{D}{W} \rceil - 1) = 3$, $v_{s'} = v_2$. Nous avons donc $s' > s$. Une telle situation apparaît parce qu'il n'existe pas de solution utilisant seulement $\lceil \frac{D}{W} \rceil = 2$ véhicules. Dans ce cas, l'algorithme pose $r = s = 1$ et la première tournée admissible dessert (v_0, v_1). Deux véhicules sont ensuite nécessaires pour servir (v_1, v_2) et (v_2, v_0).

Comme Hertz et al. le proposent dans [Her2000] pour DECOUPER_TOURNEE, tous les algorithmes que nous venons de décrire peuvent être améliorés en appliquant sur chacune des tournées admissibles de la solution obtenue les procédures de post-optimisation décrites à la section 1.2.2.5.

Finalement, terminons cette section en mentionnant quelques résultats théoriques obtenus par Jansen [Jan93]. Soit T une tournée couvrant R qui est une $\frac{3}{2}$-approximation de la tournée optimale T^* couvrant R (une telle approximation peut être obtenue en utilisant l'algorithme PPR_NON_ORIENTE de la section 1.2.2.1). Soit S la solution du PTAC obtenue à partir de T en appliquant l'algorithme PARTAGER_TOURNEE de Ulusoy, et soit S^* la solution optimale du PTAC. Jansen a montré que:

1) Si W est pair et les demandes ne sont pas toutes égales alors S est une $(\frac{7}{2} - \frac{3}{W})$-approximation de S^*.

2) Si W est impair et les demandes ne sont pas toutes égales alors S est une $(\frac{7}{2} - \frac{5}{W+1})$-approximation de S^*.

3) Si toutes les demandes sont égales et quelle que soit la parité de W alors S est une $(\frac{5}{2} - \frac{3}{2W})$-approximation de S^*.

Pour plus de détails concernant ces bornes, le lecteur pourra se référer à [Jan93].

1.3.2.2 Stratégie "Groupes & Tournées"

La stratégie "Groupes & Tournées" consiste tout d'abord à déterminer une partition des arêtes de R dont la demande totale des sous-ensembles de R qui la composent ne dépasse pas W. Pour chaque sous-ensemble, une tournée admissible desservant ses arêtes est ensuite construite. Pour cette deuxième phase, les algorithmes décrits aux sections 1.2.1.1 et 1.2.2.1 peuvent être utilisés. Nous décrivons dans cette section l'algorithme proposé par Benavent et al. [Ben90] pour partitionner les clients de G. Cette méthode détermine une affectation des clients aux véhicules en résolvant un problème d'affectation généralisé PAG (en anglais GAP pour Generalized Assignment Problem). Elle s'inspire de l'algorithme proposé par Fisher et Jaikumar pour le problème de tournées sur les sommets avec capacité [Fis81a].

Algorithme: CYCLE_AFFECTATION

ENTREE: Un graphe non orienté $G = (V, E)$, un ensemble $R \subseteq E$ et un nombre fixé K de véhicules disponibles.

SORTIE: Une affectation des arêtes de R aux véhicules.

Etape 1. Déterminer un ensemble de K sommets (appelés centres) ne contenant pas le dépôt $\{s_1, s_2, ..., s_K\}$ de la manière suivante:

> Pour $k = 1, ..., K$ choisir un sommet s_k maximisant le produit des plus courtes distances le séparant des centres $s1, ..., s_{k-1}$ et du dépôt.

Effectuer des échanges entre les centres et les sommets qui ne sont pas des centres tant que ces échanges permettent d'augmenter le produit des plus courtes distances entre les centres et des plus courtes distances entre les centres et le dépôt.

Etape 2. Soit $G' = G_R$. Résoudre un problème d'arbre maximal de coût minimum entre les composantes connexes de G_R et ajouter les arêtes de G correspondant à celles de l'arbre dans G'. Résoudre un problème de couplage de coût minimum entre les sommets de G' de degré impair, ajouter à G' les arêtes de G correspondant à celles du couplage. (voir section 1.2.2.1). Pour chaque $k = 1, ..., K$ construire le graphe G'_k contenant uniquement le sommet s_k. Déclarer tous les véhicules "ouverts".

Etape 3. Choisir le véhicule k ayant la plus grande capacité résiduelle et déterminer dans G' le cycle de demande minimum contenant au moins un sommet de G'_k.
Si un tel cycle n'existe pas ou si la demande de ce cycle est supérieure à la capacité résiduelle disponible du véhicule k, déclarer ce véhicule "fermé". Sinon, ajouter ce cycle à G'_k et enlever ses arêtes de G'.
Répéter l'étape 3 jusqu'à ce que tous les véhicules soient "fermés". Soit D_k la plus courte distance entre le dépôt et s_k dans G.

Algorithme: CYCLE_AFFECTATION (SUITE)

Etape 4. Soit $P_{k,ij}$ la chaîne de demande minimum entre v_i et v_j dans G'_k, et soit $L_{k,ij}$ sa demande. Soit encore r_k la capacité résiduelle du véhicule k.
Si deux sommets v_i et v_j sont reliés dans G' par une chaîne P de demande L, et s'il existe un véhicule k, visitant v_i et v_j tel que $0 < L - L_{k,ij} < r_k$ alors remplacer $P_{k,ij}$ par P dans G'_k et P par $P_{k,ij}$ dans G'. Répéter ce type d'échanges tant que la demande totale des arêtes à desservir de G' peut être diminuée. Si cette demande est nulle, STOP, toutes les arêtes de R sont affectées à des véhicules.

Etape 5. Pour chaque véhicule k et chaque arête e de R n'appartenant pas à G'_k déterminer la plus courte chaîne $SC_{k,e}$ dans G contenant e et reliant le dépôt à s_k. Soit $\delta_{k,e}$ la différence entre la longueur de $SC_{k,e}$ et D_k, et soit $q_e = q_{ij}$ la demande de l'arête $e = (v_i, v_j) \in R$.
Pour chaque arête e de R appartenant à G'_k poser $\delta_{k,e} = 0$.
Soit $x_{k,e}$ les variables booléennes définies pour chaque véhicule $k = 1, ...K$ et pour chaque arête e de R. Si l'arête e est affectée au véhicule k alors $x_{k,e}$ vaut 1, dans le cas contraire $x_{k,e}$ est nulle.
Résoudre le problème d'affectation généralisé suivant:

$$\text{Min} \quad \sum_{k=1}^{K} \sum_{e \in R} \delta_{k,e} x_{k,e}$$

$$\text{s.c.} \quad \sum_{k=1}^{K} x_{k,e} = 1 \qquad \forall\, e \in R$$

$$\sum_{e \in R} q_e x_{k,e} \leq W \qquad \forall\, k = 1, ..., K$$

$$x_{k,e} \in \{0, 1\} \qquad \forall\, k = 1, ..., K \text{ et } \forall\, e \in R.$$

Une affectation des arêtes de R aux différents véhicules est ainsi obtenue.

L'algorithme que nous venons de décrire détermine tout d'abord une affectation partielle des arêtes de R aux véhicules (étapes 1 à 4). Les informations concernant cette affectation sont prises en compte lors de la résolution du PAG au travers de la définition des coûts $\delta_{k,e}$. En effet, $\delta_{k,e} = 0$ pour toutes les arêtes déjà affectées au véhicule k. Les autres valeurs de $\delta_{k,e}$ sont positives et correspondent à une approximation du coût d'insertion de e dans le véhicule k. Les coûts $\delta_{k,e}$ auront ainsi tendance à faire en sorte que l'affectation partielle obtenue à la fin de l'étape 4 soit conservée.
La résolution du PAG de l'étape 5 peut se faire en utilisant par exemple l'algorithme exact proposé par Ross et Soland dans [Ros75].

1.3.3 Adaptations de méta-heuristiques

Ces dernières années, plusieurs méta-heuristiques ont été utilisées avec succès pour résoudre des problèmes d'optimisation combinatoire (voir par exemple [Ree93], [Aar97]). Dans le domaine des tournées sur les sommets, des méthodes de recherche locale (méthode tabou, recuit simulé) se sont avérées très performantes (voir par exemple [Gen94], [Roc94], [Roc95]). De telles méthodes ont été utilisées récemment pour la résolution de problèmes de tournées sur les arcs avec capacité ([Li92], [Egl94], [Her2000]).
Avant de présenter quelques adaptations de méta-heuristiques pour le PTAC, nous allons décrire de façon très générale le principe sur lequel repose une méthode de recherche locale.

Considérons le problème d'optimisation suivant: $\min_{s \in S} f(s)$ où l'ensemble S est l'ensemble des solutions du problème et f est une fonction réelle définie sur S. Etant donné une solution $s \in S$, un voisinage $N(s) \subseteq S$ est l'ensemble des solutions de S pouvant être obtenues à partir de s en lui appliquant un certain type de modifications.
Supposons qu'un voisinage $N(s) \subseteq S$ soit défini pour chaque solution $s \in S$. Une méthode de recherche locale est une méthode itérative qui a pour but de trouver une solution de S qui minimise f. Le processus itératif part d'une solution initiale se trouvant dans S et, étant donné une solution s, choisit la prochaine solution visitée s' dans $N(s)$. Un critère d'arrêt est défini pour interrompre la recherche locale.

Description générale d'une méthode de recherche locale

Etape 0. Choisir une solution initiale dans S. Poser $s_{best} = s$.

Etape 1. Trouver une "bonne" solution s' dans $N(s)$.
Si $f(s') < f(s_{best})$ alors poser $s_{best} = s'$.

Etape 2. Poser $s = s'$. Si le critère d'arrêt est satisfait, stop.
Sinon aller à l'étape 1.

Trouver la meilleure solution s' de $N(s)$ est parfois un problème qui peut être aussi difficile à résoudre que le problème original. Dans de tels cas, des heuristiques sont utilisées pour produire à chaque itération une "bonne" solution s' dans $N(s)$.

Nous n'allons pas décrire en détail les méthodes de recherche locale développées pour le PTAC. Nous allons simplement présenter brièvement les voisinages développés, et mentionner la méthode de résolution utilisée (recuit simulé, méthode tabou, etc.).

En 1994, Eglese [Egl94] a développé une méthode de recuit simulé pour résoudre un PTAC avec dépôts multiples prenant en compte d'autres contraintes additionnelles issues de la pratique. Il utilise pour cela une modélisation originale. Soit G le graphe de départ. Le graphe G_R est transformé en un graphe G' pair en résolvant un problème de couplage minimum entre ses sommets de degré impair. Le graphe G' est alors décomposé en cycles vérifiant les contraintes de capacité. Ces cycles sont ensuite représentés par les sommets

d'un graphe auxiliaire G''. Deux sommets de G'' sont reliés par une arête de coût nul si les deux cycles qu'ils représentent ont un sommet commun dans G'. Des sommets représentant les dépôts sont ensuite ajoutés à G''. Chaque sommet représentant un cycle est relié au dépôt le plus proche du cycle par une arête dont le poids est égal à la plus courte distance entre un sommet du cycle et le dépôt. Chaque arbre de G'' contenant un dépôt correspond à une tournée dans le graphe de départ G'. L'exemple ci-dessous tiré de [Egl94] illustre la construction de G''. Toutes les arêtes sont à desservir et ont une demande de 1, la capacité des véhicules vaut 7. Les arêtes en traitillés de la figure 1.129 représentent les arêtes du couplage entre les sommets de degré impair dans G_R.

Exemple

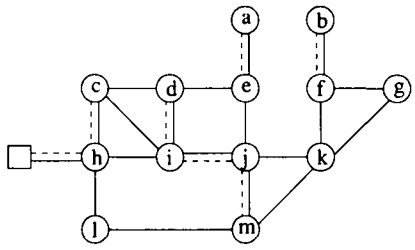


FIG. 1.129 -

Graphe pair G'.

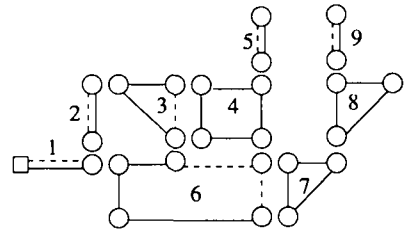


FIG. 1.130 -

Décomposition de G' en cycles.

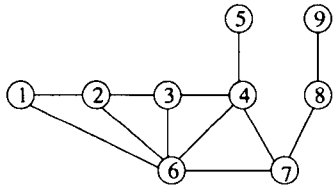


FIG. 1.131 -

Le graphe G'' sans dépôt associé à la décomposition en cycles.

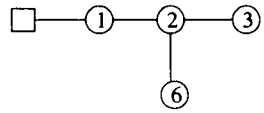


FIG. 1.132 -

Un arbre dans G'' contenant un dépôt.

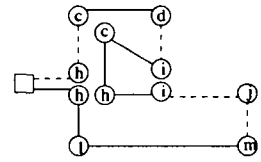


FIG. 1.133 -

La tournée correspondante dans G.

Le voisinage de la méthode de recuit simulé développée par Eglese consiste à modifier un ensemble d'arbres dans G'' ayant un dépôt comme racine. Les modifications prises en compte permettent par exemple de fusionner deux arbres (i.e. fusionner deux tournées), de créer un arbre nouveau (i.e. créer une nouvelle tournée) et de modifier un arbre sans changer les sommets qu'il couvre (i.e. modifier une tournée existante). Pour plus de détails le lecteur pourra se référer à [Egl94].

Une méthode tabou a été également développée par Eglese et al. [Egl96] pour résoudre un PTAC avec une contrainte additionnelle visant à limiter la durée d'une tournée. Etant donné une solution s du PTAC, une solution voisine est obtenue en déplaçant une portion de tournée soit vers une autre tournée, soit dans la même tournée mais ailleurs. Une portion de tournée appelée "section" par Eglese et al. est une séquence d'arêtes desservies consécutivement dans une tournée et reliées entre elles par des plus courtes chaînes. Une

"section" débute et se termine toujours par une arête desservie. Lorsqu'une "section" quitte une tournée, elle est remplacée dans la tournée par une plus courte chaîne reliant ses extrémités. Afin de limiter la taille du voisinage à examiner, seules les "sections" dont la longueur est inférieure à une valeur donnée sont considérées. Les tournées entières sont également des "sections" prises en compte.

Plus récemment, Belenguer et al. [Bel97a] ont développé une méthode tabou. Le passage d'une solution voisine à une autre se fait en déplaçant une arête (v_i, v_j) desservie dans une tournée T vers une tournée $T' \neq T$ contenant le sommet v_i ou le sommet v_j.

Nous avons également développé une méthode tabou (voir [Her2000]). Cette méthode intègre les différentes procédures décrites à la section 1.2.2.5. Nous ne faisons que mentionner cet algorithme dans cette section, il sera décrit de manière détaillée dans un prochain chapitre de cette thèse.

Citons encore Li [Li92] qui a développé avec un succès limité une méthode tabou et une méthode de recuit simulé pour résoudre un problème de salage de routes, et Greistorfer [Gre94] qui lui aussi a développé une méthode tabou pour le PPCC.

1.4 Résumé

Nous présentons dans cette section un tableau récapitulatif des divers types de problèmes de tournées sur les arcs que nous avons décrits. Dans ce tableau figurent également la complexité de chaque problème ainsi que quelques brefs commentaires concernant, soit la méthode de résolution utilisée, soit le problème lui-même. Ce tableau est une reproduction de celui présenté par Assad et Golden dans [Ass95].

Nous supposons toujours que nous travaillons sur un graphe $G = (V, E \cup A)$ fortement connexe (ou connexe si $A = \emptyset$). R est l'ensemble des clients. Le tiret (-) signifie que les arcs ou arêtes sont choisis arbitrairement (pour autant que G soit fortement connexe). La dernière colonne de ce tableau indique le numéro de la section où il est question du problème.

Problème	A	E	R	Complexité	Remarques	voir
PPC non orienté	$\emptyset$	-	$R = E$	$\mathbb{P}$	Résolu à l'aide d'un algorithme de couplage en $O(\|V\|^3)$.	1.2.1.1
PPC orienté	-	$\emptyset$	$R = A$	$\mathbb{P}$	Résolu à l'aide d'un problème de flots en $O(\|V\|^3)$.	1.2.1.2
PPC mixte	-	-	$R = E \cup A$	NP-dur	Peut être résolu polynomialement si G est pair.	1.2.1.3
PPC avec vent	$\emptyset$	-	$R = E$	NP-dur	Le coût des arêtes dépend du sens dans lequel elles sont traversées.	1.2.1.4
PPR non orienté	$\emptyset$	-	$R \subset E$	NP-dur	Peut être résolu polynomialement si G_R est connexe.	1.2.2.1
PPR orienté	-	$\emptyset$	$R \subset A$	NP-dur	Peut être résolu polynomialement si G_R est connexe.	1.2.2.2
PPR mixte	-	-	$R \subset A \cup E$	NP-dur	Peut être résolu polynomialement si G_R est connexe et pair.	1.2.2.3
SCP	-	$\emptyset$	$R \subset E$	NP-dur	Les arêtes de R ont un sens de service imposé.	1.2.2.4
PPCC non orienté	-	$\emptyset$	$R = E$	NP-dur	Tous les véhicules ont la même capacité.	1.3
PTAC non orienté	-	$\emptyset$	$R \subset E$	NP-dur	Tous les véhicules ont la même capacité.	1.3

TAB. 1.1 –

Tableau récapitulatif des différents problèmes traités dans le chapitre.

1.5 Conclusion

Nous avons décrit dans ce chapitre les principales méthodes heuristiques utilisées pour la résolution de PTA. Les problèmes que nous avons présentés sont des problèmes de base. Lorsqu'il s'agit de traiter des problèmes réels, des contraintes additionnelles doivent être prises en compte (limite sur la longueur ou la durée des tournées, routes de différentes catégories, dépôts multiples, etc.). Les méthodes que nous avons décrites ne peuvent alors pas s'appliquer directement à de tels problèmes. Elles peuvent cependant servir de base à des algorithmes tenant compte des contraintes additionnelles issues de la pratique. De telles extensions ont été utilisées et appliquées à des problèmes réels. Dror et al. par exemple [Dro87], [Dro97] ont résolu des problèmes dans lesquels des groupes de clients doivent être servis avant d'autres. Nous avons vu que Eglese dans [Egl94] traite un problème à dépôts multiples. D'autres problèmes pratiques ont été résolus notamment dans [Roy89], [Egl96], [Cha84]. Nous traiterons également dans un prochain chapitre de cette thèse un problème de confection d'horaires de bateaux que nous avons modélisé sous la forme d'un problème de tournées sur les arcs.

Chapitre 2

Nouveaux développements pour la résolution du PTAC

2.1 Introduction

Le PTAC (Problème de tournées sur les arcs avec capacité) est un des problèmes de tournées sur les arcs le plus général. Il englobe en effet le problème du postier rural et donc également le problème du postier chinois. Etant donné un ensemble de clients et une flotte homogène de véhicules basée en un lieu appelé le dépôt, le PTAC consiste à déterminer un ensemble de tournées de longueur totale minimum desservant les clients. Chaque tournée débute et s'achève au dépôt et la demande desservie au cours d'une tournée ne doit pas dépassser la capacité du véhicule qui la parcourt. Le PTAC est un problème NP-dur [Gol81]. De nombreuses heuristiques ont été développées pour tenter de le résoudre au mieux. Nous avons mentionné les principales au chapitre précédent. Parmi ces heuristiques, des méthodes de recherche locale ont été adaptées pour la résolution du PTAC au cours de ces dernières années. Citons notamment les travaux de Li [Li92], Eglese [Egl94], Greistorfer [Gre94] et plus récemment ceux de Belenguer, Benavent et Cognata [Bel97a] et ceux de Hertz et al. [Her2000].

Nous proposons dans ce chapitre de nouvelles adaptations de méthodes de recherche locale pour la résolution du PTAC. Les performances de ces méthodes seront analysées et comparées avec celles d'autres algorithmes. Pour cela, certains résultats figurant dans [Her2000] seront repris dans ce chapitre.

La première section du chapitre sera consacrée à la description générale des méthodes de recherche locale que nous avons utilisées. Dans une deuxième partie, nous traiterons du PTAC non orienté et mentionnerons la manière dont nous avons adapté à ce problème les différentes techniques décrites dans la première section. Dans la troisième section nous verrons comment les algorithmes que nous avons développés pour le cas non orienté du PTAC pourront être adaptés au cas orienté. Nous présenterons également dans cette section une nouvelle borne inférieure pour le PTAC orienté s'inspirant d'une borne proposée pour le cas non orienté par Benavent et al. [Ben92]. Nous terminerons ce chapitre par une conclusion résumant les principaux résultats obtenus et proposant quelques pistes pour de futures recherches dans le domaine.

2.2 Quelques méthodes de recherche locale

Les méthodes de recherche locale sont couramment utilisées dans le domaine de l'optimisation combinatoire. Leur caractère très général fait qu'elles ont été appliquées à de nombreux types de problèmes où le plus souvent elles se sont montrées très efficaces. De manière générale, elles consistent à appliquer successivement une suite de modifications locales à une solution initiale afin d'obtenir une solution finale de meilleure qualité.

Dans toute cette section, nous allons nous intéresser au problème d'optimisation suivant: $\min_{s \in S} f(s)$ où S est un ensemble fini contenant l'ensemble des solutions du problème et f est une fonction réelle définie sur S. L'ensemble S est également appelé espace des solutions.

Un des éléments clef d'une méthode de recherche locale est la notion de voisinage. Etant donné une solution $s \in S$, il est possible de la modifier de façon à obtenir une nouvelle solution s'. Nous noterons $s' = s \oplus m$ le passage de s à s' qui est effectué en appliquant la modification m à la solution s. Nous appellerons modification admissible, une modification m telle que si s est dans S alors $s' = s \oplus m$ se trouve dans S également. L'ensemble des modifications admissibles applicables à une solution s sera noté M_s. M_s peut donc être défini de la manière suivante: $M_s = \{m | s \oplus m \in S\}$.
Un voisinage $N(s)$ d'une solution $s \in S$ est l'ensemble des solutions pouvant être obtenues à partir de s à l'aide d'une modification appartenant à un ensemble $M \subseteq M_s$. En utilisant les notations que nous venons d'introduire, nous pouvons définir $N(s)$ de la manière suivante: $N(s) = \{s' | \exists\, m \in M : s' = s \oplus m\}$. Une solution $s' \in N(s)$ sera appelée solution voisine de s.

2.2.1 Méthode de descente

Du fait de sa simplicité, la méthode de descente est une des méthodes de recherche locale la plus utilisée. A chaque itération, on passe d'une solution $s \in S$, appelée solution courante, à une solution $s' \in N(s)$ de coût inférieur à celui de s. Ce passage se fait en appliquant à s une modification appartenant à M. La solution s' devient la solution courante de la prochaine itération, et ce procédé est répété jusqu'à ce qu'il ne soit plus possible d'améliorer la solution courante à l'aide d'une modification de M. La solution s' voisine de s peut être n'importe quelle solution de $N(s)$ dont le coût est inférieur à celui de s. En général cependant, la solution voisine de s de coût minimum est choisie pour devenir la prochaine solution courante.

Description générale d'une méthode de descente

Etape 0. Choisir une solution initiale $s_0 \in S$. Poser $s := s_0$.

Etape 1. Déterminer $s' \in N(s)$ telle que $f(s') = \min_{s'' \in N(s)} f(s'')$.

Etape 2. Si $f(s') < f(s)$, poser $s := s'$ et retourner à l'étape 1.
Sinon STOP, s est la solution fournie par l'algorithme.

Lorsque le voisinage $N(s)$ est de taille importante ou si l'évaluation de la qualité d'une modification est coûteuse du point de vue du temps de calcul, il est préférable de ne générer à chaque itération qu'une partie N^* de $N(s)$.
Lorsqu'il existe plusieurs solutions voisines de même coût minimum, une de ces solutions est choisie soit en fonction d'un critère spécifié, soit aléatoirement.

Le principal défaut de la méthode de descente réside dans le fait qu'il est possible de se retrouver piégé en un minimum local ayant une valeur très éloignée de celle de la solution optimale. La figure ci-dessous illustre un tel comportement. Les flèches indiquent le passage d'une solution à une autre.

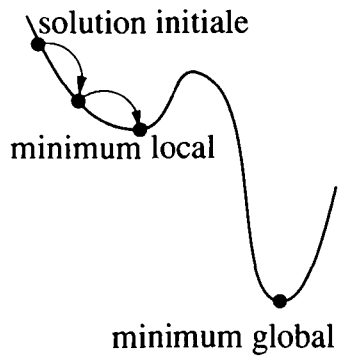


FIG. 2.1 -

Pour éviter le genre de situations représentées à la figure 2.1, il est nécessaire d'accepter dans certains cas de passer d'une solution s à une solution voisine s' moins bonne. C'est ce qui est fait dans le cas de la méthode tabou.

2.2.2 Méthode tabou

La méthode tabou a été proposée pour la première fois par Glover dans les années 80 (voir par exemple [Glo89] et [Glo90]). Une idée similaire a été développée indépendamment par Hansen (cf. [Han86]). Un peu plus compliquée que la méthode de descente, les principes sur lesquels elle repose restent néanmoins simples. Elle s'est avérée très performante pour de nombreux types de problèmes d'optimisation combinatoire. Elle offre de plus l'avantage de pouvoir se combiner facilement avec d'autres méthodes.
La méthode tabou peut être vue comme une amélioration de la méthode de descente. Le passage de la solution courante s à une solution voisine s' est autorisé lorsque $f(s') \geq f(s)$. Comme la valeur de la solution courante peut se détériorer d'une itération à l'autre, la meilleure solution visitée s_{best} est mémorisée.
Il est donc possible avec la méthode tabou de quitter un optimum local pour visiter d'autres solutions. Cependant, le fait d'accepter une solution s' de moins bonne qualité que la solution courante peut engendrer des risques de cyclage. En effet, si à une itération nous passons d'une solution s à une solution voisine $s' = s \oplus m$ moins bonne, il se peut qu'à l'itération suivante la solution voisine que nous allons visiter soit la solution s à partir de laquelle s' a été obtenue. Si un tel cas se présente, au cours des itérations suivantes, seules les deux solutions s et s' sont visitées alternativement et aucune autre nouvelle solution n'est plus examinée. L'algorithme cycle alors autour d'un minimum local dont

la valeur peut être éloignée de celle de la solution optimale. Nous retrouvons le même problème que celui rencontré avec la méthode de descente et que justement nous voulions éviter.
La figure 2.2 illustre le phénomène de cyclage. Le cycle est représenté en traitillés.

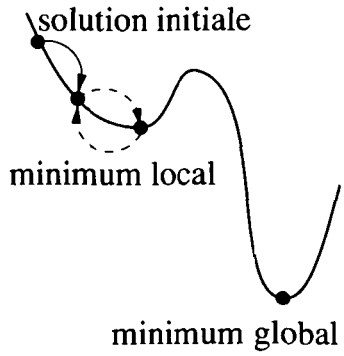


FIG. 2.2 -

Pour éviter le risque de cyclage, une idée consisterait à mémoriser les dernières solutions visitées et à en interdire l'accès. Une telle idée n'est pas réalisable en pratique car le stockage des solutions nécessite généralement une quantité considérable de mémoire. Une solution plus ingénieuse et ne posant pas de problème de mémoire consiste à stocker dans une liste bornée de type FIFO la modification inverse, m^{-1}, de celle qui vient d'être effectuée pour passer de s à s'. Toutes les modifications contenues dans cette liste sont interdites et ne peuvent plus être effectuées. Cette liste est appelée liste tabou, nous la désignerons par la lettre $\mathcal{T}$. La taille (ou longueur) de la liste tabou, notée θ, correspond au nombre d'itérations durant lesquelles une modification ne peut pas être appliquée.

Le fait d'interdire d'effectuer une modification durant un certain nombre d'itérations limite le risque de cyclage, mais peut également parfois nous empêcher d'accéder à des solutions de bonne qualité qui n'ont pas encore été visitées. Il est donc utile, dans certaines situations, de pouvoir lever provisoirement (pour une itération seulement) le statut tabou d'une modification stockée dans $\mathcal{T}$. L'ensemble des conditions devant être satisfaites pour que le statut tabou d'une modification soit levé est appelé critère d'aspiration. Le critère d'aspiration le plus couramment utilisé consiste à lever le statut tabou d'une modification $m \in \mathcal{T}$ si la solution $s' = s \oplus m$ voisine de la solution courante s est telle que $f(s') < f(s_{best})$. En d'autres termes, une modification appartenant à la liste tabou peut être effectuée si elle permet d'atteindre une solution meilleure que la meilleure solution rencontrée jusqu'ici.

La méthode tabou ne s'arrêtant plus au premier optimum local rencontré, un critère d'arrêt doit être défini. Plusieurs choix sont possibles: arrêt après un nombre fixé d'itérations, arrêt après un nombre fixé d'itérations sans amélioration de s_{best}, arrêt dès qu'un temps limite de calcul est dépassé, etc.

Description générale d'une méthode tabou

Etape 0. *(Initialisation)*
Choisir une solution initiale $s_0 \in S$.
Poser $s_{best} := s_0$, $s := s_0$ et $\mathcal{T} = \emptyset$.

Etape 1. *(Critère d'arrêt)*
Si le critère d'arrêt est satisfait, STOP, s_{best} est la solution fournie par l'algorithme.

Etape 2. *(Itération courante)*
Soit $N_{\mathcal{T}}(s)$ l'ensemble des solutions $s'' = s \oplus m''$ de $N(s)$ telles que soit:

1) $m'' \notin \mathcal{T}$

2) $m'' \in \mathcal{T}$ et le critère d'aspiration est satisfait

Déterminer $s' \in N_{\mathcal{T}}(s)$ telle que $f(s') = \min_{s'' \in N_{\mathcal{T}}(s)} f(s'')$.
Mettre à jour la liste tabou $\mathcal{T}$ et poser $s := s'$.

Etape 3. *(Mise à jour)*
Si $f(s') < f(s_{best})$, poser $s_{best} := s$.
Aller à l'étape 1.

Comme dans le cas de la méthode de descente, lorsque la taille de $N(s)$ est très importante ou lorsque l'évaluation du coût des solutions voisines nécessite un temps de calcul élevé, seule une portion N^* du voisinage de la solution courante est examinée. L'ensemble $N_{\mathcal{T}}(s)$ est alors défini parmi les solutions se trouvant dans N^* et non plus parmi celles contenues dans $N(s)$.

2.2.3 Méthode de descente à voisinage variable

Les méthodes de recherche à voisinage variable ont été formellement introduites par Hansen et Mladenović (voir [Han97] et [Mla97]). L'idée principale apparaissant dans ces méthodes consiste à considérer successivement plusieurs types de voisinage. De telles approches se sont avérées performantes pour la résolution de problèmes comme le problème du voyageur de commerce en optimisation combinatoire, ou la recherche de graphes extrémaux en théorie des graphes (voir [Han97]).
Nous allons décrire ici uniquement la méthode de descente à voisinage variable. Pour plus de détails concernant les méthodes de recherche à voisinage variable, le lecteur pourra se référer à [Han97].

Soit un ensemble fini de différents types de voisinage, $\mathcal{N} = \{N_1, N_2, ..., N_K\}$. Soit $s \in S$, $N_k(s)$ est le $k^{\text{ème}}$ type de voisinage de s $(k = 1, ..., K)$. La méthode de descente à voisinage variable consiste à effectuer successivement une série de méthodes de descente en changeant de type de voisinage entre chaque application de la méthode de descente.

Description générale d'une méthode de descente à voisinage variable

Etape 0. *(Initialisation)*
Choisir une solution initiale $s_0 \in S$.
Poser $s := s_0$.

Etape 1. Poser k=1 et $s'' := s$.
Tant que $k \leq K$ faire

Soit s' la solution fournie par la méthode de descente utilisant le voisinage N_k et en prenant s comme solution initiale.
Si $f(s') < f(s)$ poser $s := s'$.
Poser k:=k+1.

Etape 2. Si $f(s) < f(s'')$ aller à l'étape 1, sinon s est la solution fournie par l'algorithme.

Comme un minimum local pour un type de voisinage ne l'est pas forcément pour un autre, le risque de se retrouver piégé en un minimum local éloigné de l'optimum global peut être évité (voire figure 2.3). Le choix des types de voisinage utilisés de même que l'ordre dans lequel ils sont considérés peuvent jouer un rôle très important et influencer la qualité des solutions obtenues.

La figure 2.3 illustre le fonctionnement de la méthode de descente à voisinage variable. Partant de la solution initiale s_0, et utilisant le voisinage N_1, nous obtenons s_1 comme nouvelle solution courante. Cette solution est un minimum local par rapport au voisinage N_1. Nous considérons alors le voisinage N_2. La solution s_1 n'est pas un optimum local par rapport à ce voisinage et une nouvelle solution, s_2, peut être atteinte.

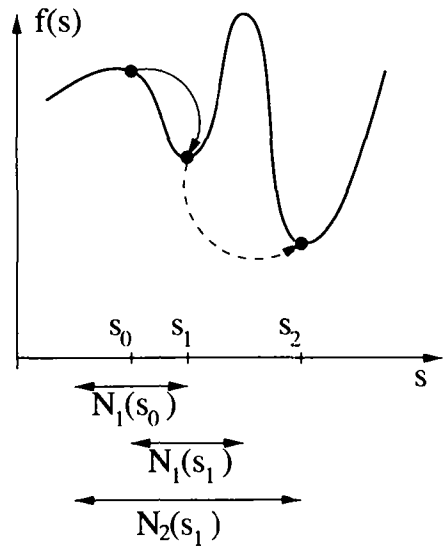


FIG. 2.3 –

2.3 Le PTAC non orienté

Dans cette section, nous allons tout d'abord décrire une procédure de post-optimisation que nous avons développée pour le cas non orienté du PTAC. Nous présenterons ensuite une adaptation de la méthode tabou et de la méthode de descente à voisinage variable pour la résolution du PTAC non orienté. Finalement, une troisième partie sera consacrée à la présentation de résultats numériques.

Rappelons qu'un PTAC non orienté est défini sur un graphe $G = (V, E)$ non orienté connexe, que R est l'ensemble des arêtes à desservir et que W est la capacité des véhicules. Une solution du PTAC dans G est un ensemble de tournées satisfaisant les trois propriétés suivantes:

1) Chaque tournée part du dépôt et revient au dépôt.

2) La demande desservie dans une tournée ne dépasse pas la capacité W de chaque véhicule.

3) Chaque arête de R est desservie exactement une fois et par un seul véhicule.

Une solution satisfaisant les propriétés **1)**, **2)** et **3)** sera appelée solution admissible. Une solution n'est pas admissible lorsqu'une des trois contraintes ci-dessus n'est pas respectée. Comme toutes les solutions que nous allons considérer dans cette section satisferont les contraintes **1)** et **3)**, nous parlerons de solution non admissible uniquement pour désigner une solution ne respectant pas la contrainte de capacité (contrainte **2)**).

2.3.1 Procédures de base

Nous avons utilisé dans les méthodes que nous avons développées plusieurs procédures de base. Ces procédures peuvent être classées en deux catégories. La première catégorie comprend les procédures s'appliquant à une tournée d'une solution du PTAC. Il s'agit des procédures développées par Hertz et al. [Her99] et décrites à la section 1.2.2.5. La deuxième catégorie n'est composée que d'une seule procédure. C'est une procédure de post-optimisation s'appliquant à l'ensemble des tournées composant une solution du PTAC. Dans cette procédure, les tournées de la solution que l'on souhaite améliorer sont regroupées en une seule tournée géante passant au moins une fois par chaque arête de R. Le procédé suivant est ensuite répété: on tente de modifier l'ordre des arêtes de la tournée géante avant de la découper de manière à obtenir une nouvelle solution. On s'arrête dès qu'une solution de valeur inférieure à la solution de départ est obtenue ou après un certain nombre de répétitions. Cette procédure de post-optimisation est décrite de manière détaillée à la page suivante.

Algorithme: POST-OPT

ENTREE: Une solution s (admissible ou non) du PTAC dans $G = (V, E)$.

SORTIE: Une solution $\bar{s}$ de valeur au pire égale à celle de s.

Etape 0. Poser $\bar{s} := s$ et $\gamma = 0$.

Etape 1. Mettre bout à bout chacune des tournées de s. Soit T la tournée ainsi obtenue. T dessert l'ensemble des arêtes de R.
Déterminer une tournée T' desservant les arêtes de R en appliquant RACCOURCIR à T.
Choisir un sommet v_i apparaissant plusieurs fois dans T'.

Etape 2. Choisir une orientation de T'. Inverser le sens de parcours de tous les tronçons de T' ayant v_i comme extrémités.

Etape 3. Construire une solution admissible s' en appliquant DECOUPER_TOURNEE sur T' (voir 1.3.2.1).
Appliquer RACCOURCIR sur chacune des tournées de s'.
Si la valeur de s' est inférieure à celle de s poser $\bar{s} := s'$, STOP.

Etape 4. Si $\gamma = 100$ ou si v_i est le seul sommet apparaissant plus d'une fois dans T', STOP.
Sinon poser $\gamma := \gamma + 1$ et choisir un sommet $v_j \neq v_i$ apparaissant plusieurs fois dans T'.

Etape 5. Poser $v_i := v_j$ et aller à l'étape 2.

2.3.2 Méthode tabou appliquée au PTAC non orienté

2.3.2.1 L'espace des solutions

Dans notre adaptation de la méthode tabou au PTAC, nous avons relaxé la contrainte de capacité (contrainte **2)**). L'espace des solutions S que nous utilisons est composé de tous les ensembles de tournées satisfaisant les contraintes **1)** et **3)** de la page précédente. Les violations de la contrainte **2)** seront pénalisées dans la fonction objectif. Nous noterons s_{best} la meilleure solution de S visitée et s^*_{adm} la meilleure solution admissible rencontrée au cours de l'exécution de la méthode tabou.

2.3.2.2 La fonction objectif

La fonction objectif est composée de deux termes. Le premier terme correspond à la longueur totale des tournées de la solution évaluée et le second est un terme pénalisant les éventuels dépassements de capacité. En notant $L(s)$ la longueur totale d'une solution $s \in S$ et $Dep(s)$ la somme des dépassements de capacité intervenant sur chacune des tournées de s, la fonction objectif que nous allons minimiser est la suivante:

$$f(s) = L(s) + \alpha Dep(s)$$

La variable $\alpha \in \mathbb{R}_+^*$ est un coefficient de pénalité. Ce coefficient s'ajuste au cours de l'exécution de la méthode tabou. Initialement, sa valeur est égale à la longueur moyenne d'une plus courte chaîne reliant le dépôt et l'extrémité d'une arête de R. Un tel choix ne favorise pas la visite de solutions non admissibles lors des premières itérations de la méthode tabou. Toutes les β itérations, la valeur de α est mise à jour. Si les β dernières solutions visitées sont toutes admissibles, α est diminué de moitié. Si les β dernières solutions visitées sont non admissibles, la valeur de α est doublée. Dans tous les autres cas, la valeur de α reste inchangée. Nous avons utilisé 5 comme valeur de β. D'autres choix ont été testés, mais ils ne se sont pas avérés plus performants.

2.3.2.3 La solution initiale

La solution initiale de notre méthode tabou est obtenue en utilisant une méthode constructive à deux phases basée sur une stratégie "Tournée & Groupes" (cf. 1.3.2.1). Dans un premier temps, nous résolvons un problème de postier rural dans G en utilisant l'algorithme PPR_NON_ORIENTE de Frederickson (cf. 1.2.2.1). Nous obtenons alors une tournée contenant au moins une fois toutes les arêtes de R. Cette tournée est raccourcie en utilisant la procédure RACCOURCIR. Elle est ensuite découpée en tournées admissibles à l'aide de la procédure DECOUPER_TOURNEE décrite à la section 1.3.2.1. Nous obtenons ainsi une solution admissible pour le PTAC non orienté. La procédure de post-optimisation POST-OPT est appliquée à cette solution. La solution fournie par POST-OPT est la solution initiale s_0 de notre méthode tabou. Nous noterons par F_0 le coût de cette solution.

2.3.2.4 Le voisinage

Le voisinage $N(s)$ d'une solution $s \in S$ est l'ensemble des solutions de S pouvant être obtenues à partir de s en déplaçant le service d'un client d'une tournée vers une autre. Cette autre tournée peut être une nouvelle tournée ne desservant que le client déplacé.
Pour supprimer le service d'un client d'une tournée, la procédure SUPPRIMER_CLIENT est appliquée à la tournée (cf. 1.2.2.5). L'insertion d'un client dans une tournée se fait en utilisant la procédure AJOUTER_CLIENT (cf. 1.2.2.5). Lorsqu'une nouvelle tournée ne desservant que le client déplacé est créée, celui-ci est relié au dépôt à l'aide de plus courtes chaînes.

L'exemple qui suit illustre le voisinage utilisé dans la méthode tabou que nous avons développée. La figure 2.4 représente le graphe de départ. Toutes les arêtes sont à desservir et la demande de chacune d'elles est de 1. Les nombres situés au-dessus des arêtes correspondent à leur coût. La capacité des véhicules vaut 3. Les arêtes en traitillés sont les arêtes qui sont traversées par un véhicule sans être desservies.
Le passage de s à s' se fait en déplaçant le service de l'arête (c, d). Initialement servie dans la tournée T_2, cette arête est insérée dans T_3 pour y être desservie. Les figures 2.5 et 2.6 représentent respectivement la solution s et sa solution voisine s'.

Exemple:

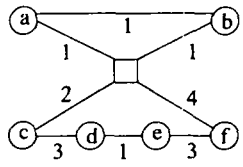


FIG. 2.4 –

Graphe initial G.

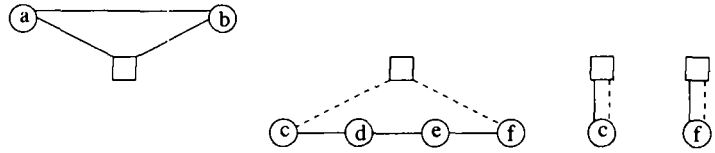


FIG. 2.5 –

Solution s composée de 4 tournées: T_1, T_2, T_3 et T_4.

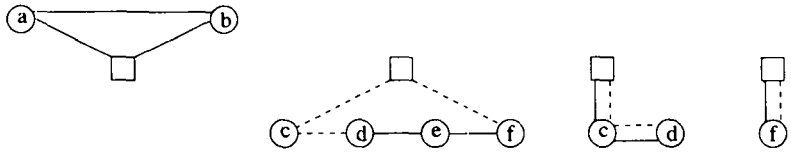


FIG. 2.6 –

Solution s' voisine de s obtenue en déplaçant le service de (c,d) de T_2 vers T_3.

2.3.2.5 Restriction de la taille du voisinage: définition de N^*

Lorsque le nombre de clients à traiter est important et que de nombreux véhicules sont utilisés, la taille du voisinage d'une solution s peut devenir très importante. Dans de tels cas, il n'est pas réaliste de vouloir calculer le coût de l'ensemble de toutes les solutions voisines de s à chaque itération. Il est alors nécessaire de se restreindre à l'examen d'une portion N^* de $N(s)$.
Nous avons défini N^* en limitant le nombre de tournées dans lesquelles un client peut se déplacer. Pour cela, une notion de distance entre les arêtes est introduite. La distance $\mathcal{D}_{(i,j),(k,l)}$ entre deux arêtes (v_i, v_j) et (v_k, v_l) de E est définie de la manière suivante:

$$\mathcal{D}_{(i,j),(k,l)} = \min\{sc_{ik}, sc_{il}, sc_{jk}, sc_{jl}\}$$

où sc_{rt} est la longueur d'une plus courte chaîne reliant v_r à v_t dans G.

Soit s une solution de S composée des tournées $T_1, ..., T_k$. Soit (v_i, v_j) un client desservi dans la tournée T_u de s $(1 \leq u \leq k)$. Le service de (v_i, v_j) peut être déplacé dans une tournée différente de T_u si et seulement si:

1) cette tournée contient au moins un client (v_k, v_l), desservi ou non, tel que: $\mathcal{D}_{(i,j),(k,l)} \leq \mathcal{D}_{min}$

2) cette tournée est "vide", c'est-à-dire si une nouvelle tournée est créée pour desservir (v_i, v_j).

La variable $\mathcal{D}_{min}$ est un paramètre permettant de définir une portion N^* de $N(s)$ plus ou moins importante. Nous avons fixé la valeur de ce paramètre égale à la longueur moyenne d'une arête de E. D'autres valeurs de $\mathcal{D}_{min}$ ont été testées. Elles n'ont pas conduit à de

meilleurs résultats que le choix que nous venons de mentionner. Nous avons également envisagé d'autres manières de définir N^*. L'une d'elles consiste à autoriser le déplacement d'un client (v_i, v_j) uniquement dans une tournée contenant un des plus proches voisins de v_i ou de v_j. Aucune de ces approches ne s'est avérée meilleure que celle qui est présentée dans cette section.

Exemple:

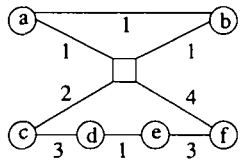


FIG. 2.7 -

Graphe initial G.

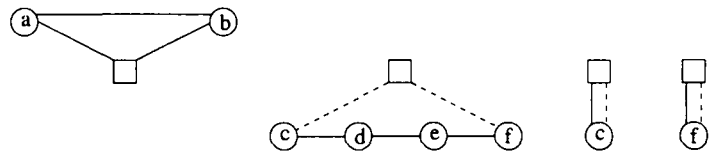


FIG. 2.8 -

Solution s composée de 4 tournées: T_1, T_2, T_3 et T_4.

Nous avons repris le graphe de départ et la solution s de la page précédente. La valeur de $\mathcal{D}_{min}$ est alors de 2. Le déplacement de l'arête (d, e) desservie dans T_2 vers la tournée T_1 n'est pas autorisé. En effet, aucun des clients se trouvant dans T_1 n'est à une distance de (d, e) inférieure ou égale à la longueur moyenne d'une arête de E qui vaut 2.

Pour comparer entre elles les solutions voisines de s se trouvant dans N^*, nous avons, comme dans [Gen94], utilisé une fonction qui diffère légèrement de la fonction objectif. Soit s'_{ij} une solution de N^* obtenue à partir de s en déplaçant le service de l'arête (v_i, v_j). La fonction $\tilde{f}$ est définie de la manière suivante:

$$\tilde{f}(s'_{ij}) = \begin{cases} f(s'_{ij}) & \text{si } f(s'_{ij}) < f(s) \\ f(s'_{ij}) + \Delta\sqrt{k}\eta\phi_{ij} & \text{si } f(s'_{ij}) \geq f(s) \end{cases} \quad \text{où}$$

- Δ est la plus grande différence en valeur absolue entre deux solutions courantes consécutives (obtenues au cours de deux itérations successives).
- k est le nombre de véhicules de la solution s'_{ij}.
- η est un facteur d'échelle fixé à 0.01.
- ϕ_{ij} est le nombre de fois que le service de l'arête (v_i, v_j) a été déplacé divisé par le nombre d'itérations effectuées.

La fonction $\tilde{f}$ est introduite pour assurer une certaine diversification des solutions visitées. Les clients les plus fréquemment déplacés sont pénalisés. Le terme $\Delta\sqrt{k}\eta$ sert à ajuster la fréquence ϕ_{ij} à la fonction f. Ce principe a été initialement suggéré par Glover [Glo89].

2.3.2.6 La liste tabou

A chaque itération, le service d'un client (v_i, v_j) de la solution courante est déplacé d'une tournée dans une autre. Afin d'éviter les risques de cyclage, (v_i, v_j) ne peut pas être à nouveau desservi dans sa tournée d'origine avant que ne se soit écoulé un nombre fixé

d'itérations θ. θ est choisi aléatoirement, à chaque itération, dans l'intervalle $[\theta_{min}, \theta_{max}]$. Comme dans l'algorithme TABUROUTE développé pour le PTSC par Gendreau et al. [Gen94] et comme suggéré par Glover et Laguna dans [Glo93], nous avons choisi $\theta_{min} = 5$ et $\theta_{max} = 10$.

2.3.2.7 Le critère d'aspiration

Une modification de la solution courante est le déplacement du service d'un client d'une tournée à une autre. Le statut tabou d'une modification est levé si cette modification engendre une solution de coût inférieur à $f(s_{best})$, ou si la solution obtenue grâce à cette modification est une solution admissible de coût inférieur à $f(s^*_{adm})$.

Nous noterons par $N_{\mathcal{T}}(s)$ l'ensemble des solutions $s'' \in N^*$, voisines de s telles que:

1) s'' est obtenue à l'aide d'une modification non tabou.

2) s'' est obtenue à l'aide d'une modification tabou et s'' satisfait le critère d'aspiration.

2.3.2.8 Le critère d'arrêt

Le premier critère d'arrêt que nous avons utilisé consiste à stopper lorsque la valeur de s^*_{adm} égale celle d'une borne inférieure $\underline{F}$. Lorsque ce critère interrompt l'exécution de notre algorithme, cela signifie que nous avons trouvé une solution optimale du problème traité.
La valeur de $\underline{F}$ est égale au maximum des trois bornes inférieures CPA, $LB2'$ et $NDLB'$ décrites dans la littérature. La première borne, CPA, est proposée par Belenguer et Benavent dans [Bel97b]. Elle est basée sur une méthode de plans sécants. Les deux autres bornes, $LB2'$ et $NDLB'$, sont des versions très légèrement modifiées des bornes $LB2$ (voir Benavent et al. [Ben92]) et $NDLB$ (voir Hirabayashi et al. [Hir92]). La modification apportée à ces deux bornes concerne le calcul d'une borne inférieure sur le nombre de véhicules nécessaires au service de l'ensemble des clients. Nous avons utilisé la borne inférieure LR proposée par Martello et Toth [Mar90] pour le "Bin Packing Problem" au lieu de la valeur $\lceil (\sum_{(v_i,v_j)\in R} q_{ij})/W \rceil$.

Le deuxième critère d'arrêt stoppe l'exécution de la méthode tabou dès que le nombre μ d'itérations consécutives sans amélioration de s_{best} ou de s^*_{adm} atteint une valeur ρ. La valeur de ρ est un paramètre que nous avons fixé à 100.

2.3.2.9 Algorithme CARPET

Nous avons appelé CARPET notre adaptation de la méthode tabou à la résolution du PTAC. Cet algorithme est également présenté dans [Her2000]. Il est décrit de manière détaillée à la page suivante.

Algorithme: CARPET

ENTREE: Un graphe $G = (V, E)$ et un sous-ensemble R de E.

SORTIE: Une solution admissible s^*_{adm} du PTAC dans G.

Etape 0. *(Initialisation)*
Poser le compteur d'itérations $i := 0$, poser $\mu := 0$ et $\rho := 100$.
Construire une solution initiale s_0 selon la méthode décrite en 2.3.2.3.
Poser $s := s_0$, $s_{best} := s_0$ et $s^*_{adm} := s_0$. La liste tabou $\mathcal{T}$ est vide.
Initialiser α comme indiqué en 2.3.2.2.

Etape 1. *(Critères d'arrêt)*
Si $\mu = \rho$, ou si $f(s^*_{adm}) = \underline{F}$, alors STOP. La solution s^*_{adm} est la solution fournie par CARPET.

Etape 2. *(Application de POST-OPT)*
Si $\mu \geq 10$, appliquer POST-OPT à s^*_{adm}. Soit $\bar{s}$ la solution fournie par POST-OPT. Si $f(\bar{s}) < f(s^*_{adm})$, poser $s := \bar{s}$, $i = i + 1$, et aller à l'étape 4.

Etape 3. *(Itération courante)*
Poser $i = i + 1$.

a) Déterminer la solution $s' \in N_{\mathcal{T}}(s)$ telle que: $\tilde{f}(s') = \min_{s'' \in N_{\mathcal{T}}(s)} \tilde{f}(s'')$.
Soit (v_r, v_j) l'arête dont le service a été déplacé pour passer de s à s'. Si s n'est pas admissible aller en c).

b) Si $f(s) < f(s')$ et si SUPPRIMER&AJOUTER_CLIENT n'a pas été utilisée à l'itération précédente, appliquer cette procédure sur chacune des tournées de s. Soit s'' la solution ainsi obtenue.
Si $f(s'') < f(s)$, poser $s := s''$ et aller à l'étape 4.

c) Poser $s := s'$ et interdire le retour du service de (v_r, v_j) dans sa tournée d'origine pour les θ prochaines itérations.

Etape 4. *(Mises à jour)*
Si s est admissible et $f(s) < f(s^*_{adm})$, poser $s^*_{adm} := s$ et poser $\mu := 0$.
Si $f(s) < f(s_{best})$ poser $s_{best} := s$ et poser $\mu := 0$.
Si s^*_{adm} et s_{best} n'ont pas été modifiées, poser $\mu := \mu + 1$.
Si i est un multiple de β, mettre à jour le coefficient de pénalité α.
Aller à l'étape 1.

Deux procédures de post-optimisation sont insérées dans l'algorithme que nous venons de décrire. La procédure POST-OPT est utilisée si 10 itérations ou plus se sont écoulées sans modification ni de s_{best} ni de s^*_{adm}. Elle est appliquée à s^*_{adm}.

La deuxième procédure de post-optimisation intégrée à CARPET est la procédure SUPPRIMER&AJOUTER_CLIENT (voir 1.2.2.5). Elle est utilisée uniquement lorsque les trois

conditions suivantes sont remplies:

- la solution courante s est admissible
- $f(s') > f(s)$
- SUPPRIMER&AJOUTER_CLIENT n'a pas été appliquée à l'itération précédente

Dans ce cas, avant de choisir s' comme prochaine solution courante, on tente d'améliorer s à l'aide de SUPPRIMER&AJOUTER_CLIENT. Si cela est possible, alors la nouvelle solution courante est la solution retournée par SUPPRIMER&AJOUTER_CLIENT, sinon s' devient la nouvelle solution courante.

2.3.3 Méthode de descente à voisinage variable

2.3.3.1 L'espace des solutions

Contrairement à ce que nous avons fait dans le cas de la méthode tabou, nous allons définir notre espace des solutions S' comme étant l'ensemble des solutions admissibles du PTAC. Chaque solution que nous visiterons à l'aide de la méthode de descente à voisinage variable satisfera donc les contraintes **1)**, **2)** et **3)** décrites au début de la section 2.3.

2.3.3.2 La fonction objectif

La fonction objectif que nous allons chercher à minimiser est simplement la longueur totale $L(s)$ des tournées qui composent une solution $s \in S'$.

2.3.3.3 La solution initiale

La solution initiale de la méthode de descente à voisinage variable est identique à la solution initiale de la méthode tabou décrite en 2.3.2.3.

2.3.3.4 Les voisinages

Soit $\underline{m} = \lceil (\sum_{(v_i,v_j)\in R} q_{ij})/W \rceil$ une borne inférieure sur le nombre de véhicules nécessaires pour satisfaire la demande des clients, et soit $s \in S'$ une solution admissible.
Etant donné un nombre k fixé ($2 \leq k \leq \underline{m}$), le $k^{\text{ème}}$ type de voisinage de s, $N_k(s)$, est obtenu en remplaçant k tournées de s par un ensemble de tournées desservant les mêmes clients. Le principe utilisé pour modifier ces k tournées est le même que celui employé dans la procédure POST-OPT. Les k tournées sont regroupées en une grande tournée. L'ordre des arêtes de celle-ci est modifié, puis elle est découpée en une série de tournées admissibles. Ces dernières ne sont pas forcément au nombre de k, elles peuvent être plus ou moins nombreuses. Elles doivent seulement desservir les mêmes clients que les k tournées de départ. La procédure VOIS(k) décrite à la page suivante présente de manière détaillée le passage d'une solution s à une solution $s' \in N_k(s)$.

Remarque:

Nous aurions pu utiliser la borne LR pour le "Bin Packing Problem" au lieu de $\underline{m}$ pour estimer le nombre minimum de véhicules nécessaires. Cette borne est au moins aussi bonne que $\underline{m}$, mais, sur des instances de grande taille, elle nécessite des temps de calcul assez importants. C'est pourquoi nous avons préféré utiliser $\underline{m}$.

Algorithme: VOIS(k)

ENTREE: Une solution admissible s et un ensemble de k tournées de s, $\{T_{i_1}, T_{i_2}, ..., T_{i_k}\}$ desservant un ensemble $R_k \subseteq R$ de clients.

SORTIE: Une solution admissible s'.

Etape 1. Mettre bout à bout les tournées $T_{i_1}, ..., T_{i_k}$.
Soit T la tournée ainsi obtenue. T dessert l'ensemble des arêtes de R_k.
Déterminer une tournée T' desservant les arêtes de R_k en appliquant à T la procédure RACCOURCIR.

Etape 2. Choisir aléatoirement un sommet v_i apparaissant plusieurs fois dans T'.
Choisir une orientation de T'. Inverser le sens de parcours des tronçons de T' ayant v_i comme extrémités.

Etape 3. Construire un ensemble de tournées admissibles, $\{T'_{j_1}, ..., T'_{j_r}\}$, desservant les arêtes de R_k en appliquant DECOUPER_TOURNEE sur T'.
Appliquer RACCOURCIR sur chacune des tournées $T'_{j_1}, ..., T'_{j_r}$.
Poser $s' := s \setminus \{T_{i_1}, T_{i_2}, ..., T_{i_k}\} \cup \{T'_{j_1}, ..., T'_{j_r}\}$.

$N_k(s)$ est l'ensemble des solutions s' pouvant être obtenues en modifiant k tournées de s à l'aide de la procédure VOIS(k) ($2 \leq k \leq \underline{m}$). Une solution s' appartenant à $N_k(s)$ va différer de s par le tracé d'au plus k tournées ($2 \leq k \leq \underline{m}$). Pour une valeur faible de k, la solution s' sera très semblable à s alors que pour des valeurs élevées de k, la différence entre s' et s pourra être très importante.

Nous allons introduire encore un dernier type de voisinage, que nous appellerons N_1. Il s'agit simplement du voisinage utilisé pour la méthode tabou avec toutefois la restriction suivante: le service d'un client est déplacé d'une tournée vers une autre seulement si cela ne provoque pas de violation des contraintes de capacité. Cette restriction nous assure de toujours rester dans l'espace des solutions S' que nous avons défini pour la méthode de descente à voisinage variable.

Etant donné une solution admissible s, nous nous sommes défini une série de voisinages $N_1(s), ..., N_{\underline{m}}(s)$. Ces voisinages sont numérotés du plus "fin" au plus "grossier". Par "fin", nous entendons un voisinage qui ne modifie que très légèrement la solution courante (la majorité des tournées restent inchangées), alors que "grossier" signifie qu'une solution voisine peut être complètement différente de la solution courante (la plupart des tournées sont modifiées). L'avantage d'un voisinage "fin" est qu'à partir d'une solution initiale,

pratiquement n'importe quelle solution de S' peut être atteinte. Cependant, il faut parfois de nombreuses itérations avant d'atteindre une solution qui soit très différente de la solution initiale. Un voisinage "grossier" permet de visiter très rapidement des solutions différentes de la solution courante, mais il est par contre difficile d'accéder à certaines solutions, en particulier à des solutions très semblables à la solution courante.

2.3.3.5 Restriction de la taille des voisinages

Etant donné une solution $s \in S'$, nous noterons par $|s|$ le nombre de tournées qui la compose.

La partie N_1^* de $N_1(s)$ que nous avons considérée, est définie de manière identique à la restriction N^* du voisinage $N(s)$ utilisé pour la méthode tabou. Le service d'un client (v_i, v_j) est donc déplacé uniquement vers des tournées contenant des clients (desservis ou non) situés à une distance inférieure ou égale à $\mathcal{D}_{min}$ de (v_i, v_j). Il peut également être déplacé vers une tournée nouvellement créée.
La taille des voisinages $N_k(s)$ $(k = 2, ..., \underline{m})$ d'une solution s peut être très importante. Nous pouvons nous en faire une idée en remarquant que, le nombre d'ensembles différents de k tournées de s est égal à $\frac{|s|!}{(|s|-k)!k!}$ et que pour un ensemble de tournées $\{T_{i_1}, ..., T_{i_k}\}$ de s, il existe $k!$ façons différentes de les mettre bout à bout. Pour une valeur de k fixée, nous obtenons une taille de $N_k(s)$ qui est de l'ordre de $|s|^k$ et cela sans prendre en compte les modifications de T' pouvant intervenir à l'étape 2 de VOIS(k).
Une portion seulement des voisinages est donc examinée. La taille de la portion N_k^* du voisinage $N_k(s)$ est la même quel que soit le problème traité et quelle que soit la valeur de k, $(2 \leq k \leq \underline{m})$. Pour une valeur de k fixée, κ_{max} ensembles de k tournées sont générés aléatoirement. Pour un ensemble de k tournées donné $\{T_{i_1}, ..., T_{i_k}\}$, les étapes 2 et 3 de VOIS(k) sont répétées ν_{max} fois. En procédant de la sorte, $\kappa_{max} \cdot \nu_{max}$ solutions sont examinées à chaque itération. Le paramètre κ_{max} influence la diversité des solutions examinées alors que ν_{max} permet d'intensifier les recherches dans une zone précise de l'espace des solutions. Nous avons fixé les valeurs de κ_{max} et ν_{max} à 50. Ainsi, à chaque itération d'une méthode de descente, 2500 solutions sont évaluées. D'autres valeurs de κ_{max} et ν_{max} ont été testées, les meilleures performances ont été obtenues avec les valeurs que nous venons de mentionner.

2.3.3.6 Ordre des voisinages

Nous avons effectué des tests numériques en ordonnant les voisinages de trois manières différentes. Le premier ordre consiste à examiner les voisinages les plus "grossiers" en premier, puis les voisinages les plus "fins". L'ordre est alors le suivant: $N_{\underline{m}}, N_{\underline{m}-1}, ..., N_1$. Le deuxième ordre correspond à l'ordre inverse du premier et le dernier ordre est un ordre aléatoire. Nous présenterons les résultats de ces tests dans la partie consacrée aux résultats numériques.

2.3.3.7 Algorithme DVV

Nous avons appelé DVV notre adaptation au PTAC de la méthode de descente à voisinage variable.

Algorithme: DVV

ENTREE: Un graphe $G = (V, E)$ et un sous-ensemble R de E.

SORTIE: Une solution admissible s du PTAC dans G.

Etape 0. *(Initialisation)*
Construire une solution initiale s_0 selon la méthode décrite en 2.3.2.3.
Poser $s := s_0$.
Générer un ordre des voisinages: $N_{o_1}, N_{o_2}, ..., N_{o_{\underline{m}}}$.

Etape 1. Poser $k := 1$ et $s'' := s$.
Tant que $k \leq \underline{m}$ faire

> Effectuer une méthode de descente en examinant à chaque itération une portion $N^*_{o_k}$ du voisinage N_{o_k} et en utilisant s comme solution initiale.
> Soit s' la solution fournie par la méthode de descente.
> Si $L(s') < L(s)$ poser $s := s'$.
> Poser k:=k+1.

Etape 2. Si $L(s) < L(s'')$ aller à l'étape 1. Sinon s est la solution fournie par l'algorithme.

Une variante dans laquelle l'étape 2 est supprimée sera testée dans la section consacrée aux tests numériques. Dans cette variante, seule l'étape 1 est effectuée, et la solution s obtenue à la fin de cette étape est la solution fournie par l'algorithme.

2.3.4 Résultats numériques

Nous allons présenter les résultats obtenus à l'aide des méthodes que nous venons de décrire. Cette section sera composée de trois parties. Dans la première partie, nous présenterons les différents problèmes tests que nous avons utilisés. Une deuxième partie sera consacrée à la présentation de certains tests que nous avons effectués sur les algorithmes CARPET et DVV. Finalement, dans la dernière partie, nous comparerons CARPET et DVV à d'autres algorithmes utilisés pour la résolution du PTAC non orienté.

Dans les différents tableaux de résultats qui suivent, N_{opt} représente le nombre de solutions dont le coût égale celui de la borne inférieure $\underline{F}$ définie en 2.3.2.8. Il s'agit donc du nombre de solutions dont l'optimalité a pu être prouvée. La valeur de la meilleure solution admissible connue à ce jour pour un problème donné est notée F_{best}. N_{best} correspond au nombre de solutions dont la valeur est égale à celle de F_{best}. E_{moyen} et E_{max} représentent respectivement l'écart moyen et l'écart maximum (en %) soit par rapport à $\underline{F}$ soit par rapport à F_{best}. Nous préciserons chaque fois par rapport à laquelle de ces deux valeurs les écarts sont calculés.

2.3.4.1 Problèmes tests

Nous avons utilisé 3 groupes de problèmes tests. Les deux premiers sont des problèmes tirés de la littérature, le dernier est composé de problèmes que nous avons générés aléatoirement.

Problèmes de DeArmon

La première série de problèmes tests correspond aux instances générées par DeArmon [DeA81] et reprises par Golden et al. [Gol83], Pearn [Pea89], Belenguer et al. [Bel97a] et Hertz et al. [Her2000]. Initialement, 25 instances sont décrites dans [DeA81] ainsi que dans [Gol83]. Les problèmes 8 et 9 comportant des erreurs, tout comme dans [Pea89], [Bel97a], et [Her2000], nous ne les traiterons pas. Nous avons néanmoins conservé la numérotation originale, de 1 à 25, mais sans mentionner les problèmes 8 et 9.
Les instances de DeArmon comportent entre 7 et 27 sommets et le nombre de clients varie entre 11 et 55. Dans les 23 problèmes, l'ensemble des arêtes à desservir est égal à l'ensemble des arêtes (i.e. $R = E$).

Problèmes de Benavent

La deuxième catégorie de problèmes tests nous a été fournie par Benavent. Nous retrouvons également ces problèmes dans [Ben92], [Bel97a], [Bel97b] et [Her2000]. Il s'agit de 34 instances possédant entre 24 et 50 sommets et ayant un nombre de clients variant entre 34 et 97. Comme dans le cas des problèmes de DeArmon, nous avons $R = E$ pour chacune des 34 instances.

Problèmes générés aléatoirement

Nous avons construit une série de problèmes où contrairement aux problèmes de DeArmon et de Benavent, toutes les arêtes des graphes associés à ces problèmes ne sont pas forcément des clients. Ces problèmes ont été générés de la manière suivante:

> Les coordonnées des sommets du graphe sont générées aléatoirement selon une loi uniforme dans le carré $[0, 10]^2$. Le sommet le plus au centre parmi les sommets générés est choisi comme dépôt.
> Un cycle hamiltonien est construit aléatoirement pour assurer la connexité du graphe. Des arêtes, sélectionnées aléatoirement, sont ajoutées jusqu'à ce qu'une certaine densité, ϵ, soit atteinte. Le coût d'une arête reliant deux sommets correspond à la partie entière inférieure de la distance euclidienne entre ces sommets.
> Des arêtes sont choisies aléatoirement et insérées dans R jusqu'à ce qu'une certaine proportion de clients, ω, soit atteinte. La demande d'un client est choisie aléatoirement selon une loi uniforme dans l'intervalle $[1, 40]$.
> La capacité des véhicules W est choisie aléatoirement de telle sorte que $\underline{m}$ appartienne à $[2, 30]$.

Nous avons considéré des graphes à 20, 40 et 60 sommets. Nous avons choisi aléatoirement la densité ϵ du graphe dans les intervalles $[0.1, 0.3]$, $[0.4, 0.6]$ et $[0.7, 0.9]$. La proportion de clients ω a été choisie aléatoirement dans $[0.1, 0.3]$, $[0.4, 0.6]$ et $[0.8, 1]$. En combinant chacun de ces paramètres, nous avons obtenu 27 types de problèmes différents. Pour

chaque type de problèmes, 10 instances ont été générées. Le nombre de clients de ces 270 instances varie entre 3 et 1562.

2.3.4.2 Tests numériques

CARPET:

Nous avons testé l'influence de la procédure de post-optimisation POST-OPT sur les performances de CARPET. Pour cela, nous avons traité les 23 problèmes de DeArmon et les 34 de Benavent avec une variante de CARPET utilisant POST-OPT et une variante ne faisant pas appel à POST-OPT (sauf pour la construction de la solution initiale). La première variante correspond à l'algorithme décrit en 2.3.2.9. La deuxième variante correspond à ce même algorithme mais où l'étape 2 est supprimée. Pour chacune de ces variantes, les valeurs des différents paramètres sont celles que nous avons spécifiées dans la section 2.3.2. Les écarts sont calculés par rapport à F_{best}.

	CARPET	
	avec POST-OPT	sans POST-OPT
E_{moyen} (en %)	0.63	1.73
E_{max} (en %)	5.14	6.29
Temps moyen [s]	23.07	15.07
N_{opt}	33	18
N_{best}	38	22

TAB. 2.1 –

Impact de la procédure POST-OPT sur le comportement de CARPET.

Le tableau 2.1 montre que l'utilisation de POST-OPT dans CARPET permet d'améliorer la qualité des solutions obtenues. De plus, l'utilisation de cette procédure permet d'atteindre un nombre bien plus important de solutions optimales. Le temps moyen de calcul augmente d'un peu plus de 50% lorsque POST-OPT est utilisée.
Le voisinage utilisé dans CARPET est un voisinage "fin". Au plus deux tournées sont modifiées lorsqu'on passe d'une solution courante à une autre. La procédure POST-OPT, lorsqu'elle parvient à améliorer la solution sur laquelle elle est appliquée, peut modifier cette solution de manière importante (toutes les tournées peuvent être changées). Elle peut ainsi contribuer à une meilleure exploration de l'espace des solutions. Cela peut expliquer la différence importante du nombre de solutions optimales obtenues par CARPET avec et sans POST-OPT.

Dès à présent lorsque nous parlerons de CARPET, nous ferons référence à la variante qui utilise la procédure POST-OPT.

DVV:

Nous avons comparé les résultats obtenus par DVV en ordonnant les voisinages de trois manières différentes. Ces ordres, O1, O2, O3, sont définis de la façon suivante:

O1 : $N_{\underline{m}}, N_{\underline{m}-1}, ..., N_1$

O2 : $N_1, ..., N_{\underline{m}-1}, N_{\underline{m}}$

O3 : ordre aléatoire des $\underline{m}$ voisinages

Ces trois ordres ont été testés sur les problèmes de DeArmon et de Benavent. Les résultats obtenus sont présentés dans le tableau 2.2. E_{max} et E_{moyen} sont respectivement l'écart maximum et l'écart moyen par rapport à F_{best}.

	DVV		
	O1	O2	O3
E_{moyen} (en %)	0.37	0.81	0.68
E_{max} (en %)	2.89	5.14	4.21
Temps moyen [s]	21.47	21.39	21.72
N_{opt}	34	29	32
N_{best}	40	32	34

TAB. 2.2 -

Influence de l'ordre des voisinages sur le comportement de DVV.

L'ordre O1 est celui qui permet d'obtenir les meilleurs résultats. Cependant, les ordres O2 et O3 bien que moins bons fournissent tout de même des solutions qui sont en moyenne de très bonne qualité. Le nombre de solutions optimales atteintes avec ces deux ordres reste également important.

Nous avons également testé une version de DVV dans laquelle l'étape 2 est supprimée (voire 2.3.3.7). Cette version, que nous appellerons DVV2, consiste alors simplement à effectuer successivement les $\underline{m}$ méthodes de descente associées à chacun des voisinages que nous avons définis. Le tableau 2.3 contient les résultats obtenus à l'aide de cette variante. L'ordre des voisinages utilisé est l'ordre O1.

	DVV2
E_{moyen} (en %)	0.41
E_{max} (en %)	2.89
Temps moyen [s]	14.67
N_{opt}	34
N_{best}	39

TAB. 2.3 -

Résultats obtenus à l'aide de DVV2.

Les résultats obtenus avec DVV2 sont quasiment identiques du point de vue de la qualité à ceux obtenus à l'aide de DVV. En fait, sur les 57 problèmes traités, nous obtenons des résultats différents pour un problème uniquement. Pour les 56 autres, les coûts des

solutions fournies par DVV2 et DVV sont identiques. En utilisant DVV2, les temps moyens d'exécution sont réduits d'environ 30%. Compte tenu de ces résultats, seule la variante DVV2 sera utilisée par la suite. L'ordre des voisinages sera toujours l'ordre O1.

2.3.4.3 Comparaisons

Les algorithmes CARPET et DVV2 vont être comparés à d'autres méthodes figurant dans la littérature. Il s'agit des algorithmes CONSTRUCT-STRIKE (CS), CONSTRUCT-STRIKE_MODIFIE (CSM), PATH-SCANNING (PS) et AUGMENT-MERGE (AM) décrits à la section 1.3.1. Nous avons également pris en compte la version modifiée du PATH-SCANNING (PSM) proposée par Pearn [Pea89] ainsi que la méthode tabou (BEL) proposée par Belenguer et al. [Bel97a].

Problèmes de DeArmon:

Les résultats obtenus par toutes ces méthodes sur les 23 problèmes de DeArmon sont présentés dans le tableau 2.4. Le numéro de chaque problème figure dans la colonne Pb. Ce tableau contient également pour chaque problème, la valeur F_0 de la solution initiale de CARPET et DVV2, la valeur F_{best} de la meilleure solution connue à ce jour, ainsi que la valeur de la borne inférieure $\underline{F}$. Les valeurs des solutions qui égalent F_{best} sont en gras, celles qui égalent $\underline{F}$ sont soulignées. Les valeurs E_{moyen} et E_{max} sont calculées par rapport à F_{best}. Les temps d'exécution en secondes sont donnés pour CARPET et DVV2. Nous ne disposons malheureusement pas des temps de calcul des autres méthodes.

Les valeurs de F_{best} figurant dans le tableau 2.4 sont toutes atteintes par une des méthodes présentées, sauf celles des problèmes 11 et 15. Les meilleures solutions connues à ce jour pour ces deux instances ont été obtenues par des variantes de DVV au cours des tests numériques que nous avons effectués.

Les résultats du tableau 2.4 permettent de répartir les algorithmes en trois groupes.
Le premier groupe est composé des algorithmes constructifs les moins élaborés à savoir: CONSTRUCT-STRIKE, PATH-SCANNING, AUGMENT-MERGE et l'algorithme fournissant la solution initiale de CARPET et DVV2. Ces méthodes se caractérisent par un écart moyen par rapport à F_{best} de plus de 5%. Elles ne sont de plus pas très robustes, l'écart maximum avec F_{best} est toujours de plus de 20%. Le nombre de solutions optimales atteintes est très faible.
Le deuxième groupe contient les variantes modifiées du CONSTRUCT-STRIKE et du PATH-SCANNING. L'écart moyen est inférieur à 5%, un nombre de solutions optimales plus important est obtenu. Ces algorithmes restent néanmoins aussi peu robustes que ceux du premier groupe.
Le dernier groupe se compose des méthodes de recherche locale. L'écart par rapport à F_{best} est inférieur à 0.5%, 18 solutions optimales sont atteintes sur les 23 problèmes traités. Ces algorithmes sont de plus très robustes puisque l'écart maximum par rapport à F_{best} ne dépasse pas 4%. Cet écart est d'ailleurs inférieur à l'écart moyen des algorithmes des deux premiers groupes. Pour le problème 25 cependant, aucune des méthodes de recherche locale ne parvient à battre le CONSTRUCT-STRIKE_MODIFIE qui est le seul à trouver la solution optimale de ce problème.

En comparant les méthodes de recherche locale entre elles, on constate que CARPET

et DVV2 trouvent des solutions qui en moyenne sont de qualité très légèrement supérieure à celles produites par le tabou de Belenguer et al. CARPET et DVV2 sont quasiment équivalents sur les instances de DeArmon, que l'on considère la qualité des solutions fournies ou les temps d'exécution.

Pb	F	F_{best}	PS	AM	CS	MCS	MPS	BEL	F_0	CARPET		DVV2	
										coût	temps	coût	temps
1	316	316	**316**	326	331	323	**316**	**316**	342	**316**	0.16	**316**	0.04
2	339	339	367	367	418	345	355	**339**	369	**339**	2.72	**339**	0.20
3	275	275	289	316	313	**275**	283	**275**	289	**275**	0.16	**275**	0.00
4	287	287	320	290	350	**287**	292	**287**	321	**287**	0.16	**287**	0.20
5	377	377	417	383	475	386	401	**377**	409	**377**	1.60	**377**	2.18
6	298	298	316	324	356	315	319	**298**	329	**298**	0.64	**298**	0.24
7	325	325	357	**325**	355	**325**	**325**	**325**	353	**325**	0.00	**325**	0.04
10	344	348	416	356	407	366	380	358	390	**348**	29.92	350	52.40
11	303	309	355	339	364	346	357	319	391	317	52.64	315	86.16
12	275	275	302	302	364	**275**	281	**275**	312	**275**	0.16	**275**	0.00
13	395	395	424	443	501	406	424	**395**	410	**395**	0.32	**395**	0.04
14	448	458	560	573	655	645	566	**458**	616	**458**	8.96	**458**	16.04
15	536	536	592	560	560	544	551	544	606	544	18.56	544	10.84
16	100	100	102	102	112	102	**100**	**100**	104	**100**	0.16	**100**	0.04
17	58	58	**58**	**58**	**58**	**58**	**58**	**58**	**58**	**58**	0.00	**58**	0.04
18	127	127	131	131	149	**127**	131	**127**	143	**127**	1.44	**127**	1.56
19	91	91	93	**91**	**91**	**91**	93	**91**	93	**91**	0.00	**91**	0.04
20	164	164	168	170	174	**164**	167	**164**	166	**164**	0.00	**164**	0.04
21	55	55	57	63	63	63	**55**	**55**	65	**55**	0.16	**55**	0.08
22	121	121	125	123	125	123	123	**121**	133	**121**	4.80	**121**	1.20
23	156	156	168	158	165	**156**	163	**156**	170	**156**	2.24	**156**	1.40
24	200	200	207	204	204	**200**	202	**200**	208	**200**	7.52	**200**	7.76
25	233	233	241	237	237	**233**	244	235	245	235	24.64	235	28.12
E_{moyen}			7.26	5.71	14.03	4.02	4.45	0.37	10.05	0.17		0.17	
E_{max}			22.27	25.11	43.01	40.83	23.58	3.24	34.50	2.59		1.94	
N_{opt}			2	3	2	11	5	18	1	18		18	
N_{best}			2	3	2	11	5	19	1	20		19	
Temps moyen											6.82		5.48

TAB. 2.4 -

Comparaison: problèmes de DeArmon.

Problèmes de Benavent:

Nous avons comparé les performances de CARPET et de DVV2 avec la méthode tabou (BEL) proposée par Belenguer et al. dans [Bel97a]. D'autres valeurs, meilleures que celles figurant dans [Bel97a] sont présentées dans [Bel97b]. Nous n'avons malheureusement pas beaucoup de précisions quant à la manière dont elles ont été obtenues. Il est seulement indiqué, dans [Bel97b], que la plupart de ces valeurs ont été produites à l'aide de la méthode tabou décrite dans [Bel97a]. Les valeurs présentées dans [Bel97b] seront prises en compte pour déterminer la valeur des meilleures solutions connues à ce jour. Pour un problème donné, F_{best} est le minimum entre la valeur mentionnée dans [Bel97b] et la meilleure solution obtenue pour ce problème par les différentes variantes de CARPET et de DVV que nous avons testées. Les valeurs de F_{best} dues à Belenguer et al. sont suivies de la lettre b, celles obtenues par une variante de CARPET sont suivies de la lettre c et celles fournies par une version de DVV sont suivies de la lettre d. Les écarts sont mesurés par rapport à F_{best}.

Pb	E	F_{best}	BEL	CARPET		DVV2	
				coût	temps	coût	temps
1.A	173	173 b,c,d	**173**	**173**	0.50	**173**	0.08
1.B	173	173 b,c,d	**173**	179	6.79	178	4.36
1.C	235	245 b,c,d	262	250	16.68	248	16.32
2.A	227	227 b,c,d	**227**	**227**	0.01	**227**	0.00
2.B	259	259 b,c,d	**259**	**259**	1.15	**259**	1.96
2.C	455	457 c,d	465	465	43.19	**457**	34.40
3.A	81	81 b,c,d	**81**	**81**	0.61	**81**	0.00
3.B	87	87 b,c,d	**87**	**87**	0.21	**87**	0.00
3.C	138	138 b,c,d	143	**138**	42.14	140	26.92
4.A	400	400 b,c,d	400	**400**	2.31	**400**	1.36
4.B	412	412 b,c,d	**412**	414	28.09	414	20.32
4.C	428	428 d	430	450	41.27	**428**	39.08
4.D	520	540 d	551	550	53.54	544	65.56
5.A	423	423 b,d	**423**	428	21.30	**423**	1.20
5.B	446	446 b,c,d	**446**	**446**	4.32	449	14.08
5.C	469	474 b,c,d	**474**	476	24.75	**474**	24.20
5.D	571	593 c	603	608	90.88	599	42.48
6.A	223	223 b,c,d	**223**	**223**	0.72	**223**	0.20
6.B	231	233 b,d	**233**	241	16.42	**233**	11.52
6.C	311	317 c	321	323	72.84	325	42.00
7.A	279	279 b,c,d	**279**	**279**	0.33	**279**	0.24
7.B	283	283 b,c,d	**283**	**283**	0.38	**283**	0.20
7.C	333	334 b,c,d	**334**	**334**	66.56	335	44.84
8.A	386	386 b,c,d	**386**	**386**	7.16	**386**	1.68
8.B	395	395 b,c	**395**	**395**	9.39	403	10.60
8.C	517	528 c	540	546	82.63	543	56.28
9.A	323	323 b,c,d	**323**	**323**	16.68	**323**	13.68
9.B	326	326 b,c,d	**326**	328	33.63	**326**	26.36
9.C	332	332 b,c	**332**	**332**	43.96	336	28.36
9.D	382	397 d	401	402	196.88	399	62.40
10.A	428	428 b,c,d	**428**	**428**	4.31	**428**	1.76
10.B	436	436 b,c,d	**436**	**436**	23.06	**436**	7.64
10.C	446	446 b,d	**446**	452	44.22	**446**	33.12
10.D	524	536 c	542	541	160.94	538	76.68
E_{moyen}			0.65	0.93		0.54	
E_{max}			6.94	5.14		2.89	
N_{opt}			20	17		17	
N_{best}			24	18		20	
Temps moyen					34.05		20.89

TAB. 2.5 –

Comparaison: problèmes de Benavent.

Les trois méthodes sont pratiquement équivalentes au niveau des écarts moyens par rapport à F_{best}. Elles se situent toutes en moyenne à moins de 1% des meilleures solutions connues. En observant les valeurs de E_{max} et de E_{moyen}, c'est DVV2 qui a le meilleur comportement. Toutefois, les écarts par rapport aux deux autres méthodes restent faibles. La méthode tabou de Belenguer et al. parvient à atteindre un nombre plus important de solutions optimales que les autres algorithmes. En comparant CARPET à DVV2, on constate que cette dernière méthode est légèrement plus performante. Elle est un peu plus rapide, et la qualité des solutions obtenues est en moyenne meilleure.

En observant les valeurs de F_{best}, on remarque que celles-ci peuvent toujours être obtenues soit par une variante de CARPET soit par une variante de DVV. Les valeurs de

F_{best} des problèmes 4.C, 4.D et 9.D sont dues uniquement à une version de DVV. Celles des problèmes 5.D, 6.C et 10.D sont obtenues seulement à l'aide d'une variante de CARPET.

Problèmes générés aléatoirement

Nous avons effectué une dernière série de tests sur des instances générées aléatoirement. Un des buts de ces tests est d'examiner le comportement de CARPET et DVV2 lorsque toutes les arêtes d'un graphe ne sont pas des arêtes à desservir. Un autre objectif est d'évaluer les performances de ces deux algorithmes sur des problèmes de taille importante (plus de 1000 clients).
Les valeurs E_{moyen} et E_{max} représentent l'écart moyen et l'écart maximum (sur les 10 instances de chacun des 27 types de problèmes) par rapport à la borne inférieure $\underline{F}$. Le nombre de sommets des graphes traités est noté $|V|$, leur densité ϵ et la proportion de clients par rapport au nombre d'arêtes ω.

$\|V\|$	ϵ	ω	CARPET				DVV2			
			E_{moyen}	E_{max}	N_{opt}	Temps	E_{moyen}	E_{max}	N_{opt}	Temps
20	[0.1,0.3]	[0.1,0.3]	0.87	7.06	8	1.26	0.87	7.06	8	0.97
		[0.4,0.6]	1.95	6.17	5	9.36	1.94	5.56	5	8.08
		[0.8,1.0]	0.68	2.51	5	8.24	0.51	3.58	7	4.14
	[0.4,0.6]	[0.1,0.3]	1.80	5.08	3	11.52	1.92	6.78	4	4.20
		[0.4,0.6]	0.51	2.62	7	14.11	0.34	3.39	9	7.16
		[0.8,1.0]	0.30	2.12	8	10.52	0.21	2.12	9	5.41
	[0.7,0.9]	[0.1,0.3]	1.37	3.15	4	4.89	1.28	3.15	5	2.47
		[0.4,0.6]	0.94	3.86	5	36.15	0.67	5.08	6	23.00
		[0.8,1.0]	0.41	2.07	4	51.25	0.20	0.89	6	25.42
40	[0.1,0.3]	[0.1,0.3]	1.43	5.68	5	13.69	1.41	5.14	5	5.71
		[0.4,0.6]	1.37	4.84	5	83.58	0.85	2.96	4	39.46
		[0.8,1.0]	0.05	0.52	9	11.46	0	0	10	0.38
	[0.4,0.6]	[0.1,0.3]	1.41	8.89	5	43.86	1.26	9.16	6	17.71
		[0.4,0.6]	0.64	2.93	4	200.26	0.19	0.82	6	42.12
		[0.8,1.0]	0.27	1.35	7	462.18	0.09	0.40	7	60.97
	[0.7,0.9]	[0.1,0.3]	0.94	4.43	5	57.64	0.29	0.97	5	20.54
		[0.4,0.6]	0.13	1.27	9	91.40	0.08	0.78	9	19.30
		[0.8,1.0]	0.04	0.40	9	315.05	0.04	0.36	9	33.30
60	[0.1,0.3]	[0.1,0.3]	0.55	3.55	6	19.42	0.28	2.26	7	5.03
		[0.4,0.6]	1.75	7.32	1	209.50	1.45	8.56	3	44.06
		[0.8,1.0]	0.72	2.77	3	349.16	0.26	1.38	5	65.43
	[0.4,0.6]	[0.1,0.3]	0.69	3.26	6	179.10	0.28	1.34	6	33.94
		[0.4,0.6]	0.10	0.85	7	507.79	0.01	0.13	9	60.80
		[0.8,1.0]	0.01	0.10	9	904.95	0.002	0.02	9	154.33
	[0.7,0.9]	[0.1,0.3]	0.13	0.49	4	169.94	0.02	0.20	9	16.56
		[0.4,0.6]	0.18	0.80	7	1226.04	0.04	0.26	7	175.71
		[0.8,1.0]	0.004	0.03	8	4452.57	0	0	10	271.24
Moyenne sur les 270 instances			0.71			349.81	0.54			42.50
Total ou maximum				8.89	158			9.16	185	

TAB. 2.6 -

Résultats de CARPET et DVV2 sur des instances générées aléatoirement.

Le tableau 2.6 montre que DVV2 est plus performant que CARPET sur les instances que nous avons générées aléatoirement. La qualité moyenne des solutions obtenues par DVV2 est meilleure que celle fournie par les solutions de CARPET. Les deux méthodes produisent toutefois des solutions de très bonne qualité puisqu'en moyenne l'écart par rapport à $\underline{F}$

ne dépasse pas 1%. DVV2 parvient à atteindre un nombre plus important de solutions optimales.
La différence la plus importante apparaît lorsqu'on observe les temps de calcul. DVV2 est en moyenne 8 fois plus rapide que CARPET. Le temps d'exécution le plus important enregistré pour CARPET sur les 270 problèmes traités est de 20524 secondes (problème où $|R| = 1416$ et $\underline{m} = 9$) alors qu'avec DVV2 le temps de calcul le plus élevé n'est que de 1024 secondes (problème où $|R| = 1250$ et $\underline{m} = 12$). La méthode de descente à voisinage variable semble donc particulièrement efficace lorsqu'il s'agit de traiter des instances de grande taille.

Pour un nombre de sommets fixés et une densité ϵ donnée, les écarts moyens les plus faibles apparaissent lorsque la proportion de clients est la plus élevée. C'est dans ces cas-là également que le nombre de solutions optimales atteintes est généralement le plus important. Il est difficile d'expliquer un tel comportement. Peut-être est-ce la borne inférieure qui est plus performante sur ce genre d'instances.

Pour conclure cette comparaison entre CARPET et DVV2, notons que cette dernière méthode est beaucoup plus simple dans sa conception que la méthode tabou que nous avons développée. Elle n'utilise en effet que deux paramètres qui sont l'ordre dans lequel les voisinages sont considérés et la portion du voisinage examinée à chaque itération d'une méthode de descente. Très peu de tests ont été nécessaires au réglage de ces paramètres contrairement à CARPET où le réglage de l'ensemble des paramètres a nécessité de très nombreux tests numériques.

2.4 PTAC orienté

Cette section sera tout d'abord consacrée à la description d'une nouvelle borne inférieure que nous avons développée pour le PTAC orienté. Nous présenterons dans un deuxième temps une adaptation au cas orienté des heuristiques CARPET et DVV2. Une comparaison entre la valeur des solutions fournies par les heuristiques et celle de la borne inférieure sera finalement effectuée.

2.4.1 Une nouvelle borne inférieure pour le PTAC orienté

Le problème de tournées sur les arcs avec capacité est un problème NP-dur [Gol81]. Pour traiter des instances de grande taille, les seules méthodes dont nous disposons aujourd'hui sont des méthodes heuristiques. Afin d'évaluer leur qualité, il est utile de posséder des bornes inférieures. Dans le cas non orienté, de très nombreuses bornes existent, citons par exemple celles développées par Golden et Wong [Gol81], par Assad, Pearn et Golden [Ass87], par Win [Win88], par Pearn [Pea88], et plus récemment par Benavent, Campos et Mota [Ben92] et par Belenguer et Benavent [Bel97b]. Les bornes les plus simples pour le cas non orienté peuvent facilement être adaptées au cas orienté. Pour des bornes plus complexes, une telle adaptation nécessite plus de travail. Nous allons nous baser sur la borne $LB1$ décrite pour le cas non orienté dans [Ben92] afin de développer une borne inférieure pour le PTAC orienté.

Après avoir introduit quelques notations, la borne $LB1$ de Benavent et al. sera brièvement décrite. Nous adapterons ensuite cette borne au cas orienté et démontrerons sa validité.

2.4.1.1 Notations

Le PTAC non orienté est défini sur un graphe $G = (V, E)$ où $V = \{v_0, v_1, ..., v_n\}$ est l'ensemble des sommets et E l'ensemble des arêtes. A chaque arête (v_i, v_j) de E est associé un coût $c_{ij} \geq 0$ ainsi qu'une demande $q_{ij} \geq 0$. Nous appellerons R l'ensemble des arêtes (v_i, v_j) de E ayant une demande non nulle. W représentera la capacité des véhicules et le dépôt sera désigné par le sommet v_0 de V. La demande totale des arêtes de R sera notée Q_T et $C(R)$ représentera le coût total de ces arêtes. Nous noterons par $\underline{m} = \lceil \frac{Q_T}{W} \rceil$ la borne inférieure sur le nombre de véhicules minimum nécessaires pour satisfaire Q_T.

A toute solution admissible s du PTAC dans $G = (V, E)$, nous pouvons associer un graphe $G_s = (V, R \cup E_s)$ que nous appellerons graphe augmenté. E_s contient autant de copies de l'arête (v_i, v_j) de E que le nombre de fois où (v_i, v_j) est traversée dans s sans être desservie. Nous appellerons arêtes artificielles les arêtes de E_s. Leur coût total sera noté $C(E_s)$. Les arêtes de s lorsqu'elles sont traversées sans être desservies seront également appelées arêtes artificielles.

Le PTAC orienté est défini sur un graphe $G = (V, A)$ où A est l'ensemble des arcs. Nous utiliserons les mêmes notations que celles présentées dans le cas non orienté pour désigner: le coût des arcs, le dépôt, les demandes, la capacité des véhicules, la demande totale à couvrir, l'ensemble des arcs à desservir, la longueur totale de ces arcs ainsi que la borne inférieure sur le nombre de véhicules minimum nécessaires.

Dans le cas orienté, le graphe augmenté associé à une solution admissible s sera simplement désigné par $G_s = (V, R \cup A_s)$ au lieu de $G_s = (V, R \cup E_s)$ et nous parlerons d'arcs à la place d'arêtes.

Nous appellerons $G_R = (V_R, R)$ le sous-graphe partiel de $G = (V, E)$ (ou de $G = (V, A)$) induit par les arêtes (ou les arcs) de R. V_R contient l'ensemble des sommets incidents à une arête (ou à un arc) de R ainsi que le dépôt. Nous supposerons que les sommets se trouvant dans V_R sont ceux ayant les indices compris entre 0 et $|V_R| - 1$. Nous avons alors $V_R = \{v_0, v_1, ..., v_{|V_R|-1}\}$.
Nous noterons par f_i la différence $d^-_{G_R}(i) - d^+_{G_R}(i)$, $(i = 0, ..., n)$. Si v_i est symétrique dans G_R ou si $v_i \notin V_R$ alors $f_i = 0$.

Le plus court chemin reliant le sommet v_i au sommet v_j dans le réseau initial G sera noté SP_{ij}. La longueur de ce chemin sera notée sp_{ij}. La plus courte chaîne reliant v_i à v_j dans G sera notée SC_{ij} et sc_{ij} désignera sa longueur.

Etant donné un graphe $G = (V, E)$ non orienté et pair, nous noterons par $M_{min}(G)$ le couplage parfait de coût minimum dans G. Le coût de ce couplage sera noté $C(M_{min}(G))$.

2.4.1.2 Borne $LB1$ dans le cas non orienté

Nous décrivons dans cette section la borne inférieure $LB1$ développée par Benavent et al. [Ben92] pour le cas non orienté du PTAC. Pour cela, nous allons commencer par décrire deux types de chaînes composées uniquement d'arêtes artificielles. Ces chaînes peuvent se rencontrer soit dans les tournées d'une solution admissible s, soit dans le graphe augmenté G_s associé à s.

2.4.1.2.1 Chaînes artificielles de type I

Le degré du dépôt dans G_R est $d_{G_R}(0)$. Comme $\underline{m}$ véhicules au minimum sont nécessaires pour satisfaire Q_T, il existe au moins, dans toute solution admissible, $2\underline{m}$ arêtes incidentes au dépôt qui sont traversées (en étant desservies ou pas).
Posons $\lambda = \max\{0, 2\underline{m} - d_{G_R}(0)\}$. Comme chaque arête de R n'est desservie qu'une seule fois, dans toute solution admissible s, au moins λ arêtes incidentes au dépôt sont traversées sans être desservies. De façon un peu plus précise, il y a dans s au moins λ chaînes composées d'arêtes artificielles uniquement et qui ont pour extrémités le dépôt et un sommet v_i $(\neq v_0)$ incident à une arête desservie. De telles chaînes seront appelées chaînes artificielles de type I. Elles correspondent en fait soit à des chaînes artificielles reliant le dépôt au premier client desservi d'une tournée, soit à des chaînes artificielles reliant le dernier client desservi d'une tournée au dépôt. Comme le graphe augmenté G_s contient l'ensemble des arêtes artificielles d'une solution s, il contient donc les chaînes artificielles de type I. Un sommet v_i de G_R, différent du dépôt, est l'extrémité d'au plus $d_{G_R}(i)$ chaînes artificielles de type I car il est incident à $d_{G_R}(i)$ arêtes de R.

Exemple:

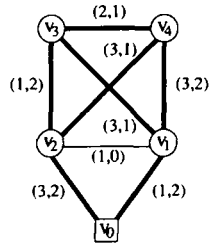


FIG. 2.9 -

Graphe initial G.

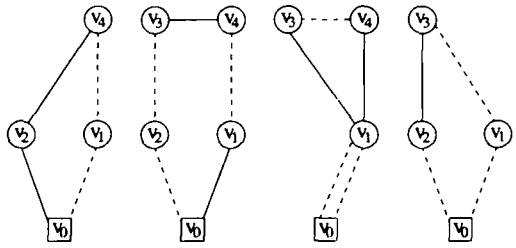


FIG. 2.10 -

Une solution admissible s.

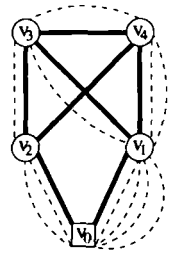


FIG. 2.11 -

Graphe augmenté G_s.

Dans l'exemple ci-dessus, le graphe initial G est représenté dans la figure 2.9. Les arêtes en gras correspondent aux arêtes de R. Dans les parenthèses, le premier nombre indique la longueur de l'arête et le second sa demande. La demande totale Q_T vaut 11 et la capacité des véhicules $W = 3$. La valeur de $\underline{m}$ est alors de 4. Il faut donc au minimum 4 véhicules pour satisfaire la demande totale sans provoquer de dépassement de capacité.
La figure 2.10 représente une solution admissible du PTAC dans G. Les arêtes en traitillés sont les arêtes qui sont traversées sans être desservies. La valeur de λ est de $2\underline{m} - d_{G_R}(0) = 8 - 2 = 6$. Il y a donc au moins 6 chaînes artificielles de type I dans s. Ces chaînes sont les suivantes:

$$(v_0, v_1, v_4),\ (v_3, v_2, v_0),\ (v_0, v_1),\ (v_1, v_0),\ (v_0, v_1, v_3) \text{ et } (v_2, v_0).$$

Le graphe augmenté G_s correspondant à la solution de la figure 2.10 est décrit dans la figure 2.11. Les chaînes artificielles de type I que nous venons d'énumérer figurent également dans G_s.

Renumérotons les sommets de $V_R \setminus v_0 = \{v_1, ..., v_{|V_R|-1}\}$ selon la longueur non décroissante de la plus courte chaîne les reliant au dépôt dans G (le dépôt est toujours v_0). Compte tenu de cette nouvelle numérotation, nous avons donc $sc_{01} \leq sc_{02} \leq ... \leq sc_{0|V_R|-1}$. Soit i^* le plus petit entier tel que $d_{G_R}(1) + ... + d_{G_R}(i^*) \geq \lambda$. Nous allons définir pour chaque sommet $v_1, v_2, ..., v_{i^*}$ de V_R une valeur $d(i)$ $(i = 1, 2, ..., i^*)$ de la manière suivante: $d(i) = d_{G_R}(i)$ si $1 \leq i < i^*$ et $d(i^*) = \lambda - (d_{G_R}(1) + ... + d_{G_R}(i^* - 1))$.
Les sommets $v_1, ..., v_{i^*}$ correspondent aux extrémités des λ chaînes artificielles de type I les plus courtes, et la valeur $D = d(1)sc_{01} + d(2)sc_{02} + ... + d(i^*)sc_{0i^*}$ est une borne inférieure sur la longueur totale des chaînes artificielles de type I de toute solution admissible. Assad, Pearn et Golden [Ass87] ont proposé une borne inférieure pour le PTAC non orienté basée sur ces λ chaînes artificielles de type I. Cette borne est la suivante:

$$LB_{Assad} = C(R) + D.$$

Dans l'exemple ci-dessus, la numérotation initiale des sommets est telle que: $sc_{01} \leq sc_{02} \leq sc_{03} \leq sc_{04}$. La valeur de i^* est donc 2 et celle de D est égale à $3 \times 1 + 3 \times 2 = 9$. La borne LB_{Assad} vaut alors $C(R) + D = 16 + 9 = 25$.

La borne LB_{Assad} se base uniquement sur le nombre de chaînes artificielles de type I minimum que doit comporter toute solution admissible. Elle ne prend pas en compte le fait que tout graphe augmenté G_s d'une solution admissible s est pair.

2.4.1.2.2 Chaînes artificielles de type II

Le fait que G_s soit pair implique qu'il existe des chaînes dans G_s composées d'arêtes artificielles uniquement et qui relient entre eux deux sommets de degré impair dans G_R. De telles chaînes seront appelées chaînes artificielles de type II. Une borne inférieure sur la longueur totale des chaînes artificielles de type II peut être obtenue en résolvant un problème de couplage parfait de coût minimum dans un graphe complet G_{aux}. Les sommets de G_{aux} sont les sommets de degré impair dans G_R. La longueur d'une arête reliant deux sommets est égale à celle de la plus courte chaîne reliant ces sommets dans le graphe de départ G. En additionnant $C(R)$ et le coût de ce couplage, nous obtenons également une borne inférieure pour le PTAC non orienté.

Reprenons l'exemple de la page précédente. Le graphe G_R correspondant au graphe de la figure 2.9 possède 4 sommets de degré impair: v_1, v_2, v_3, v_4. Il est représenté à la figure 2.12. Le graphe G_{aux} est représenté à la figure 2.13. Le nombre au-dessus d'une arête représente son coût. Les arêtes du couplage parfait de coût minimum dans G_{aux} sont dessinées en gras. Le coût du couplage vaut 3. En additionnant 3 à la valeur de $C(R)$, nous obtenons une borne inférieure dont la valeur est 19.

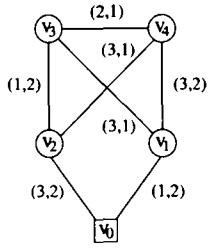


FIG. 2.12 -

Graphe G_R.

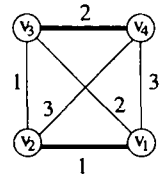


FIG. 2.13 -

Graphe G_{aux}.

2.4.1.2.3 Calcul de $LB1$

Le principe sur lequel repose la borne $LB1$ de Benavent et al. consiste à prendre en compte les deux types de chaînes artificielles que nous venons de décrire. Ces deux types ne sont pas forcément distincts. Il existe des chaînes artificielles qui sont à la fois de type I et de type II. Il est possible également qu'une chaîne artificielle de type II contienne une chaîne artificielle de type I et inversement qu'une chaîne artificielle de type I contienne une chaîne artificielle de type II. Il faut donc prendre en compte les éventuelles interactions entre ces deux types de chaînes. Pour cela, Benavent et al. commencent tout d'abord par construire un graphe auxiliaire complet $G_a = (V_a, E_a)$ où $V_a = K \cup L \cup V_{imp}$. Les ensembles K, L et V_{imp} sont définis de la manière suivante:

$K = \{k_1, ..., k_\lambda\}$ contient λ copies du dépôt.
$L = \{l_1, ..., l_r\}$ contient $d_{G_R}(i)$ copies des sommets v_i de G_R $(i = 1, ..., i^*)$.
$V_{imp} = \{v_{imp_1}, ..., v_{imp_t}\}$ contient une copie de chaque sommet de degré impair dans G_R n'ayant pas déjà de copie ni dans L ni dans K.

Remarque:

Si $\lambda = 0$, les ensembles K et L sont vides. Nous avons alors $V_a = V_{imp}$. Seules les chaînes artificielles de type II doivent être alors prises en compte puisque le nombre minimum de chaînes artificielles de type I est 0. Nous retombons sur la borne inférieure décrite en 2.4.1.2.2.
Si tous les sommets de G_R sont pairs, V_{imp} est vide. Dans ce cas, seules les chaînes artificielles de type I sont prises en compte car il n'existe aucune chaîne artificielle de type II. Nous retombons alors sur la borne inférieure LB_{Assad}.

Les coûts des arêtes de G_a sont définis de la façon suivante:

Le coût de chaque arête reliant deux sommets de K est infini.

Le coût d'une arête (l_i, l_j) reliant deux sommets de L est égal à la longueur de la plus courte chaîne dans G reliant les sommets dont l_i et l_j sont les copies.

Le coût d'une arête (v_{imp_i}, v_{imp_j}) reliant deux sommets de V_{imp} est égal à la longueur de la plus courte chaîne dans G reliant les sommets dont v_{imp_i} et v_{imp_j} sont les copies.

Le coût d'une arête (l_i, k_j) reliant un sommet de L à un sommet de K est égal à la longueur de la plus courte chaîne dans G reliant le dépôt au sommet dont l_i est la copie.

Le coût d'une arête (v_{imp_i}, k_j) reliant un sommet de V_{imp} à un sommet de K est égal à la longueur de la plus courte chaîne dans G reliant le dépôt au sommet dont v_{imp_i} est la copie.

Le coût d'une arête (v_{imp_i}, l_j) reliant un sommet de V_{imp} à un sommet de L est égal à la longueur de la plus courte chaîne dans G reliant les sommets dont v_{imp_i} et l_j sont les copies.

Une fois le graphe auxiliaire construit, un couplage parfait de coût minimum $M_{min}(G_a)$ est recherché dans G_a. La borne $LB1$ est obtenue en sommant $C(R)$ et la valeur $C(M_{min}(G_a))$ du couplage optimum. Nous avons donc:

$$LB1 = C(R) + C(M_{min}(G_a)).$$

Le couplage parfait de coût minimum dans G_a nous assure que toutes les λ copies du dépôt se trouvant dans K sont reliées à un sommet de L ou de V_{imp}. Les λ arêtes de $M_{min}(G_a)$ ayant une copie du dépôt comme extrémité correspondent à des chaînes artificielles de type I. Les arêtes de $M_{min}(G_a)$ reliant deux sommets de V_{imp} correspondent à des chaînes artificielles de type II. Les interactions entre les deux types de chaînes sont prises en compte lorsqu'un sommet de K est couplé avec un sommet de V_{imp} ou lorsqu'un sommet de K est couplé avec un sommet de L, copie d'un sommet de degré impair dans G_R.

Nous ne présentons pas ici la preuve de la validité de la borne $LB1$. Le lecteur désireux de connaître cette démonstration pourra se référer à [Ben92]. Nous nous contentons d'illustrer la construction du graphe G_a et le calcul de $LB1$ à l'aide de l'exemple qui va suivre.

Exemple:

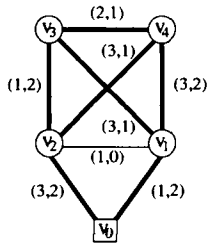


FIG. 2.14 -

Graphe initial G.

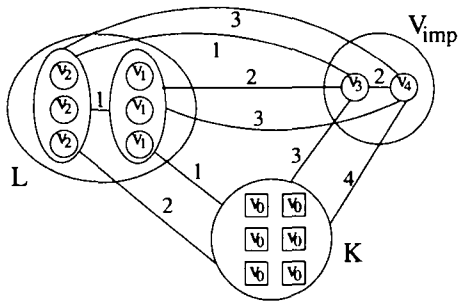


FIG. 2.15 -

Graphe auxiliaire G_a.

Dans l'exemple ci-dessus, la figure 2.14 représente le graphe de départ G. Il s'agit du même graphe que celui de la figure 2.9. La figure 2.15 correspond au graphe auxiliaire G_a obtenu à partir de G.
La numérotation initiale des sommets de G est telle que $sc_{01} \leq sc_{02} \leq sc_{03} \leq sc_{04}$. La valeur de λ est 6. L'ensemble K est donc composé de 6 copies du dépôt.
Comme $d_{G_R}(1) + d_{G_R}(2) = 3 + 3 = \lambda$, l'ensemble de sommets L contient 3 copies du sommet v_1 et 3 copies du sommet v_2.
Il reste 2 sommets de degré impair dans G_R différents du dépôt et ne figurant pas déjà dans L. Il s'agit des sommets v_3 et v_4. Ces sommets se trouvent dans V_{imp}.
Pour des raisons de clarté, toutes les arêtes de G_a ne sont pas dessinées. Nous avons uniquement représenté les arêtes reliant un groupe de copies d'un même sommet à un autre. Le coût des arêtes entre copies d'un même sommet différent du dépôt est nul. Le coût des arêtes entre copies du dépôt est infini.
Dans le couplage optimum $M_{min}(G_a)$, tous les sommets de K sont couplés avec les sommets de L et une arête relie les sommets v_3 et v_4 de V_{imp}. Le coût, $C(M_{min}(G_a))$, de ce couplage vaut alors 11. Comme $C(R) = 16$, la borne inférieure $LB1$ a une valeur de 27.

2.4.1.3 Adaptation de la borne $LB1$ au cas orienté

Dans cette section, nous allons tout d'abord adapter la notion de chaînes artificielles de type I et de type II au cas orienté. Nous décrirons ensuite la façon de construire un graphe auxiliaire dans lequel, tout comme dans le cas non orienté, nous rechercherons un couplage parfait de coût minimum. Le coût de ce couplage ajouté à $C(R)$ nous donnera une borne inférieure, notée $\vec{LB}1$, pour le PTAC orienté. La validité de cette borne sera finalement démontrée.

2.4.1.3.1 Chemins artificiels de type I

La façon d'estimer le nombre minimum de véhicules nécessaires pour satisfaire la demande totale Q_T ne change pas dans le cas orienté. Nous avons toujours $\underline{m} = \lceil \frac{Q_T}{W} \rceil$. Cela signifie que dans toute solution admissible du PTAC orienté, il y a au moins $\underline{m}$ arcs qui quittent le dépôt et $\underline{m}$ arcs qui y arrivent.

Posons $\lambda_+ = \max\{0, \underline{m} - d^+_{G_R}(0)\}$ et $\lambda_- = \max\{0, \underline{m} - d^-_{G_R}(0)\}$. Les arcs de R étant desservis exactement une fois, il y a, dans toute solution admissible s, au moins λ_+ arcs

quittant le dépôt et au moins λ_- arcs aboutissant au dépôt qui sont traversés sans être desservis. Il est possible d'améliorer l'estimation du nombre minimum d'arcs artificiels entrant et sortant du dépôt en prenant en compte le fait que celui-ci est symétrique dans le graphe augmenté de toute solution admissible. Soit $z^+ = \max\{\lambda_+, f_0\}$ et soit $z^- = \max\{\lambda_-, -f_0\}$. La valeur de z^+ (resp. z^-) correspond au nombre minimum d'arcs artificiels qui quittent le dépôt (resp. qui entrent au dépôt) dans toute solution admissible s. La solution s possède donc au moins z^+ chemins, composés uniquement d'arcs artificiels, partant du dépôt et se terminant en un sommet v_i ($v_i \neq v_0$) incident à un arc desservi (v_i, v_k). Elle contient également au moins z^- chemins, composés uniquement d'arcs artificiels, partant d'un sommet v_i ($v_i \neq v_0$) incident à un arc desservi (v_k, v_i) et aboutissant au dépôt. De tels chemins seront appelés chemins artificiels de type I. Ils se retrouvent également dans le graphe augmenté G'_s. Un sommet v_i de G_R est l'extrémité d'au plus $d^+_{G_R}(i)$ chemins artificiels de type I quittant le dépôt. Il est l'origine d'au plus $d^-_{G_R}(i)$ chemins artificiels de type I aboutissant au dépôt.

Exemple:

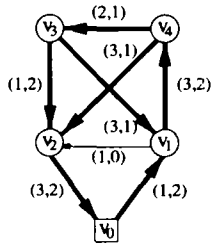


FIG. 2.16 – *Graphe initial G.*

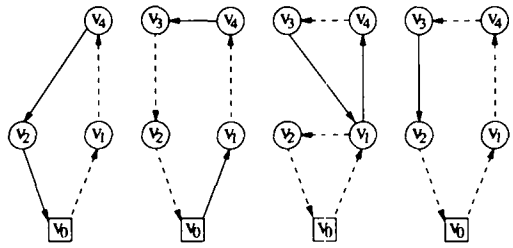


FIG. 2.17 – *Une solution admissible s.*

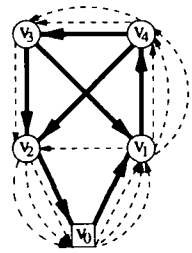


FIG. 2.18 – *Graphe augmenté G_s.*

Dans l'exemple ci-dessus, le graphe initial G est représenté dans la figure 2.16. La demande totale Q_T vaut 11 et la capacité des véhicules 3. La valeur de $\underline{m}$ est alors de 4. La figure 2.17 correspond à une solution admissible s du PTAC dans G. Les valeurs respectives de λ_+ et de λ_- sont $\underline{m} - d^+_{G_R}(0) = 4 - 1 = 3$ et $\underline{m} - d^-_{G_R}(0) = 4 - 1 = 3$. Comme $f_0 = 0$, z^+ est égal à λ_+ et z^- a la même valeur que λ_-. Il y a donc au moins 6 chemins artificiels de type I dans s. Trois d'entre eux quittent le dépôt: (v_0, v_1, v_4), (v_0, v_1) et (v_0, v_1, v_4, v_3), et les trois autres y arrivent: (v_3, v_2, v_0), (v_1, v_2, v_0) et (v_2, v_0).
Le graphe augmenté associé à s est représenté dans la figure 2.18. Il contient les 6 chemins artificiels de type I que nous venons de mentionner.

Nous allons définir 2 permutations π et σ de l'ensemble des indices des sommets de $V_R \setminus \{v_0\} = \{v_1, ..., v_{|V_R|-1}\}$.

La permutation π est telle que: $sp_{0\pi_1} \leq sp_{0\pi_2} \leq ... \leq sp_{0\pi_{|V_R|-1}}$ où $sp_{0\pi_j}$ est la longueur du plus court chemin $SP_{0\pi_j}$ entre v_0 et v_{π_j} ($j = 1, ..., |V_R| - 1$). Soit i^*_π le plus petit entier tel que $d^+_{G_R}(\pi_1) + d^+_{G_R}(\pi_2) + ... + d^+_{G_R}(\pi_{i^*_\pi}) \geq z^+$. Pour chaque sommet v_{π_j} ($j = 1, ..., i^*_\pi$), nous allons définir une valeur $d_\pi(\pi_j)$ de la manière suivante:

$$d_\pi(\pi_j) = d^+_{G_R}(\pi_j) \text{ si } 1 \leq j < i^*_\pi \text{ et}$$
$$d_\pi(\pi_{i^*_\pi}) = z^+ - (d^+_{G_R}(\pi_1) + d^+_{G_R}(\pi_2) + ... + d^+_{G_R}(\pi_{i^*_\pi - 1})).$$

Les sommets $v_{\pi_1}, ..., v_{\pi_{i_\pi^*}}$ correspondent aux extrémités des z^+ chemins artificiels de type I les plus courts qui partent du dépôt. Posons $D_\pi = d_\pi(\pi_1)sp_{0\pi_1} + ... + d_\pi(\pi_{i_\pi^*})sp_{0\pi_{i_\pi^*}}$. Toute solution admissible possède des chemins artificiels de type I qui sortent du dépôt et dont la longueur totale est supérieure ou égale à D_π.

La permutation σ est telle que: $sp_{\sigma_1 0} \leq sp_{\sigma_2 0} \leq ... \leq sp_{\sigma_{|V_R|-1}0}$ où $sp_{\sigma_j 0}$ est la longueur du plus court chemin $SP_{\sigma_j 0}$ entre v_{σ_j} et v_0 $(j = 1, ..., |V_R| - 1)$. Soit i_σ^* le plus petit entier tel que $d^-_{G_R}(\sigma_1) + d^-_{G_R}(\sigma_2) + ... + d^-_{G_R}(\sigma_{i_\sigma^*}) \geq z^-$. Pour chaque sommet v_{σ_j} $(j = 1, ..., i_\sigma^*)$ nous allons définir une valeur $d_\sigma(\sigma_j)$ de la manière suivante:

$$d_\sigma(\sigma_j) = d^-_{G_R}(\sigma_j) \text{ si } 1 \leq j < i_\sigma^* \text{ et}$$
$$d_\sigma(\sigma_{i_\sigma^*}) = z^- - (d^-_{G_R}(\sigma_1) + d^-_{G_R}(\sigma_2) + ... + d^-_{G_R}(\sigma_{i_\sigma^*-1})).$$

Les sommets $v_{\sigma_1}, ..., v_{\sigma_{i_\sigma^*}}$ correspondent aux origines des z^- chemins artificiels de type I les plus courts qui aboutissent au dépôt. Posons $D_\sigma = d_\sigma(\sigma_1)sp_{\sigma_1 0} + ... + d_\sigma(\sigma_{i_\sigma^*})sp_{\sigma_{i_\sigma^*}0}$. Toute solution admissible possède des chemins artificiels de type I qui arrivent au dépôt et dont la longueur totale est supérieure ou égale à D_σ.

Nous avons pu borner inférieurement la longueur totale des chemins artificiels de type I apparaissant dans toute solution admissible du PTAC orienté. Nous pouvons en déduire une borne inférieure qui est une adaptation au cas orienté de la borne proposée par Assad et al. Cette borne que nous noterons $\vec{LB}_{Assad}$ s'obtient de la manière suivante:

$$\vec{LB}_{Assad} = C(R) + D_\pi + D_\sigma.$$

Dans l'exemple de la page précédente, les permutations π et σ sont les suivantes: $\pi_1 = 1, \pi_2 = 2, \pi_3 = 4, \pi_4 = 3$ et $\sigma_1 = 2, \sigma_2 = 1, \sigma_3 = 3, \sigma_4 = 4$. Comme $z^+ = z^- = 3$, les valeurs de i_π^* et i_σ^* sont respectivement 3 et 2. Nous avons donc $D_\pi = 1 \times 1 + 1 \times 2 + 1 \times 4 = 7$ et $D_\sigma = 2 \times 3 + 1 \times 4 = 10$. La valeur de $\vec{LB}_{Assad}$ est égale à $16 + 7 + 10 = 33$.

La borne $\vec{LB}_{Assad}$ se base uniquement sur le nombre minimum de véhicules nécessaires au service des arcs appartenant à R et sur le fait que le dépôt est symétrique dans le graphe augmenté G_s de toute solution admissible s. La symétrie des sommets de G_s différents du dépôt n'est pas prise en compte.

2.4.1.3.2 Chemins artificiels de type II

Le graphe augmenté G_s de toute solution admissible s est symétrique. Dans G_s, de chaque sommet v_i tel que $f_i > 0$ partent f_i chemins, composés d'arcs artificiels uniquement et aboutissant à un sommet v_j tel que $f_j < 0$. De tels chemins seront appelés chemins artificiels de type II. Ils relient dans G_s deux sommets non symétriques dans G_R.
Une borne inférieure sur la longueur totale des chemins artificiels de type II peut être obtenue en résolvant un problème de transbordement dans G où chaque sommet v_i tel que $f_i > 0$ possède une disponibilité valant f_i et chaque sommet v_j tel que $f_j < 0$ une demande égale à $|f_j|$. En ajoutant à $C(R)$ la valeur de la solution optimale de ce problème de transbordement, nous obtenons une borne inférieure pour le PTAC orienté.

Reprenons l'exemple de la page précédente. Les sommets non symétriques du graphe

G_R associé au graphe G de la figure 2.16 sont les sommets: v_1, v_2, v_3 et v_4. Les valeurs de f_1, f_2, f_3 et f_4 sont les suivantes: $f_1 = f_2 = 1$ et $f_3 = f_4 = -1$. La figure 2.19 représente le problème de transbordement à résoudre dans G. Les nombres au-dessus des arcs correspondent à leur coût. Un carré noir ou gris près d'un sommet représente une unité disponible alors qu'un carré blanc correspond à une unité de demande. Une solution optimale du problème de transbordement dans G est représentée à la figure 2.20. Elle consiste à faire passer une unité disponible de v_1 à v_4 et une unité disponible de v_2 à v_3. La première unité transportée emprunte le chemin (v_1, v_4) de longueur 3. La deuxième passe par le chemin $(v_2, v_0, v_1, v_4, v_3)$ dont la longueur est égale à 9. Nous obtenons alors une borne inférieure dont la valeur est égale à 16+3+9=28.

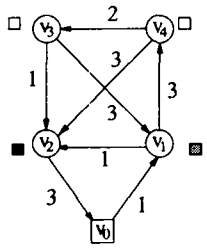


FIG. 2.19 -

Problème de transbordement dans G.

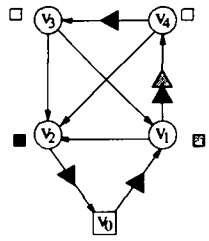


FIG. 2.20 -

Une solution optimale du problème de transbordement.

2.4.1.3.3 Calcul de $L\vec{B}1$

Comme dans le cas non orienté, la borne $L\vec{B}1$ va prendre en compte les deux types de chemins artificiels que nous avons décrits. Cela se fera en résolvant un problème de couplage parfait de poids minimum dans un graphe auxiliaire non orienté $G_a = (V_a, E_a)$. L'ensemble des sommets V_a de ce graphe est la réunion de 6 ensembles:

$$V_a = K_{out} \cup K_{in} \cup V_{out} \cup V_{in} \cup L_{out} \cup L_{in}.$$

Nous allons définir ces 6 ensembles et expliquer à quoi ils correspondent.

K_{out} **et** K_{in}

Les ensembles K_{out} et K_{in} sont définis de la manière suivante:

L'ensemble $K_{out} = \{k^1_{out}, k^2_{out}, ..., k^{z^+}_{out}\}$ contient z^+ copies du dépôt.
L'ensemble $K_{in} = \{k^1_{in}, k^2_{in}, ..., k^{z^-}_{in}\}$ contient z^- copies du dépôt.

Il n'existe aucune arête dans G_a reliant entre eux les sommets de K_{out}. Il en va de même pour les sommets appartenant à K_{in}. Aucune arête non plus ne relie dans G_a un sommet de K_{out} à un sommet de K_{in}. Si $z^+ = 0$ alors $K_{out} = \emptyset$, de même $K_{in} = \emptyset$ lorsque $z^- = 0$.

Lemme 1 $|K_{out}| - |K_{in}| = f_0$

preuve:

Comme $|K_{out}| = \max\{\lambda_+, f_0\}$ et $|K_{in}| = \max\{\lambda_-, -f_0\}$, nous nous trouvons toujours dans un des trois cas suivants:

$$\begin{array}{lllll} 1) & |K_{out}| = \lambda_+ & \text{et} & |K_{in}| = \lambda_- & \\ 2) & |K_{out}| = f_0 & \text{et} & |K_{in}| = \lambda_- & (f_0 > \lambda_+) \\ 3) & |K_{out}| = \lambda_+ & \text{et} & |K_{in}| = -f_0 & (-f_0 > \lambda_-) \end{array}$$

Le cas où $|K_{out}| = f_0$ et $|K_{in}| = -f_0$ est impossible car les valeurs de λ_+ et λ_- sont toujours positives ou nulles.

cas 1:

$$\begin{array}{rcll} |K_{out}| - |K_{in}| & = & \lambda_+ - \lambda_- & = \underline{m} - d^+_{G_R}(0) - (\underline{m} - d^-_{G_R}(0)) \\ & = & d^-_{G_R}(0) - d^+_{G_R}(0) & = f_0 \end{array}$$

cas 2:

$$\begin{array}{rl} f_0 > \lambda_+ & \Longrightarrow \; d^-_{G_R}(0) - d^+_{G_R}(0) > \underline{m} - d^+_{G_R}(0) \\ & \Longrightarrow \; d^-_{G_R}(0) > \underline{m} \\ & \Longrightarrow \; \underline{m} - d^-_{G_R}(0) < 0 \Longrightarrow \lambda_- = 0 \\ & \Longrightarrow \; |K_{out}| - |K_{in}| = f_0 - \lambda_- = f_0 \end{array}$$

cas 3:

$$\begin{array}{rl} -f_0 > \lambda_- & \Longrightarrow \; d^+_{G_R}(0) - d^-_{G_R}(0) > \underline{m} - d^-_{G_R}(0) \\ & \Longrightarrow \; d^+_{G_R}(0) > \underline{m} \\ & \Longrightarrow \; \underline{m} - d^+_{G_R}(0) < 0 \Longrightarrow \lambda_+ = 0 \\ & \Longrightarrow \; |K_{out}| - |K_{in}| = \lambda_+ - (-f_0) = f_0 \end{array}$$

♯

V_{out} **et** V_{in}

L'ensemble $V_{out} = \{v^1_{out}, ..., v^r_{out}\}$ contient f_i copies de tout sommet v_i de G_R (à l'exception de v_0) tel que $f_i > 0$.
L'ensemble $V_{in} = \{v^1_{in}, ..., v^t_{in}\}$ contient $-f_i$ copies de tout sommet v_i de G_R (à l'exception de v_0) tel que $f_i < 0$.
Aucune arête n'existe ni entre les sommets de V_{out}, ni entre les sommets de V_{in}.
Tous les sommets de V_{out} sont reliés à ceux de V_{in}. Le coût d'une arête reliant le sommet v^j_{out} de V_{out}, copie d'un sommet v_a de G_R, au sommet v^i_{in} de V_{in}, copie d'un sommet v_b de G_R est égal à la longueur sp_{ab} de SP_{ab}. Tout sommet v^j_{out} de V_{out}, copie d'un sommet v_a de G_R, est relié aux sommets de K_{in} par une arête de coût égal à sp_{a0}. Tout sommet v^i_{in} de V_{in}, copie d'un sommet v_b de G_R, est relié aux sommets de K_{out} par une arête dont le coût vaut sp_{0b}.
Il n'y a pas d'arête entre les paires de sommets de $V_{out} \times K_{out}$ ni entre celles de $V_{in} \times K_{in}$.

$\underline{L_{in}}$ **et** $\underline{L_{out}}$

Soit ϕ la permutation de l'ensemble des indices des sommets de $V_R \setminus \{v_0\}$ telle que:

$$sp_{0\phi_1} + sp_{\phi_1 0} \leq sp_{0\phi_2} + sp_{\phi_2 0} \leq ... \leq sp_{0\phi_{|V_R|-1}} + sp_{\phi_{|V_R|-1}0}.$$

Soit i^*_ϕ le plus petit entier tel que $d^+_{G_R}(\phi_1) + d^+_{G_R}(\phi_2) + ... + d^+_{G_R}(\phi_{i^*_\phi}) \geq z^+$. Pour chaque sommet v_{ϕ_j} $(j = 1, ..., i^*_\phi)$ nous allons définir une valeur $d_\phi(\phi_j)$ de la manière suivante:

$$d_\phi(\phi_j) = d^+_{G_R}(\phi_j) \text{ si } 1 \leq j < i^*_\phi \text{ et}$$
$$d_\phi(\phi_{i^*_\phi}) = z^+ - (d^+_{G_R}(\phi_1) + d^+_{G_R}(\phi_2) + ... + d^+_{G_R}(\phi_{i^*_\phi - 1})).$$

L'ensemble $L_{in} = \{l^1_{in}, l^2_{in}, ..., l^{z^+}_{in}\}$ contient $d_\phi(\phi_j)$ copies des sommets v_{ϕ_j} de G_R tels que $1 \leq j \leq i^*_\phi$.
L'ensemble L_{out} est identique à l'ensemble L_{in}. Nous noterons l^i_{out} la copie de l^i_{in} dans L_{out} $(i = 1, ..., z^+)$.

Les sommets de L_{out} ne sont pas reliés entre eux dans G_a, les sommets de L_{in} non plus. Tous les sommets L_{out} sont reliés aux sommets de K_{in} dans G_a. Le coût d'une arête reliant un sommet l^j_{out}, copie d'un sommet v_a de G_R, à un sommet de K_{in} est égal à sp_{a0}. Tous les sommets L_{in} sont reliés aux sommets de K_{out} dans G_a. Le coût d'une arête reliant un sommet l^j_{in}, copie d'un sommet v_a de G_R, à un sommet de K_{out} est égal à sp_{0a}.
Chaque sommet $l^j_{out} \in L_{out}$, copie d'un sommet v_a de G_R, est relié aux copies de ce même sommet dans L_{in} par une arête de coût nul.
Il n'y a pas d'arête entre les paires de sommets de $V_{out} \times L_{out}$, $V_{out} \times L_{in}$, $V_{in} \times L_{out}$, ni entre celles de $V_{in} \times L_{in}$.

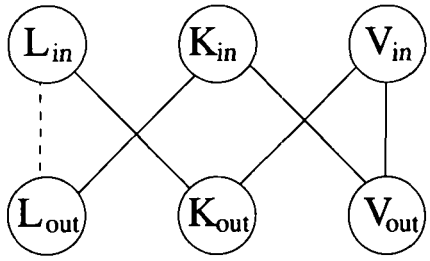


FIG. 2.21 –

Structure générale du graphe auxiliaire G_a.

La figure 2.21 représente les connexions qui existent entre les différents sous-ensembles que nous venons de décrire. Dans cette figure, une arête normale reliant deux sous-ensembles indique que chaque sommet appartenant à un sous-ensemble est relié à tous les sommets de l'autre. Une arête en traitillés signifie qu'un sommet d'un sous-ensemble, copie de $v_i \in G_R$, est relié uniquement aux sommets de l'autre sous-ensemble qui sont également des copies de v_i.

Lemme 2 *Le graphe G_a comporte un nombre pair de sommets.*

preuve:

Les ensembles L_{in} et L_{out} sont identiques. Il nous reste donc à considérer la parité de la somme suivante: $|K_{out}| + |K_{in}| + |V_{in}| + |V_{out}|$.

La somme des degrés entrant des sommets d'un graphe orienté est égale à la somme des degrés sortant. Nous avons donc:

$$\sum_{\substack{v_i \in G_R \\ f_i > 0}} f_i = \sum_{\substack{v_i \in G_R \\ f_i < 0}} |f_i|$$

Par définition des ensembles V_{in} et V_{out}:

$$|V_{in}| + |V_{out}| = \left(\sum_{\substack{v_i \in G_R \\ f_i > 0}} f_i + \sum_{\substack{v_i \in G_R \\ f_i < 0}} |f_i| \right) - |f_0|$$

Par le lemme 1 nous avons également:

$$|K_{out}| + |K_{in}| = |K_{out}| - |K_{in}| + 2|K_{in}| = f_0 + 2|K_{in}|$$

Nous obtenons finalement:

$$\begin{aligned} |K_{out}| + |K_{in}| + |V_{in}| + |V_{out}| &= f_0 - |f_0| + 2|K_{in}| + \sum_{\substack{v_i \in G_R \\ f_i > 0}} f_i + \sum_{\substack{v_i \in G_R \\ f_i < 0}} |f_i| \\ &= f_0 - |f_0| + 2|K_{in}| + 2 \sum_{\substack{v_i \in G_R \\ f_i > 0}} f_i \end{aligned}$$

L'expression $f_0 - |f_0|$ est nulle si $f_0 \geq 0$ et vaut $2f_0$ lorsque $f_0 < 0$. Elle est donc toujours paire. Ainsi nous avons bien que le graphe G_a possède un nombre pair de sommets.

♯

Le lemme 2 nous assure que le graphe auxiliaire G_a possède toujours un nombre pair de sommets. Chercher un couplage parfait dans G_a a donc un sens. Nous allons rechercher dans ce graphe un couplage parfait $M_{min}(G_a)$ de coût minimum. La borne inférieure $\vec{LB}_1$ est alors définie de la manière suivante: $\vec{LB}_1 = C(R) + C(M_{min}(G_a))$.

Avant de démontrer la validité de la borne $\vec{LB}_1$, nous allons illustrer par un exemple la construction du graphe auxiliaire G_a.

Exemple:

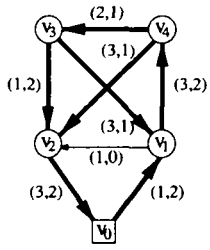


FIG. 2.22 -

Graphe initial G.

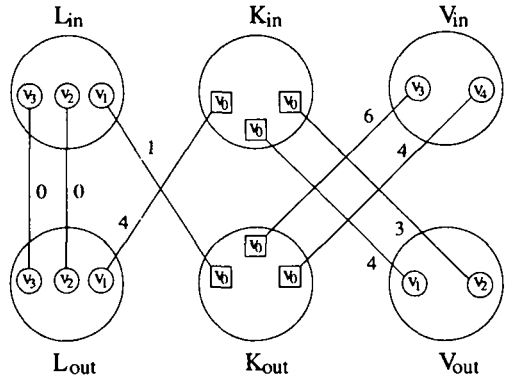


FIG. 2.23 -

Couplage optimal dans le graphe auxiliaire G_a.

Nous avons repris le graphe G de la figure 2.16. La demande totale Q_T vaut 11, la capacité des véhicules 3, $\underline{m}$ est égal à 4, $C(R)$ à 16, z^+ et z^- valent 3.
Un choix possible pour la permutation ϕ des indices des sommets de $V_R \setminus v_0$ est le suivant: $\phi_1 = 2$, $\phi_2 = 1$, $\phi_3 = 3$ et $\phi_4 = 4$. Ce n'est cependant pas la seule permutation satisfaisant la propriété:

$$sp_{0\phi_1} + sp_{\phi_1 0} \leq sp_{0\phi_2} + sp_{\phi_2 0} \leq \ldots \leq sp_{0\phi_4} + sp_{\phi_4 0}$$

Comme $d^+_{G_R}(2) + d^+_{G_R}(1) + d^+_{G_R}(3) = 1 + 1 + 2 = 4 \geq z^+$, la valeur de i^*_ϕ est 3. Nous avons alors $d_\phi(2) = 1$, $d_\phi(1) = 1$ et $d_\phi(3) = 1$. C'est pourquoi les ensembles L_{in} et L_{out} contiennent une copie des sommets v_1, v_2 et v_3.
La figure 2.23 représente le couplage optimal obtenu dans le graphe auxiliaire G_a. Le nombre situé au-dessus de chaque arête correspond à son coût. Le coût total du couplage est de 22. La borne $L\vec{B}1$ a donc la valeur suivante: $L\vec{B}1 = 16 + 22 = 38$.

2.4.1.4 Démonstration de la validité de $L\vec{B}1$

Soit s^* une solution optimale du PTAC dans le graphe $G = (V, A)$. Soit $G_{s^*} = (V, R \cup A_{s^*})$ le graphe augmenté associé à s^*. L'ensemble A_{s^*} représente l'ensemble des arcs artificiels se trouvant dans la solution optimale, c'est-à-dire l'ensemble des arcs qui sont traversés sans être desservis dans s^*. Nous noterons $G_I = (V, A_{s^*})$ le graphe partiel de G_{s^*} constitué des arcs de A_{s^*}. Soit $C(A_{s^*})$ la longueur totale des arcs se trouvant dans A_{s^*}. La valeur de s^*, que nous noterons C^*, est égale à $C(R) + C(A_{s^*})$.

A partir des arcs de A_{s^*}, nous allons construire un couplage parfait, $M(G_a)$ dans le graphe auxiliaire G_a. Nous allons montrer que le coût $C(M(G_a))$ de ce couplage est toujours inférieur ou égal à $C(A_{s^*})$. Nous aurons ainsi démontré la validité de la borne $L\vec{B}1$ puisque:

$$L\vec{B}1 = C(R) + C(M_{min}(G_a)) \leq C(R) + C(M(G_a)) \leq C(R) + C(A_{s^*}) = C^*.$$

2.4.1.4.1 Construction du couplage $M(G_a)$

Nous allons tout d'abord décrire la manière dont le couplage parfait $M(G_a)$ est construit dans G_a. Une justification des différentes étapes de l'algorithme suivra.

Construction du couplage parfait $M(G_a)$.

Etape 0. Poser $M(G_a) := \emptyset$, $A'_{s^\bullet} := A_{s^\bullet}$ et $G'_I := (V, A'_{s^\bullet})$. Le graphe G'_I est le graphe partiel de $G_{s^\bullet}$ composé des arcs de $A'_{s^\bullet}$.

Etape 1. Tant qu'il existe dans V_{out} un sommet non couplé faire:

> Choisir un sommet non couplé v^i_{out} de V_{out}. Soit v_k le sommet de G'_I dont v^i_{out} est la copie. Considérer dans G'_I un chemin artificiel de type II quittant v_k. Ce chemin relie dans G'_I v_k à un sommet v_q tel que $f_q < 0$. Soit P_{kq} ce chemin. En partant de v_k, déterminer le premier sommet, $v_j \in P_{kq}$, possédant une copie non couplée dans $K_{in} \cup V_{in}$. Notons P_{kj} ($\subseteq P_{kq}$) le chemin artificiel reliant v_k à v_j dans G'_I.
> Ajouter à $M(G_a)$ une arête reliant v^i_{out} à une copie de v_j non couplée se trouvant dans $K_{in} \cup V_{in}$. Poser $A'_{s^\bullet} := A'_{s^\bullet} \setminus P_{kj}$.

Etape 2. Tant qu'il existe un sommet non couplé dans V_{in} faire:

> Choisir un sommet v^i_{in} non couplé de V_{in}. Soit v_q le sommet de G'_I dont il est la copie. Déterminer dans G'_I un chemin, P_{0q} reliant le dépôt à v_q. Ajouter à $M(G_a)$ une arête reliant un sommet non couplé de K_{out} à v^i_{in}.
> Poser $A'_{s^\bullet} := A'_{s^\bullet} \setminus P_{0q}$.

Si tous les sommets de K_{out} sont couplés aller à l'étape 4.
Sinon poser n_{out} égal au nombre de sommets non couplés de K_{out}.

Etape 3. Tant qu'il existe des sommets de K_{out} non couplés faire:

> Déterminer le plus petit indice k, $(1 \leq k \leq i^*_\phi)$ tel que v_{ϕ_k} possède une copie non couplée l^i_{in} dans L_{in}. Ajouter à $M(G_a)$ une arête reliant un sommet non couplé de K_{out} à l^i_{in} ainsi qu'une arête reliant le sommet l^i_{out} de L_{out} à un sommet non couplé de K_{in}.

Enlever de $A'_{s^\bullet}$ n_{out} circuits qui débutent par un chemin artificiel de type I quittant le dépôt.

Etape 4. Pour chaque sommet l^i_{in} non couplé de L_{in}, ajouter à $M(G_a)$ une arête reliant le sommet l^i_{in} au sommet l^i_{out} de L_{out}.
Poser $A'_{s^\bullet} := \emptyset$.

2.4.1.4.2 Justifications

Lemme 3 *Soit v_k un sommet de G_R tel que $f_k > 0$. v_k possède n_k copies non couplées dans V_{out} si et seulement s'il existe n_k chemins artificiels de type II ayant v_k pour origine dans G'_I.*

preuve:

Avant de débuter l'étape 1, G'_I est égal à G_I. Comme le graphe G_{s^*} est symétrique, il y a f_k chemins artificiels de type II quittant v_k dans G_I. Comme $M(G_a) = \emptyset$, il y a également f_k copies non couplées de v_k dans V_{out}.

En effectuant l'étape 1, chaque fois qu'une copie de v_k se trouvant dans V_{out} est couplée, un morceau P_{kj} d'un chemin artificiel de type II partant de v_k est enlevé de A'_{s^*}. Réciproquement lorsqu'une portion P_{kj} de chemin artificiel de type II quittant v_k est ôtée de A'_{s^*}, une copie de v_k se trouvant dans V_{out} est couplée. Le nombre de chemins artificiels de type II quittant v_k diminue donc d'une unité si et seulement si une copie de v_k se trouvant dans V_{out} est couplée.

♯

Le lemme 3 nous assure qu'il existe toujours un chemin artificiel de type II qui quitte un sommet v_k possédant une copie non couplée dans V_{out}.

Lemme 4 *A la fin de l'étape 1, les sommets v_k différents du dépôt et tels que $f_k \geq 0$ sont symétriques dans G'_I.*

preuve:

Soit v_k un sommet de G'_I, différent du dépôt et tel que $f_k \geq 0$.
Au début de l'étape 1, $G'_I = G_I$. Comme G_{s^*} est symétrique, nous avons:

$$d^+_{G'_I}(k) - d^-_{G'_I}(k) = d^-_{G_R}(k) - d^+_{G_R}(k) = f_k$$

A la fin de l'étape 1, toutes les copies de v_k se trouvant dans V_{out} sont couplées. Trois types de chemins ont été supprimés de G'_I au cours de l'étape 1:

i) f_k chemins artificiels ayant v_k comme origine
ii) des chemins artificiels ne contenant pas v_k
iii) un nombre $\gamma_k \geq 0$ de chemins artificiels traversant v_k, c'est-à-dire entrant et sortant de v_k

Nous avons ainsi à la fin de l'étape 1:

$$d^+_{G'_I}(k) - d^-_{G'_I}(k) = (d^+_{G_I}(k) - f_k - \gamma_k) - (d^-_{G_I}(k) - \gamma_k) = 0$$

Le sommet v_k est donc bien symétrique dans G'_I à la fin de l'étape 1.

♯

Lemme 5 *Soit v_q un sommet différent du dépôt tel que $f_q < 0$. A la fin de l'étape 1: v_q possède n_q copies non couplées dans V_{in} $\Leftrightarrow d^+_{G'_I}(q) - d^-_{G'_I}(q) = -n_q$.*

preuve:

Avant le début de l'étape 1, v_q possède $-f_q$ copies non couplées dans V_{in}. De plus $G'_I = G_I$ ce qui signifie que $d^+_{G'_I}(q) - d^-_{G'_I}(q) = d^-_{G_R}(q) - d^+_{G_R}(q) = f_q$.

Au cours de l'étape 1, chaque fois qu'une copie de v_q se trouvant dans V_{in} est couplée, un chemin artificiel se terminant en v_q est ôté de $A'_{s\bullet}$. Réciproquement, chaque fois qu'un chemin artificiel se terminant en v_q est enlevé de $A'_{s\bullet}$, une copie de v_q se trouvant dans V_{in} est couplée.

Si, à la fin de l'étape 1, il reste n_q copies non couplées de v_q dans V_{in}, cela signifie que ses $-f_q - n_q$ autres copies se trouvant dans V_{in} sont couplées. Il y a donc $-f_q - n_q$ chemins artificiels se terminant en v_q qui sont supprimés de $A'_{s\bullet}$ au cours de l'étape 1. Les autres chemins supprimés de $A'_{s\bullet}$ soit ne passent pas par v_q soit traversent v_q. Soit γ_q le nombre de chemins enlevés de $A'_{s\bullet}$ et traversant v_q. Par un raisonnement identique à celui du lemme 4 nous avons alors:

$$d^+_{G'_I}(q) - d^-_{G'_I}(q) = d^+_{G_I}(q) - \gamma_q - (d^-_{G_I}(q) - (-f_q - n_q) - \gamma_q) = -n_q.$$

Si nous avons $d^+_{G'_I}(q) - d^-_{G'_I}(q) = -n_q$ à la fin de l'étape 1, cela signifie que la différence, dans G'_I, entre le nombre d'arcs sortant et le nombre d'arcs entrant dans v_q a augmenté de $-n_q - f_q$. Comme au cours de l'étape 1, on ne fait que supprimer des chemins de $A'_{s\bullet}$, cette augmentation n'est due qu'à une diminution du nombre d'arcs entrant dans v_q. Il y a donc $-n_q - f_q$ chemins se terminant en v_q qui ont été supprimés de $A'_{s\bullet}$ en effectuant l'étape 1. Chaque fois qu'un de ces chemins est supprimé, une copie de v_q se trouvant dans V_{in} est couplée. Il y a ainsi $-n_q - f_q$ copies de v_q qui sont couplées et donc $-f_q - (-n_q - f_q) = n_q$ copies de v_q non couplées dans V_{in}.

♯

Par les lemmes 4 et 5, nous savons que les seuls sommets non symétriques dans G'_I à la fin de l'étape 1 sont les sommets qui possèdent des copies non couplées dans V_{in} et le dépôt (si tous les sommets de V_{in} sont couplés alors G'_I est symétrique). Celui-ci est de plus le seul sommet qui a, dans G'_I, plus d'arcs sortant que d'arcs entrant. Tout sommet $v_q \neq v_0$ ayant n_q copies non couplées dans V_{in} est alors l'extrémité d'au moins n_q chemins de G'_I originaires du dépôt. Il est donc possible d'associer, à chaque sommet non couplé de V_{in} copie d'un sommet v_q de G_R, un chemin dans G'_I quittant v_0 et se terminant en v_q. C'est ce qui est fait à l'étape 2.

A la fin de l'étape 2, tous les sommets de V_{in} sont couplés. En utilisant un raisonnement semblable à celui employé dans la démonstration du lemme 4, nous pouvons montrer qu'à la fin de l'étape 2, tous les sommets de G'_I sont symétriques. Ainsi si G'_I contient encore des arcs à la fin de l'étape 2, alors tous ces arcs appartiennent à des circuits.

Lemme 6 *Soit n_{out} le nombre de sommets non couplés de K_{out} et soit n_{in} celui de K_{in}. A la fin de l'étape 2, $n_{out} = n_{in}$.*

preuve:

Au cours des étapes 1 et 2, quatre types de chemins artificiels ont été supprimés de G'_I:

i) $|K_{out}| - n_{out}$ chemins artificiels ayant v_0 comme origine

ii) $|K_{in}| - n_{in}$ chemins artificiels se terminant en v_0

iii) des chemins artificiels ne contenant pas v_0

iv) un nombre $\gamma_0 \geq 0$ de chemins artificiels traversant v_0, c'est-à-dire entrant et sortant de v_0

Comme tous les sommets de G'_I sont symétriques à la fin de l'étape 2, nous avons:

$$\begin{aligned} d^+_{G'_I}(0) - d^-_{G'_I}(0) &= d^+_{G_I}(0) - (\gamma_0 + |K_{out}| - n_{out}) - (d^-_{G_I}(0) - (\gamma_0 + |K_{in}| - n_{in})) \\ &= d^+_{G_I}(0) - d^-_{G_I}(0) - |K_{out}| + |K_{in}| + n_{out} - n_{in} = 0 \end{aligned}$$

Nous savons que $d^+_{G_I}(0) - d^-_{G_I}(0) = f_0$. De plus par le lemme 1 nous avons $|K_{out}| - |K_{in}| = f_0$. Nous obtenons finalement:

$$d^+_{G'_I}(0) - d^-_{G'_I}(0) = f_0 - f_0 + n_{out} - n_{in} = 0 \Rightarrow n_{out} = n_{in}$$

♯

Lemme 7 *Soit n_{out} le nombre de sommets non couplés appartenant à K_{out} à la fin de l'étape 2, et soit n_{in} le nombre de sommets non couplés appartenant à K_{in} à la fin de cette même étape.*
Dans G'_I, à la fin de l'étape 2, il y a au moins n_{out} chemins artificiels de type I quittant le dépôt et n_{in} chemins artificiels de type I aboutissant au dépôt.

preuve:

Les ensembles K_{out} et K_{in} ont été définis de telle sorte que $|K_{out}| = z^+$ correspond au nombre minimum de chemins artificiels de type I quittant le dépôt et $|K_{in}| = z^-$ représente le nombre minimum de chemins artificiels de type I arrivant au dépôt.

Comme il reste n_{out} sommets non couplés dans K_{out}, $|K_{out}| - n_{out}$ sommets de K_{out} ont déjà été couplés au cours des étapes 1 et 2. Au plus $|K_{out}| - n_{out}$ chemins artificiels de type I quittant le dépôt ont ainsi été ôtés de G'_I. Il doit donc rester au moins n_{out} chemins artificiels de type I quittant le dépôt dans G'_I.
De manière similaire nous montrons que G'_I contient au moins n_{in} chemins artificiels de type I qui se terminent au dépôt.

♯

Le lemme 7 nous indique qu'au cours de l'étape 3, il est possible d'associer à chaque sommet non couplé de K_{out} un circuit dans G'_I débutant par un chemin artificiel de type I qui a le dépôt pour origine.

Nous allons noter par $T'_1, T'_2, ..., T'_{n_{out}}$ les circuits ôtés de G'_I à la fin de l'étape 3. Ces circuits débutent tous par un chemin artificiel de type I quittant le dépôt. L'extrémité de ces chemins est un sommet de G_R différent du dépôt. Ainsi les circuits T'_j $(1 \leq j \leq n_{out})$ contiennent au moins deux sommets de G_R: le dépôt et l'extrémité du chemin artificiel de type I qui le quitte. Soient $C(T'_1), ..., C(T'_{n_{out}})$ les coûts de ces circuits. Nous supposerons que ces circuits sont numérotés selon leur longueur non décroissante, c'est-à-dire que: $C(T'_1) \leq C(T'_2) \leq ... \leq C(T'_{n_{out}})$.

Rappelons que ϕ est la permutation des indices des sommets de G_R qui satisfait:

$$sp_{0\phi_1} + sp_{\phi_1 0} \leq sp_{0\phi_2} + sp_{\phi_2 0} \leq ... \leq sp_{0\phi_{|V_R|-1}} + sp_{\phi_{|V_R|-1}0}.$$

Soit i_{out} le plus petit entier tel que $d^+_{G_R}(\phi_1) + d^+_{G_R}(\phi_2) + ... + d^+_{G_R}(\phi_{i_{out}}) \geq n_{out}$. Pour chaque sommet v_{ϕ_j} $(j = 1, ..., i_{out})$ nous allons définir une valeur $d'(\phi_j)$ de la manière suivante:

$$d'(\phi_j) = d^+_{G_R}(\phi_j) \text{ si } 1 \leq j < i_{out} \text{ et}$$
$$d'(\phi_{i_{out}}) = n_{out} - (d^+_{G_R}(\phi_1) + d^+_{G_R}(\phi_2) + ... + d^+_{G_R}(\phi_{i_{out}-1})).$$

Soit T_{ϕ_j} le circuit de longueur minimum dans G contenant à la fois le dépôt et $v_{\phi_j} \in G_R$ et soit $C(T_{\phi_j})$ sa longueur $(j = 1, ..., i_{out})$. Cette longueur est égale à $sp_{0\phi_j} + sp_{\phi_j 0}$.

Lemme 8 *Soit n_{out} le nombre de sommets non couplés dans K_{out} à la fin de l'étape 2.*

a) $C(T_{\phi_1}) \leq C(T'_j), \ \forall\ 1 \leq j \leq n_{out}$.

b) *Si $i_{out} > 1$, $\forall\ 1 < k \leq i_{out}$ nous avons:*
$C(T_{\phi_k}) \leq C(T'_j), \ \forall\ d'(\phi_1) + ... + d'(\phi_{k-1}) + 1 \leq j \leq n_{out}$.

preuve:

T_{ϕ_1} est le plus petit circuit contenant le dépôt et un sommet de G_R différent de v_0. Comme les circuits T'_j $(j = 1, ..., n_{out})$ contiennent également le dépôt et un sommet de G_R différent du dépôt, nous avons $C(T_{\phi_1}) \leq C(T'_j), \ \forall\ 1 \leq j \leq n_{out}$. Le point **a)** est ainsi démontré.

Nous allons démontrer le point **b)** par induction sur k.

Initialisation: $C(T_{\phi_2}) \leq C(T'_j),\ \forall\ d'(\phi_1)+1 \leq j \leq n_{out}$

Le cas où $C(T_{\phi_2}) = C(T_{\phi_1})$ est évident (voir point **a**)).
Nous allons considérer la situation où $C(T_{\phi_2}) > C(T_{\phi_1})$. Dans ce cas, le circuit T_{ϕ_1} ne contient pas d'autre sommet de G_R que v_0 et v_{ϕ_1}. En effet, si T_{ϕ_1} contenait un sommet $v_k \in G_R$ différent de v_0 et de v_{ϕ_1}, nous aurions:

$$sp_{0\phi_1} + sp_{\phi_1 0} = sp_{0k} + sp_{k0} < sp_{0\phi_2} + sp_{\phi_2 0}.$$

Cela n'est pas possible vu la façon dont la permutation ϕ est définie.

Il y a, dans G_{s^*} et donc dans G'_I, au plus $d'(\phi_1) = d^+_{G_R}(\phi_1)$ chemins artificiels de type I quittant le dépôt et se terminant en v_{ϕ_1}. Il existe ainsi, dans G'_I, au plus $d'(\phi_1)$ circuits débutant par un chemin artificiel de type I et reliant le dépôt à v_{ϕ_1}. Les seuls circuits T'_j, $(j = 1, ..., n_{out})$ pouvant être de longueur égale à $C(T_{\phi_1})$ sont alors les circuits: $T'_1, T'_2, ..., T'_{d'(\phi_1)}$. Le circuit $T'_{d'(\phi_1)+1}$ est donc de coût supérieur à $C(T_{\phi_1})$, et nous avons:

$$C(T_{\phi_2}) \leq C(T'_{d'(\phi_1)+1}) \leq C(T'_{d'(\phi_1)+2}) \leq ... \leq C(T'_{n_{out}}).$$

Induction: Supposons que le point **b)** soit vérifié pour tous les circuits T_{ϕ_i} tels que $1 < i \leq k-1$ et montrons que ce point est vérifié pour le circuit T_{ϕ_k}.

Posons $q = d'(\phi_1) + ... + d'(\phi_{k-1}) + 1$.
Comme les circuits T'_j sont numérotés selon leur longueur croissante, il suffit de démontrer que $C(T_{\phi_k}) \leq C(T'_q)$ pour prouver que T_{ϕ_k} vérifie **b)**.

Supposons que $C(T_{\phi_k}) > C(T'_q)$. Il existe alors ϕ_r avec $r < k$ tel que $C(T_{\phi_r}) = C(T'_q)$. Par hypothèse d'induction et par définition de la permutation ϕ, nous avons:

$$\begin{array}{lllllll} & C(T'_q) & = & C(T_{\phi_r}) & \leq & C(T_{\phi_{k-1}}) & \leq & C(T'_q) \\ \Rightarrow & C(T_{\phi_{k-1}}) & = C(T'_q) & & & & & \end{array}$$

Comme $C(T_{\phi_{k-1}}) \leq C(T_{\phi_k})$ nous avons que $C(T'_q) \leq C(T_{\phi_k})$, ce qui contredit la supposition selon laquelle $C(T_{\phi_k}) > C(T'_q)$. Il est donc faux de supposer que $C(T_{\phi_k}) > C(T'_q)$ et ainsi $C(T_{\phi_k}) \leq C(T'_q)$.

♯

Lemme 9 $\displaystyle\sum_{j=1}^{i_{out}} (d'(\phi_j) \cdot C(T_{\phi_j})) \leq \sum_{j=1}^{n_{out}} C(T'_j)$

preuve:

Si $i_{out} = 1$, alors $d'(\phi_1) = n_{out}$. Par le point **a)** du lemme 8, nous avons $n_{out} C(T_{\phi_1}) \leq \displaystyle\sum_{j=1}^{n_{out}} C(T'_j)$.

Lorsque $i_{out} > 1$, nous pouvons utiliser le point **b)** du lemme 8. Nous obtenons alors:

$$\begin{array}{lcl}
d'(\phi_1) \cdot C(T_{\phi_1}) & \leq & C(T'_1) + \ldots + C(T'_{d'(1)}) \\
d'(\phi_2) \cdot C(T_{\phi_2}) & \leq & C(T'_{d'(1)+1}) + \ldots + C(T'_{d'(1)+d'(2)}) \\
\vdots & & \vdots \\
d'(\phi_{i_{out}}) \cdot C(T_{\phi_{i_{out}}}) & \leq & C(T'_{d'(\phi_1)+\ldots+d'(\phi_{i_{out}-1})+1}) + \ldots + C(T'_{d'(\phi_1)+\ldots+d'(\phi_{i_{out}})})
\end{array}$$

En sommant les termes à gauche de l'inégalité et prenant la somme des membres de droite, nous obtenons finalement:

$$\sum_{j=1}^{i_{out}} (d'(\phi_j) \cdot C(T_{\phi_j})) \leq \sum_{j=1}^{n_{out}} C(T'_j)$$

‡

Lemme 10 *Le coût du couplage parfait $M(G_a)$ obtenu à la fin de l'étape 4 est inférieur ou égal à la longueur totale des arcs de A^*_s.*

preuve:

Au cours des étapes 1 et 2 de la construction de $M(G_a)$, chaque fois qu'un chemin P_{kj} reliant v_k à v_j est enlevé de $A'_{s\bullet}$, une arête de coût égal à sp_{kj} est ajoutée à $M(G_a)$. Par définition, sp_{kj} est le coût du plus court chemin reliant v_k à v_j dans G. La longueur de P_{kj} est donc toujours supérieure ou égale à sp_{kj}.
A la fin de l'étape 2, l'ensemble $A_{s\bullet} \setminus A'_{s\bullet}$ contient tous les chemins supprimés de $A'_{s\bullet}$ aux étapes 1 et 2. Le coût total de ces chemins vaut $C(A_{s\bullet}) - C(A'_{s\bullet})$. Nous avons donc à la fin de l'étape 2:

$$C(M(G_a)) \leq C(A_{s\bullet}) - C(A'_{s\bullet}).$$

La longueur totale des arêtes ajoutées à $M(G_a)$ lors de l'étape 3 est égale à $\sum_{j=1}^{i_{out}} (d'(\phi_j) \cdot C(T_{\phi_j}))$ et celle des circuits supprimés de $A'_{s\bullet}$ vaut $\sum_{j=1}^{n_{out}} C(T'_j)$.
Par le lemme 9, nous avons à la fin de l'étape 3:

$$C(M(G_a)) \leq C(A_{s\bullet}) - C(A'_{s\bullet}).$$

Finalement à l'étape 4, le coût des arêtes insérées dans $M(G_a)$ est nul. L'ensemble $A'_{s\bullet}$ est vide à la fin de cette étape. A la fin de la dernière étape de l'algorithme, nous obtenons bien:

$$C(M(G_a)) \leq C(A_{s\bullet}) - C(A'_{s\bullet}) = C(A_{s\bullet}).$$

‡

Théorème 1 *Soit $M_{min}(Ga)$ un couplage parfait de coût minimum dans G_a.*
$\vec{LB}1 = C(R)+C(M_{min}(G_a))$ est une borne inférieure pour le PTAC dans le cas orienté.

preuve:

Soit C^* le coût de la solution optimale s^*, $C^* = C(R) + C(A_{s^*})$. Par le lemme 10, nous savons que $C(M(G_a)) \leq C(A_{s^*})$. Comme $M_{min}(G_a)$ est un couplage parfait de coût minimum, nous avons également: $C(M_{min}(G_a)) \leq C(M(G_a))$. Finalement nous obtenons:

$$\vec{LB}1 = C(R) + C(M_{min}(G_a)) \leq C(R) + C(M(G_a)) \leq C(R) + C(A_{s^*}) = C^*$$

La valeur de $\vec{LB}1$ est toujours inférieure ou égale à la valeur d'une solution optimale du problème de tournées sur les arcs avec capacité dans un graphe orienté. $\vec{LB}1$ est donc bien une borne inférieure pour le PTAC dans le cas orienté.

♯

Remarque:

Nous avons présenté les bornes inférieures $LB1$ et $\vec{LB}1$ en utilisant $\underline{m}$ pour estimer le nombre minimum de véhicules nécessaires au service de tous les clients. Nous aurions pu remplacer $\underline{m}$ par n'importe quelle autre borne inférieure sur le nombre de véhicules à utiliser. Tous les résultats présentés dans cette section restent valables quelle que soit la manière dont la borne inférieure sur le nombre de véhicules minimum nécessaires est calculée.

2.4.2 Procédures de base

Cette section est consacrée à la description de procédures qui seront utilisées dans les méthodes heuristiques que nous avons développées pour le PTAC orienté. Ces procédures sont essentiellement des adaptations au cas orienté de procédures qui existent pour le cas non orienté et que nous avons décrites précédemment.

2.4.2.1 Adaptation de la procédure RACCOURCIR au cas orienté

La procédure RACCOURCIR décrite pour le cas non orienté ne peut pas être utilisée sans modification dans le cas orienté. Nous redonnons ci-dessous la description de cette procédure pour le cas non orienté. Cette description est identique à celle présentée à la section 1.2.2.5.

Algorithme: RACCOURCIR

ENTREE: Un cycle T dans $G = (V, E)$ desservant un ensemble $R_T \subseteq R$ d'arêtes.

SORTIE: Un cycle desservant les arêtes de R_T de coût au pire égal à celui de T.

Etape 1. Choisir une orientation de T et un sommet v_i de T. On considère v_i comme étant le premier sommet de T et on suppose que le service d'une arête de R_T se fait lors de sa dernière apparition dans T.

Etape 2. Déterminer le sommet v_j de T origine de la première arête desservie dans T rencontrée en partant de v_i. La chaîne P reliant v_i à v_j dans T est une chaîne inutile.

Etape 3. Soit Q la chaîne reliant v_j à v_i dans T ($Q = T \setminus P$).
Si toutes les arêtes de Q entrant dans un sommet identique à v_j sont des arêtes desservies, aller à l'étape 4.
Soit (v_k, v_j) une arête non desservie dans Q. Considérer le cycle C=$(v_j, ..., v_k, v_j)$, renverser son orientation dans T et aller à l'étape 2.

Etape 4. Si la longueur de la plus courte chaîne SC_{ij} reliant v_i à v_j dans G est inférieure à celle de P, remplacer P par SC_{ij}.

Etape 5. Répéter les étapes 2 à 4 en considérant les deux orientations possibles de T et tous les sommets de départ v_i de T jusqu'à ce qu'aucune amélioration ne puisse plus être obtenue.

Dans le cas orienté, une seule orientation est considérée qui est celle du circuit à raccourcir. Aucun choix d'orientation n'est alors effectué à l'étape 1 et une seule orientation est prise en compte à l'étape 5. Les étapes 2 et 4 ne nécessitent aucune adaptation. Finalement, seule l'étape 3 nécessite de profondes modifications dans le cas orienté. En effet, l'orientation de certains arcs est renversée au cours de cette étape ce qui peut ainsi faire intervenir dans le circuit traité des arcs qui n'existent pas dans le réseau initial.
Deux adaptations de RACCOURCIR au cas orienté seront étudiées. La première, que nous appellerons RACCOR_SIMPLE, consiste simplement à supprimer l'étape 3. Dans la

deuxième adaptation, nommée RACCOR, l'étape 3 est remplacée par une étape 3'. Le but de cette nouvelle étape reste cependant le même que celui de l'étape 3, à savoir, tenter d'augmenter la taille du chemin inutile P qui va être raccourci. Cela se fera à l'aide de la procédure ALLONGER.
Avant de présenter la procédure ALLONGER, rappelons brièvement la notation introduite à la section 1.3. Soient $P = (x, ..., y)$ et $P' = (y, ...z)$ deux chemins , $P + P'$ représente le chemin obtenu en mettant bout à bout P et P'. Nous avons alors $P+P' = (x, ..., y, ..., z)$.

Algorithme: ALLONGER

ENTREE: Un circuit T dans $G = (V, A)$ desservant un ensemble $R_T \subseteq R$ d'arêtes. On suppose que le sommet v_i est le premier sommet de T et que le service d'un arc de R_T s'effectue lors de sa dernière apparition dans T. Un chemin inutile $P = (v_i, ..., v_j)$ ayant pour origine le sommet v_i et pour extrémité le sommet v_j, origine du premier arc desservi dans T.

SORTIE: Un circuit T' desservant les arcs de R_T et un chemin inutile $P' \in T'$ ayant v_i comme origine et contenant P.

Etape 1. Soit $Q = T \setminus P = (v_j, v_{q_1}, ..., v_{q_t}, v_i)$.
Par définition du chemin P, l'arc (v_j, v_{q_1}) est desservi dans T.
Chercher dans Q un arc non desservi, $(v_{q_s}, v_{q_{s+1}})$, tel que $v_{q_s} = v_j$.
Si un tel arc n'existe pas dans Q alors poser $P' := P$ et $T' := T$, STOP.
Sinon poser $Q' := (v_j, v_{q_1}, ..., v_{q_s}, v_{q_{s+1}})$.

Etape 2. Si (v_j, v_{q_1}) est le seul arc desservi de Q' alors poser $v_{q_r} := v_{q_s}$. Sinon v_{q_r} est l'origine du deuxième arc desservi de Q'.
Poser $P' := P + (v_j, v_{q_1}, ..., v_{q_r})$ et $Q'' := (v_{q_r}, v_{q_{r+1}}, ..., v_{q_s})$.
Poser $T' := P' + Q'' + (v_{q_s}, v_{q_1}) + SP_{q_1 q_{s+1}} + (v_{q_{s+1}}, v_{q_{s+2}}, ..., v_{q_t}, v_i)$.

Etape 3. Soient $L_{Q'}$ et $L_{Q''}$ les longueurs de Q' et Q'' respectivement.
Poser $L_1 := sp_{ij} + L_{Q'}$ et $L_2 := sp_{iq_r} + L_{Q''} + c_{q_s q_1} + sp_{q_1 q_{s+1}}$.
Si $L_1 \leq L_2$ poser $P' := P$ et $T' := T$.

Le principe de la procédure ALLONGER consiste à essayer de déplacer le service de l'arc (v_j, v_{q_1}) de manière à pouvoir augmenter la taille du chemin inutile P. Par définition de P, l'arc (v_j, v_{q_1}) n'apparaît qu'une fois dans Q. En effet, nous avons supposé qu'un client est desservi lors de sa dernière apparition dans T. Il faut donc, pour déplacer le service de (v_j, v_{q_1}), introduire une copie supplémentaire de cet arc dans Q. Pour cela, un arc non desservi $(v_{q_s}, v_{q_{s+1}}) = (v_j, v_{q_{s+1}})$ est recherché dans Q. Si un tel arc existe, une tournée T' est construite en remplaçant $(v_{q_s}, v_{q_{s+1}})$ par le chemin $(v_{q_s}, v_{q_1}) + SP_{q_1 q_{s+1}}$ (figure 2.25).
Dans T', un chemin inutile P' est obtenu en prolongeant le chemin P jusqu'à l'origine du prochain arc desservi de T'.
On détermine finalement s'il est plus avantageux de raccourcir P' (i.e. remplacer P' par un plus court chemin reliant ses extrémités) dans T' ou de raccourcir P dans T (voir figures 2.26 et 2.27). La procédure ALLONGER retourne la tournée et le chemin inutile correspondant à la meilleure de ces deux variantes.

Les figures qui suivent illustrent le fonctionnement de la procédure ALLONGER.

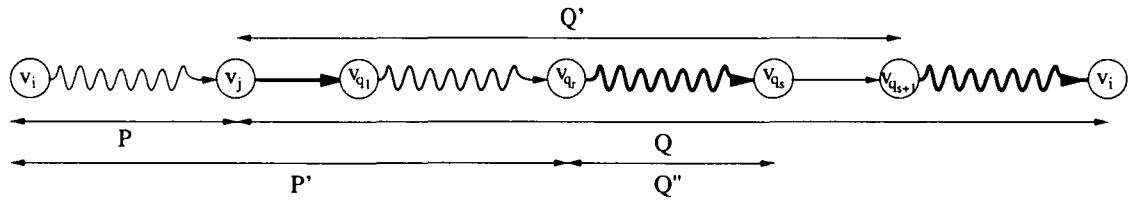


FIG. 2.24 -

Tournée T.

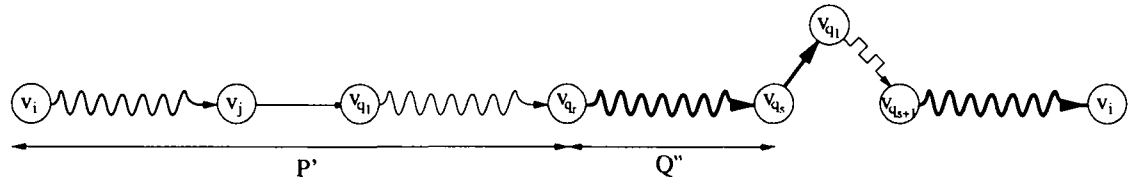


FIG. 2.25 -

Tournée T': le service de l'arc (v_j, v_{q_1}) est déplacé sur l'arc (v_{q_s}, v_{q_1}).

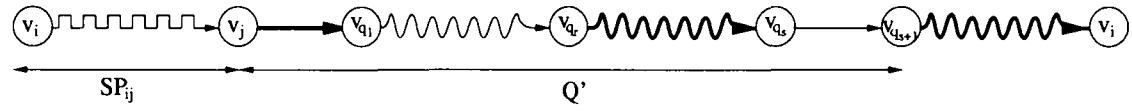


FIG. 2.26 -

Si le chemin inutile P de T est remplacé par le plus court chemin, SP_{ij}, reliant v_i à v_j, alors la longueur, L_1, du tronçon de la tournée ainsi obtenue situé entre v_i et $v_{q_{s+1}}$ est égale à $sp_{ij} + L_{Q'}$.

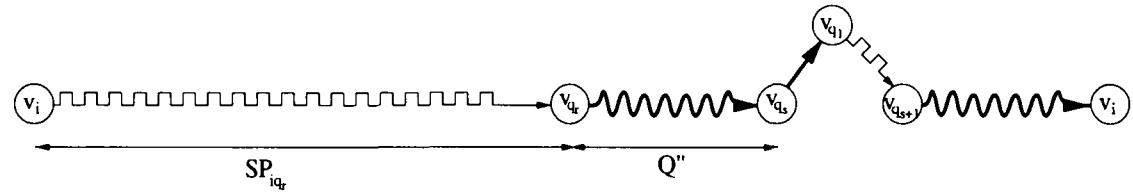
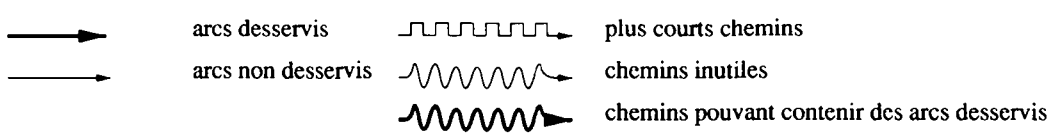


FIG. 2.27 -

Si le chemin inutile P' de T' est remplacé par le plus court chemin, SP_{iq_r}, reliant v_i à v_{q_r}, alors la longueur, L_2, du tronçon de la tournée ainsi obtenue situé entre v_i et $v_{q_{s+1}}$ est égale à $sp_{iq_r} + L_{Q''} + c_{q_s q_1} + sp_{q_1 q_{s+1}}$.



Nous pouvons maintenant décrire la procédure RACCOR de la manière suivante:

Algorithme: RACCOR

ENTREE: Un circuit T dans $G = (V, A)$ desservant un ensemble $R_T \subseteq R$ d'arcs.

SORTIE: Un circuit desservant les arcs de R_T de coût au pire égal à celui de T.

Etape 1. Choisir un sommet v_i de T. On considère v_i comme étant le premier sommet de T et on suppose que le service d'un arc de R_T se fait lors de sa dernière apparition dans T.

Etape 2. Soit v_j le sommet de T origine du premier arc desservi dans T rencontré en partant de v_i. Le chemin P reliant v_i à v_j dans T est un chemin inutile.

Etape 3'. Soit Q le chemin reliant v_j à v_i dans T.
Appliquer la procédure ALLONGER au circuit $T = P + Q$.
Soient T' et P' la tournée et le chemin inutile fournis par ALLONGER.
Poser $T := T'$ et $P := P'$.

Etape 4. Si la longueur du plus court chemin SP_{ij} reliant v_i à v_j dans G est inférieure à celle de P, remplacer P par SP_{ij}.

Etape 5. Répéter les étapes 1 à 4 en considérant tous les sommets de départ v_i de T jusqu'à ce qu'aucune amélioration ne puisse plus être obtenue.

La procédure RACCOR_SIMPLE est identique à la procédure décrite ci-dessus si ce n'est que l'étape 3' n'est pas effectuée. Bien que nécessitant un nombre d'étapes plus important que RACCOR_SIMPLE, la procédure RACCOR ne donne pas forcément de meilleurs résultats comme le montre l'exemple ci-dessous.

Exemple:

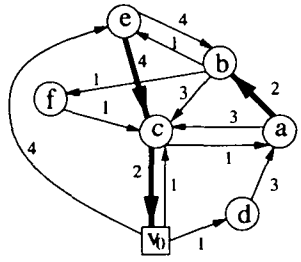


FIG. 2.28 -

Graphe initial G.

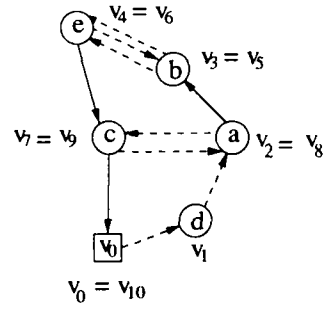


FIG. 2.29 -

Tournée $T = (v_0, d, a, b, e, b, e, c, a, c, v_0)$ à raccourcir.

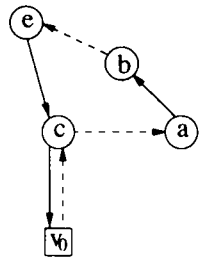


FIG. 2.30 -

Tournée fournie par RACCOR_SIMPLE.

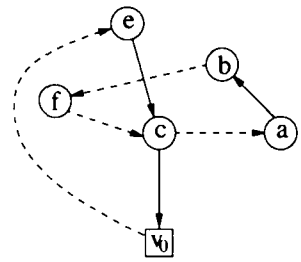


FIG. 2.31 -

Tournée fournie par RACCOR.

Dans la figure 2.28, les arcs en gras représentent les arcs à desservir. Les nombres au-dessus des arcs correspondent à leur longueur. La tournée T de la figure 2.29 dessert tous les clients. Les arcs en traitillés sont les arcs traversés sans être desservis. Les sommets de T sont numérotés: $T = (v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}) = (v_0, d, a, b, e, b, e, c, a, c, v_0)$. Lorsque la procédure RACCOR_SIMPLE est appliquée à T, les trois chemins inutiles (v_0, v_1, v_2), (v_3, v_4, v_5, v_6) et (v_7, v_8, v_9) sont successivement remplacés par les plus courts chemins $SP_{02} = (v_0, c, a)$, $SP_{36} = (b, e)$ et $SP_{79} = \emptyset$. La tournée ainsi obtenue est représentée dans la figure 2.30. La longueur de cette tournée est de 11.
En appliquant RACCOR à la tournée T, nous avons, avant de débuter l'étape 3', un chemin inutile $P = (v_0, v_1, v_2)$ et un chemin $Q = (v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10})$. La procédure ALLONGER est ensuite appliquée à l'étape 3'. Au cours de cette procédure, le service de l'arc (v_2, v_3) est repoussé en fin de tournée en remplaçant l'arc $(v_8, v_9) = (a, c)$ par le chemin (a, b, f, c). Nous obtenons alors le circuit $T' = (v_0, d, a, b, e, b, e, c, a, b, f, c, v_0)$ dans lequel l'arc (a, b) est desservi lors de sa deuxième apparition, et un chemin inutile $P' = (v_0, d, a, b, e, b, e)$ dans T'. A l'étape 4, ce chemin est remplacé par le plus court chemin dans G reliant v_0 à e: (v_0, e). Nous obtenons alors la tournée $(v_0, e, c, a, b, f, c, v_0)$. Cette tournée, représentée dans la figure 2.31, ne peut plus être raccourcie et sa longueur est de 15.

Afin de déterminer si une des adaptations de RACCOURCIR au cas orienté que nous venons de décrire est plus efficace que l'autre, nous allons les appliquer sur une série de 180 circuits. Ces circuits sont générés de la manière suivante:

> 180 instances du PTAC orienté sont générées aléatoirement en utilisant un procédé identique à celui employé pour le cas non orienté et décrit à la section 2.3.4.
>
> Pour chacune des instances générées, un circuit est construit en résolvant un problème de postier chinois dans le graphe initial G à l'aide de PPC_ORIENTE (cf. section 1.2.1.2). Nous obtenons ainsi un circuit passant au moins une fois par chaque arc de G. Les arcs desservis dans ce circuit sont les arcs de R.

Les 180 instances du PTAC orienté sont divisées en 18 catégories qui dépendent du nombre de sommets du graphe initial G, de sa densité, ϵ, ainsi que de la proportion d'arcs à desservir dans G, ω. Pour chaque catégorie, 10 instances sont générées. Nous avons considéré des problèmes à 20 et 50 sommets. Nous avons choisi aléatoirement la valeur de ϵ dans les intervalles $[0.1, 0.3]$, $[0.4, 0.6]$ et $[0.7, 0.9]$ et celle de ω dans les intervalles $[0.1, 0.3]$, $[0.4, 0.6]$ et $[0.8, 1.0]$. Les différentes combinaisons possibles de ces trois paramètres définissent les

18 types de problèmes que nous avons considérés.

Le tableau de la page suivante contient les caractéristiques des 18 types de problèmes à partir desquels nous avons construit nos 180 circuits. Il contient également la réduction moyenne (sur les 10 circuits associés à chaque catégorie) en % de la longueur des circuits obtenue après application de RACCOR et de RACCOR_SIMPLE. Cette réduction moyenne sera notée R_{moy}. Les temps moyens de calcul nécessaires pour effectuer RACCOR_SIMPLE et RACCOR figurent aussi dans le tableau, de même que le nombre de fois (parmi les 10 instances de chaque catégorie), NB_{best}, qu'une procédure est meilleure que l'autre.

$\|V\|$	ϵ	ω	RACCOR_SIMPLE			RACCOR		
			R_{moy}	temps [s]	NB_{best}	R_{moy}	temps [s]	NB_{best}
20	[0.7,0.9]	[0.8,1.0]	3.01	0.40	0	4.02	0.30	8
		[0.4,0.6]	33.45	0.40	0	38.15	0.40	10
		[0.1,0.3]	67.20	0.20	0	71.07	0.10	10
	[0.4,0.6]	[0.8,1.0]	4.93	0.10	2	5.32	0.20	4
		[0.4,0.6]	36.88	0.10	1	39.38	0.20	9
		[0.1,0.3]	69.03	0.10	0	72.67	0.00	10
	[0.1,0.3]	[0.8,1.0]	5.64	0.00	2	6.00	0.10	3
		[0.4,0.6]	32.68	0.00	2	34.45	0.20	6
		[0.1,0.3]	66.72	0.10	0	68.43	0.00	9
50	[0.7,0.9]	[0.8,1.0]	2.21	18.00	2	2.70	15.20	6
		[0.4,0.6]	36.28	22.20	0	41.16	14.00	10
		[0.1,0.3]	69.89	9.50	0	74.45	5.10	10
	[0.4,0.6]	[0.8,1.0]	3.04	7.30	0	3.94	6.70	7
		[0.4,0.6]	32.38	9.30	0	36.23	7.10	10
		[0.1,0.3]	69.60	4.60	0	73.67	2.30	10
	[0.1,0.3]	[0.8,1.0]	3.69	2.10	1	3.96	1.70	5
		[0.4,0.6]	33.61	1.60	1	35.87	1.70	9
		[0.1,0.3]	68.62	0.90	0	71.97	0.40	10
Moyenne sur les 180 instances			35.49	4.27		37.97	3.09	
Total sur les 180 instances					11			146

TAB. 2.7 -

Comparaison des procédures RACCOR_SIMPLE et RACCOR.

Les résultats qui figurent dans le tableau 2.7 montrent clairement que la procédure RACCOR est plus efficace que RACCOR_SIMPLE. En effet, sur les 180 instances traitées, la procédure RACCOR bat RACCOR_SIMPLE 146 fois alors qu'elle n'est battue par cette dernière que 11 fois. Elle est de plus, en moyenne, légèrement plus rapide. Nous pouvons également remarquer que RACCOR se comporte mieux que RACCOR_SIMPLE que ce soit sur des circuits contenant de nombreux clients ou, au contraire, sur des circuits dans lesquels seule une faible quantité d'arcs est desservie. En particulier, lorsque la densité des clients est la plus faible ($\omega \in [0.1, 0.3]$), RACCOR est toujours meilleure que RACCOR_SIMPLE.

Lorsqu'il s'agira, par la suite, de raccourcir des circuits, nous utiliserons exclusivement la procédure RACCOR.

2.4.2.2 Adaptation de la procédure POST-OPT au cas orienté

La procédure de post-optimisation POST-OPT (cf. 2.3.1) contribue de façon non négligeable aux bonnes performances de la méthode tabou que nous avons développée pour le

cas non orienté du PTAC. Son adaptation au cas orienté nous semble donc utile.
Comme dans le cas de l'adaptation de RACCOURCIR, certaines étapes de POST-OPT ne nécessitent que très peu ou pas de modifications. L'étape 2 exceptée, toutes les autres étapes ne subissent que des modifications mineures. Nous ne mentionnerons pas ces modifications explicitement, le lecteur pourra toujours comparer la description de POST-OPT se trouvant à la section 2.3.1 avec celle de son adaptation au cas orienté qui va suivre.
Rappelons qu'au cours des étapes 0 et 1, un cycle T' est construit en mettant bout à bout l'ensemble des tournées d'une solution admissible s. Au cours de l'étape 2, le sens de parcours de certains tronçons de T' est inversé. Le but de ces inversions est de modifier l'ordre dans lequel les arcs à desservir apparaissent dans T'. De telles inversions ne sont plus possibles dans le cas orienté. Il nous faut donc utiliser une autre technique afin de modifier l'ordre d'apparition des clients dans T'. Au lieu d'inverser le sens de parcours de tronçons, nous allons permuter deux tronçons Tr_1 et Tr_2 de T. Pour que cela soit possible, il faut que le sommet initial de Tr_1 soit identique à celui de Tr_2 et que le sommet terminal de Tr_1 soit le même que celui de Tr_2.
La procédure PERMUTER_TRONCONS que nous allons décrire est basée sur ce principe.

Algorithme: PERMUTER_TRONCONS

ENTREE: Un circuit $T = (v_0, v_{t_1}, v_{t_2}, ..., v_{t_r}, v_0)$ dans $G = (V, A)$.

SORTIE: Un circuit T' composé des mêmes arcs que T.

Etape 1. Poser $T' := T$.
Choisir deux sommets v_{t_i} et v_{t_j} de T (on suppose que $i \leq j$).
En partant du sommet v_{t_j}, déterminer le premier sommet $v_{t_q} (j \leq q \leq r)$ tel que $v_{t_q} = v_{t_i}$. Si un tel sommet n'a pas pu être trouvé, STOP.
En partant du sommet v_{t_q}, déterminer le dernier sommet $v_{t_k} (q \leq k \leq r)$ tel que $v_{t_k} = v_{t_j}$. Si un tel sommet n'a pas pu être trouvé, STOP.

Etape 2. Soit $P_1 = (v_0, v_{t_1}, ..., v_{t_i})$ le chemin reliant v_0 à v_{t_i} dans T.
Soit $P_2 = (v_{t_i}, v_{t_{i+1}}, ..., v_{t_j})$ le chemin reliant v_{t_i} à v_{t_j} dans T.
Soit $P_3 = (v_{t_{j+1}}, ..., v_{t_q})$ le chemin reliant v_{t_j} à v_{t_q} dans T.
Soit $P_4 = (v_{t_q}, v_{t_{q+1}}, ..., v_{t_k})$ le chemin reliant v_{t_q} à v_{t_k} dans T.
Soit $P_5 = (v_{t_k}, ..., v_0)$ le chemin reliant $v_{t_{k+1}}$ à v_0 dans T.
Poser $T' := P_1 + P_4 + P_3 + P_2 + P_5$.

L'exemple qui va suivre illustre le fonctionnement de PERMUTER_TRONCONS. La tournée initiale est représentée dans la figure 2.32. Aucune distinction n'est faite entre les arcs desservis et les autres. Les tronçons en traitillés correspondent aux chemins P_2 et P_4 qui sont permutés. Les sommets v_{t_i} et v_{t_j} que nous avons choisis sont les sommets $v_{t_1} = a$ et $v_{t_3} = c$. En partant du sommet v_{t_3}, le premier sommet égal à v_{t_1} est le sommet $v_{t_q} = v_{t_6}$. En partant de v_{t_6}, le dernier sommet identique à v_{t_3} est le sommet $v_{t_{11}}$.

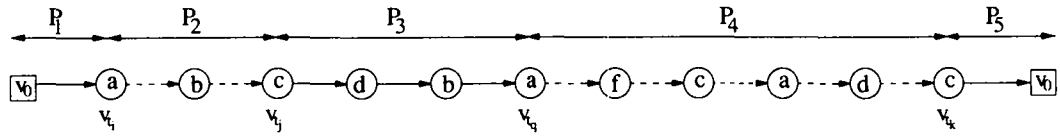


FIG. 2.32 -

Tournée T avant permutation.

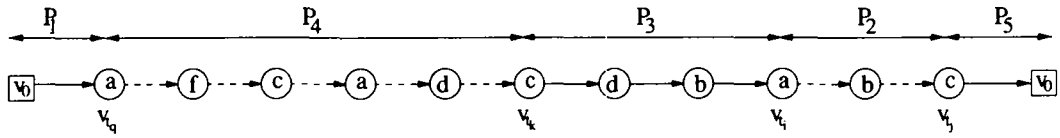


FIG. 2.33 -

Tournée T' obtenue après PERMUTER_TRONCONS.

La procédure PERMUTER_TRONCONS est utilisée dans l'adaptation de POST-OPT au cas orienté. Cette adaptation sera appelée POST-OPT_ORIENTE. Elle est décrite ci-dessous.

Algorithme: POST-OPT_ORIENTE

ENTREE: Une solution s (admissible ou non) du PTAC dans $G = (V, A)$.

SORTIE: Une solution $\bar{s}$ de valeur au pire égale à celle de s.

Etape 0. Poser $\bar{s} := s$ et $\gamma = 0$.

Etape 1. Mettre bout à bout chacune des tournées de s. Soit T la tournée ainsi obtenue. T dessert l'ensemble des arcs de R.
Déterminer une tournée T'' desservant les arcs de R en appliquant RACCOR à T.

Etape 2. Appliquer PERMUTER_TRONCONS à la tournée T''.
Soit T' la tournée fournie par PERMUTER_TRONCONS.

Etape 3. Construire une solution admissible s' en appliquant une adaptation au cas orienté de la procédure DECOUPER_TOURNEE (voir 1.3.2.1) sur T'.
Appliquer RACCOR sur chacune des tournées de s'.
Si la valeur de s' est inférieure à celle de s poser $\bar{s} := s'$, STOP.

Etape 4. Si $\gamma = 100$ STOP. Sinon poser $\gamma := \gamma + 1$.

Etape 5. Retourner à l'étape 2.

2.4.2.3 Adaptation d'autres procédures au cas orienté

L'adaptation d'AJOUTER_CLIENT ne nécessite que de légères modifications par rapport à la version décrite pour le cas non orienté à la section 1.2.2.5.
Ces modifications interviennent lorsqu'un client (v_i, v_j) est ajouté à une tournée T contenant déjà le sommet v_i mais ne contenant pas l'arc (v_i, v_j). Le circuit $(v_i, v_j) + SP_{ji}$ est alors ajouté à T.

Algorithme: AJOUTER_CLIENT_ORIENTE

ENTREE: Un circuit T dans $G = (V, E)$ desservant un ensemble $R_T \subset R$ d'arcs et un arcs de R, $(v_i, v_j) \notin R_T$.

SORTIE: Un circuit T' desservant les arcs de $R_T \cup (v_i, v_j)$.

Etape 1. Si le sommet v_i n'est pas présent dans T, trouver le sommet $v_k \in T$ minimisant la valeur $sp_{ki} + sp_{jk}$ et ajouter à T le circuit $SP_{ki} + \{(v_i, v_j)\} + SP_{jk}$.
Sinon, si v_i se trouve dans T et que T ne contient pas l'arc (v_i, v_j), ajouter à T le circuit $(v_i, v_j) + SP_{ji}$.

Etape 2. Poser $R_T = R_T \cup (v_i, v_j)$ et appliquer RACCOR à T pour obtenir une tournée T' au moins aussi courte que T.

La procédure SUPPRIMER_CLIENT ne nécessite aucune adaptation, si ce n'est qu'il faut utiliser la procédure RACCOR au lieu de RACCOURCIR.
Pour adapter la procédure SUPPRIMER&AJOUTER_CLIENT, il suffit de remplacer AJOUTER_CLIENT par AJOUTER_CLIENT_ORIENTE.

2.4.3 Méthode tabou appliquée au PTAC orienté

La méthode tabou que nous allons employer pour le PTAC orienté est une simple adaptation de CARPET au cas orienté. Les procédures décrites à la section 2.4.2 remplacent celles utilisées dans le cas non orienté. L'espace des solutions, la fonction objectif, le voisinage, la restriction sur la taille du voisinage, la liste tabou, le critère d'aspiration et le critère d'arrêt sont définis de la même manière que dans CARPET. Seule la manière de construire la solution initiale diffère légèrement. Nous appellerons CARPET_OR l'adaptation de CARPET au cas orienté.

2.4.3.1 La solution initiale

La solution initiale est construite à l'aide d'une méthode à deux phases basée sur une stratégie "Tournée & Groupes" (cf. 1.3.2.1). Une tournée T contenant l'ensemble des clients est obtenue en résolvant un problème de postier rural dans le graphe initial G. La résolution du PPR dans G se fait en utilisant l'algorithme S&C_PPR_ORIENTE décrit à la section 1.2.2.2. La tournée T est raccourcie puis découpée en tournées admissibles à l'aide de DECOUPER_TOURNEE (cf. 1.3.2.1). Chacune de ces tournées admissibles est

raccourcie à l'aide de RACCOR. Nous obtenons ainsi la solution initiale s_0 de la méthode de descente.

2.4.4 Adaptation de DVV2 au cas orienté

Comme dans le cas de la méthode tabou, l'adaptation de DVV2 au cas orienté consiste principalement à remplacer les procédures utilisées dans le cas non orienté par celles décrites à la section 2.4.2. Seule exception, l'étape 2 de la procédure VOIS(k) qui consiste simplement dans le cas orienté à appliquer PERMUTER_TRONCONS sur la tournée T' (voir section 2.3.3.4). L'espace des solutions est l'ensemble des solutions admissibles (aucune violation des contraintes de capacité n'est autorisée). La taille du voisinage examiné à chaque itération d'une méthode de descente ne change pas, 2500 solutions sont évaluées à chaque itération. La solution initiale est la même que celle de CARPET_OR. Nous appellerons DVV2_OR l'adaptation de DVV2 au cas orienté.

2.4.5 Comparaisons

Nous allons comparer les performances des heuristiques décrites en 2.4.3 et en 2.4.4. Nous verrons si les comportements observés dans le cas non orienté se répètent dans le cas orienté. Ces tests numériques vont nous permettre également d'évaluer la qualité de la borne inférieure $L\vec{B}1$ décrite à la section 2.4.1.3. Pour le calcul de cette borne, nous avons remplacé la borne inférieure $\underline{m}$ sur le nombre de véhicules nécessaires par la borne inférieure LR développée par Martello et Toth [Mar90] pour le "Bin Packing Problem".

Les problèmes tests que nous avons traités sont ceux que nous avons déjà décrits en 2.4.2.1. Rappelons que ces problèmes sont divisés en 18 catégories et que chaque catégorie comporte 10 instances. La plus petite de ces 180 instances comprend 20 sommets, 42 arcs dont 13 sont à desservir. L'instance la plus importante possède 50 sommets, 2132 arcs dont 2026 sont des clients. Dans les tableaux qui suivent, E_{moyen}, et E_{max} représentent respectivement l'écart moyen et l'écart maximum (sur les 10 instances de chaque catégorie) entre la valeur de la solution obtenue par les algorithmes et celle de la borne $L\vec{B}1$. Ces écarts sont mesurés en %. Figurent également dans le tableau le temps moyen d'exécution des algorithmes ainsi que le nombre d'instances (parmi les 10 de chaque catégorie) qui sont résolues de façon optimale. Ce nombre est noté N_{opt}.

Le tableau 2.8 contient les résultats obtenus à l'aide de DVV2_OR et CARPET_OR. En comparant ces deux méthodes, les mêmes remarques que celles déjà énoncées dans le cas non orienté peuvent être faites. Les 2 algorithmes sont très proches l'un de l'autre si l'on considère l'écart moyen (sur les 180 instances) par rapport à $L\vec{B}1$, avec un léger avantage toutefois à RVV2_OR. En ce qui concerne le temps moyen de calcul, une très nette différence existe entre les deux méthodes, CARPET_OR nécessitant un temps d'exécution moyen beaucoup plus important que RVV2_OR. Cette grande différence apparaît principalement lorsque des instances de taille très importante sont traitées. Pour des problèmes de taille plus réduite, il arrive que CARPET_OR soit plus rapide que RVV2_OR. Cela s'explique par le fait que le nombre de solutions évaluées à chaque itération par CARPET_OR dépend en grande partie du nombre de clients à desservir, alors que dans le cas de RVV2_OR, ce nombre est fixe quel que soit le problème traité.

$\|V\|$	ϵ	ω	DVV2_OR				CARPET_OR			
			E_{moyen}	E_{max}	Temps [s]	N_{opt}	E_{moyen}	E_{max}	Temps [s]	N_{opt}
20	[0.7,0.9]	[0.8,1.0]	0.58	2.26	235.01	3	0.91	2.37	479.99	2
		[0.4,0.6]	0.99	3.09	146.88	3	1.69	5.39	189.77	2
		[0.1,3.0]	6.46	17.98	100.69	1	7.08	19.83	75.96	1
	[0.4,0.6]	[0.8,1.0]	1.36	5.61	132.306	5	1.39	3.88	204.19	1
		[0.4,0.6]	3.87	11.97	123.90	2	3.89	9.76	145.25	3
		[0.1,0.3]	12.06	45.30	64.84	1	12.12	45.30	47.74	1
	[0.1,0.3]	[0.8,1.0]	6.36	21.73	129.55	1	6.19	21.13	130.65	1
		[0.4,0.6]	6.52	10.00	64.69	1	7.10	12.43	55.91	0
		[0.1,0.3]	12.68	29.37	23.24	1	13.34	37.41	20.63	1
50	[0.7,0.9]	[0.8,1.0]	0.14	0.61	2141.00	2	0.18	0.80	31325.86	2
		[0.4,0.6]	0.40	1.61	1257.1	3	0.73	1.90	8241.40	0
		[0.1,0.3]	1.16	4.10	555.23	1	1.94	4.61	1917.07	0
	[0.4,0.6]	[0.8,1.0]	0.31	0.70	1324.75	3	0.56	1.36	10576.12	2
		[0.4,0.6]	0.84	2.08	933.04	0	1.26	2.12	3948.47	0
		[0.1,0.3]	1.88	4.50	377.23	0	2.13	4.07	1160.66	1
	[0.1,0.3]	[0.8,1.0]	1.69	3.35	637.96	1	1.66	3.23	2626.68	1
		[0.4,0.6]	2.67	6.72	364.50	2	3.09	5.96	954.39	1
		[0.1,0.3]	5.26	11.92	190.88	0	5.88	12.06	314.75	0
Moyenne sur les 180 instances			3.62		489.04		3.95		3467.53	
Total ou maximum				45.30		30		45.30		19

TAB. 2.8 -

Résultats de DVV2_OR et CARPET_OR.

Comparés aux écarts enregistrés par rapport à $\underline{F}$ dans le cas non orienté, les écarts par rapport à $\vec{LB}1$ sont plus importants. La proportion de solutions prouvées optimales est plus faible dans le cas orienté. Ces constatations ne nous permettent pas de nous faire une idée précise de la qualité de $\vec{LB}1$, dans la mesure où ce sont peut-être nos algorithmes qui se montrent moins performants dans le cas orienté.
Indépendamment des résultats que nous avons obtenus, nous pouvons cependant relever un défaut de la borne $\vec{LB}1$. Celle-ci ne tient pas véritablement compte de la répartition des demandes dans le réseau qui est traité. Ces dernières n'apparaissent dans le calcul de $\vec{LB}1$ que pour déterminer une borne inférieure sur le nombre de véhicules qu'il faut utiliser pour pouvoir servir tous les clients. Dans ce calcul, la façon dont sont réparties les demandes dans le réseau n'intervient pas. Dès lors, en permutant les demandes de chacun des clients, la valeur de la borne n'est pas modifiée. Cela, ainsi que le fait qu'une borne inférieure est utilisée pour estimer le nombre de véhicules nécessaires au service des clients, sont peut-être des raisons pour lesquelles, dans certains cas, des écarts très importants apparaissent entre la valeur de $\vec{LB}1$ et celle des solutions fournies par CARPET_OR et DVV2_OR.

2.5 Conclusion

Nous avons présenté dans ce chapitre de nouveaux outils utiles à la résolution du PTAC orienté et non orienté. Deux heuristiques ont été développées, l'une basée sur une méthode tabou, l'autre sur une méthode de descente à voisinage variable. Du point de vue de la qualité des solutions obtenues, ces heuristiques figurent parmi les meilleures méthodes disponibles actuellement. En ce qui concerne leur temps de calcul, nous manquons d'éléments de comparaison. Nous avons pu cependant mettre en évidence la supériorité, de ce point de vue-là, de notre adaptation de la méthode de descente à voisinage variable

sur celle de la méthode tabou. Nous avons également adapté au cas orienté une borne inférieure développée à l'origine pour le cas non orienté du PTAC.

Pour de futures recherches concernant la résolution du PTAC, les méthodes à voisinage variable semblent prometteuses. Celle que nous avons développée est très simple. Des techniques plus élaborées utilisant également de nouveaux types de voisinages pourront sans doute s'avérer plus efficaces. Le développement de bornes inférieures plus performantes pour le PTAC orienté, qui tiennent compte de la répartition des demandes dans le réseau, constitue également un terrain qui mériterait d'être exploré. Finalement, nous avons traité les cas orienté et non orienté du PTAC. Nous avons pu facilement adapter au cas orienté des procédures initialement développées pour le cas non orienté. Il serait intéressant de voir si ces procédures peuvent s'adapter aussi facilement au cas mixte du PTAC. A notre connaissance, ce cas du PTAC n'a jamais été traité dans la littérature, de très nombreux développements restent donc à faire dans ce domaine.

Chapitre 3

Une application des PTA: la confection d'horaires de bateaux

La Compagnie Générale de Navigation (CGN) gère une flotte composée de 16 bateaux. Cette flotte dessert 39 ports répartis tout autour du lac Léman. Les services effectués entre les ports sont réguliers et répondent soit à des besoins de service public, soit à des demandes de liaisons touristiques. La flotte de la CGN est constituée de divers types de bateaux. Ceux-ci se distinguent essentiellement par leur mode de propulsion. Les ports qui doivent être desservis ont des importances diverses qui peuvent dépendre de leur taille, leur situation géographique, leur attractivité touristique, etc. Actuellement, l'élaboration des horaires des bateaux se fait manuellement. La CGN souhaite pouvoir disposer d'un outil d'aide à la décision permettant de construire tout ou partie des horaires automatiquement.

Ce chapitre est principalement consacré à la modélisation de ce problème d'horaire sous la forme d'un problème de tournées sur les arcs avec contraintes de capacité. Nous présenterons également quelques approches que nous avons utilisées pour tenter de résoudre ce problème. Des travaux sont actuellement en cours visant à développer, sur la base de la modélisation décrite dans ce chapitre, un logiciel d'aide à la décision pour la confection des horaires de la CGN.

Une première partie de ce chapitre sera consacrée à la description détaillée du problème proposé par la CGN. Nous présenterons les différentes contraintes à prendre en compte. Dans un deuxième temps, nous proposerons une modélisation du problème en termes de PTAC. Nous décrirons ensuite des méthodes de résolution basées sur des algorithmes d'insertion. Finalement, une analyse des résultats obtenus sera effectuée.

3.1 Description du problème

3.1.1 Données du problème

3.1.1.1 Les bateaux

La CGN dispose aujourd'hui de 16 bateaux dont la capacité maximum varie entre 1500 et 150 passagers. Certains bateaux sont propulsés par des moteurs Diesel-électriques. Ils sont divisés en deux catégories: les bateaux à roues à aubes et les bateaux à hélices. D'autres embarcations sont mues à l'aide de machines à vapeur et sont toutes munies de roues à aubes. Les quatre bateaux à moteur Diesel-électriques possédant des roues à aubes vont progressivement être transformés en bateaux à vapeur. Ce processus de transformation est appelé "revaporisation". Il va s'étaler sur plusieurs années, immobilisant de ce fait successivement un des quatre bateaux durant plusieurs mois. Le processus de revaporisation s'inscrit dans le cadre d'un programme de sauvegarde du patrimoine lémanique. Les quatre bateaux concernés vont être modifiés de manière à revenir au mode de propulsion utilisé autrefois.

Suivant son type et sa fréquentation, un bateau est amené à effectuer une ou plusieurs fois par jour une vidange des réservoirs contenant les eaux usées. La durée des vidanges dépend du type de bateau. Seuls certains ports sont équipés des installations appropriées pour effectuer les vidanges.

Le tableau ci-dessous contient les caractéristiques des 16 bateaux de la CGN.

Nom du bateau	Capacité maximale (places debout comprises)	Capacité places assises et couvertes	Mode de propulsion et type de bateau	Année de mise en service	Temps de vidange
Simplon	1500	450	à vapeur, à roues à aubes	1920	20 min
Lausanne	1500	500	à moteur, à hélices	1991	20 min
Helvétie	1400	430	à moteur, à roues à aubes	1926	20 min
La Suisse	1200	450	à vapeur, à roues à aubes	1910	20 min
Rhône	850	270	à vapeur, à roues à aubes	1927	20 min
Léman	850	537	à moteur, à hélices	1990	15 min
Montreux	800	220	à moteur, à roues à aubes	1904	20 min
Italie	800	220	à moteur, à roues à aubes	1908	20 min
Savoie	800	270	à vapeur, à roues à aubes	1914	20 min
Vevey	750	200	à moteur, à roues à aubes	1907	20 min
Henry-Dunant	700	190	à moteur, à hélices	1963	15 min
Général-Guisan	700	190	à moteur, à hélices	1964	15 min
Chablais	560	130	à moteur, à hélices	1974	20 min
Ville-de-Genève	560	200	à moteur, à hélices	1978	15 min
Grèbe	170	45	vedette à moteur, à hélices	1961	15 min
Col-Vert	150	45	vedette à moteur, à hélices	1960	15 min

TAB. 3.1 -

Caractéristiques des bateaux de la CGN.

Parmi les 16 bateaux formant la flotte de la CGN, certains d'entre eux ne seront pas pris en compte lors de la construction des horaires. Il s'agit du *Montreux* qui est en cours de revaporisation, et des deux vedettes *Grèbe* et *Col-Vert*. Les services assurés par les vedettes sont des services locaux effectués dans les ports situés aux environs immédiats

de Genève. Nous n'allons pas nous en occuper, nous travaillerons uniquement avec les 13 bateaux restants.

3.1.1.2 Le lac et les ports

Actuellement, 39 ports sont desservis par la CGN. Ces ports sont répartis sur la rive suisse et la rive française du lac Léman. Certains ports possèdent des caractéristiques particulières qui les distinguent des autres. Le port de Genève, par exemple, est constitué de 4 débarcadères (Mont-Blanc, Jardin Anglais, Eaux-Vives et Pâquis) entre lesquels les bateaux effectuent des liaisons. Il sera cependant considéré comme un seul port (nous reparlerons de ce port dans la partie consacrée à la modélisation). Nous avons déjà mentionné le fait que seuls quelques ports sont équipés du matériel nécessaire aux vidanges des bateaux. Il s'agit des ports de Lausanne, Genève et du Bouveret. Les bateaux ne peuvent stationner la nuit que dans 4 ports et cela en nombre généralement limité: Genève (4 bateaux au maximum), Lausanne (nombre illimité de bateaux), le Bouveret (1 bateau) et Yvoire (1 bateau). Une dernière caractéristique des ports est le nombre de points d'amarrage dont ils disposent. Seuls les ports de Lausanne, Thonon et Yvoire en possèdent plus d'un et permettent ainsi le transbordement direct de passagers d'un bateau à un autre. La figure 3.1 représente le lac Léman ainsi que les ports desservis par la CGN.

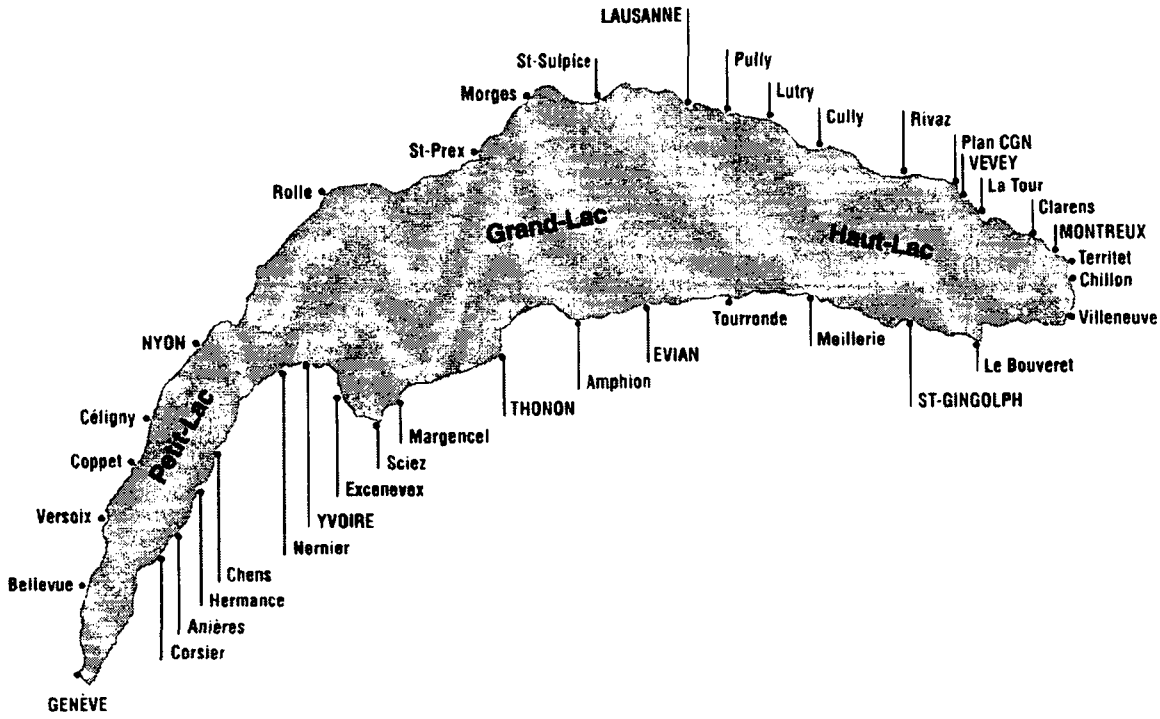


FIG. 3.1 -

Le lac Léman et les ports desservis par la CGN.

3.1.1.3 Les horaires actuels

La CGN utilise 4 horaires différents au cours d'une année (un par saison). L'horaire assurant le plus grand nombre de liaisons est celui d'été. Il débute fin mai et se termine fin septembre. Les deux tiers des passagers transportés le sont au cours de cette période. Onze bateaux assurent les services réguliers de l'horaire d'été actuellement.
Nous allons travailler uniquement sur l'horaire d'été. Il s'agit de la période la plus difficile à traiter.

3.1.2 Les objectifs de la CGN

Depuis 1991, un recul de la fréquentation des bateaux a été enregistré par la CGN. Ce recul tend à se stabiliser depuis 1995 avec un nombre de passagers transportés annuellement d'environ 1'330'000. Une légère augmentation a même été observée en 1997. On reste cependant loin du chiffre de 1991: environ 1'719'000 passagers transportés. Afin de revenir à un niveau proche de celui de 1991, la CGN cherche à rendre ses horaires plus attractifs.

Les horaires doivent satisfaire des contraintes de service public. Il s'agit essentiellement de transporter les travailleurs frontaliers entre Lausanne et Evian et entre Nyon et Yvoire. Il s'agit également d'assurer un "service minimum" dans les 39 ports actuellement desservis. Cela se traduit par un nombre minimum de passages qu'il faut effectuer dans chaque port. Les horaires se doivent aussi d'être attractifs pour une clientèle touristique et locale variée. La possibilité doit être offerte de faire des circuits de plus ou moins longue durée dans les différentes portions du lac.

La CGN dispose également d'un "service charter". Ce service consiste à louer (à des entreprises, à des associations, etc.) des bateaux qui ne sont pas affectés aux lignes régulières. Ces bateaux accomplissent alors des courses spéciales. Un bateau affecté au service charter dégage plus de bénéfices que s'il était en service sur les lignes régulières. Un des buts de la CGN est de tenter de réduire le nombre d'unités circulant sur les lignes régulières afin de pouvoir utiliser les bateaux ainsi libérés pour le service charter. Cette diminution du nombre de bateaux affectés aux lignes régulières, si elle est possible, ne doit cependant pas se faire au détriment de la qualité de l'horaire.

En résumé, les objectifs de la CGN sont, d'une part de pouvoir disposer d'un horaire attractif correspondant au rôle de service public que joue la compagnie ainsi qu'à sa vocation touristique, et d'autre part d'affecter au mieux les différents bateaux dont elle dispose afin d'augmenter ses bénéfices.

3.1.3 Les contraintes

Nous allons énumérer les différents types de contraintes intervenant lors de la confection des horaires de la CGN. Nous mentionnerons tout d'abord l'ensemble des contraintes, puis nous préciserons celles que nous avons prises en compte et celles que nous avons négligées.

3.1.3.1 Contraintes liées aux bateaux

Capacité maximum des bateaux:

Comme nous l'avons déjà mentionné, la capacité des bateaux n'est pas la même pour toutes les unités de la flotte (voir tableau 3.1). Les 13 bateaux disponibles pour la conception des horaires ont des capacités qui vont de 560 passagers (pour le bateau le plus petit) à 1500 passagers (pour les plus grosses embarcations). Il peut arriver que sur certains trajets, à certaines périodes, le nombre de passagers soit supérieur à 600 (636 passagers sur la course Nyon-Nernier le 30 août 1998). Il faut dès lors faire en sorte que des bateaux disposant de suffisamment de places assurent ces liaisons.

Les vidanges:

Une à deux fois par jour, les bateaux doivent vidanger leurs réservoirs d'eaux usées. Ils peuvent le faire uniquement s'ils se trouvent aux ports de Genève, Lausanne ou au Bouveret.

Le stationnement de nuit:

Les bateaux ne peuvent passer la nuit qu'aux ports de Genève (4 bateaux), Lausanne (nombre illimité de bateaux), Yvoire (1 bateau) et Bouveret (1 bateau). Nous avons supposé que chaque bateau revenait à son port d'origine. Cela n'est pas forcément le cas en réalité, un bateau stationnant à Genève peut par exemple terminer sa course à Lausanne. Les 4 ports pouvant accueillir les bateaux la nuit seront parfois appelés dépôts.

Durée de service:

Le temps de travail du personnel navigant de la CGN est limité à 13 heures par jour. Un bateau ne peut donc pas être en service plus de 13 heures à moins de changer d'équipage.
Les bateaux assurant un service régulier peuvent quitter un port au plus tôt à 5h00 et doivent terminer leur service (i.e. arriver à un port) au plus tard à 23h30.

3.1.3.2 Contraintes liées aux ports

En fonction de la catégorie à laquelle ils appartiennent, les ports doivent être desservis un certain nombre de fois par jour "dans les deux sens" (i.e. dans les deux directions le long de la rive du lac). Nous parlerons de sens positif lorsqu'un bateau circule le long de la rive du lac dans le sens des aiguilles d'une montre et de sens négatif dans le cas contraire (l'orientation du lac est celle de la figure 3.1). Les différentes catégories sont numérotées de 1 à 5, selon l'importance décroissante des ports. Le tableau 3.2 contient la liste des ports, la catégorie (ou type) à laquelle ils appartiennent et le nombre de passages journaliers minimum devant y être effectués. Il contient également les appellations officielles des ports. Les noms que nous utiliserons pour les désigner dans ce chapitre sont écrits en gras. Les cas spéciaux du Bouveret et de St-Gingolph seront traités dans la partie consacrée à la modélisation du problème. Au port de Genève, les sens positif et négatif de circulation ne sont pas distingués. Cela est dû à la situation géographique de Genève qui se trouve à une extrémité du lac. Ce port doit simplement être visité au moins 4 fois par jour.

Numéro du port	Nom du port	Type	Nombre de passages par jour	
			sens positif	sens négatif
0	**Lausanne**-Ouchy	1	4×	4×
1	**Pully**	2	3×	3×
2	**Lutry**	3	3×	3×
3	**Cully**	3	3×	3×
4	**Rivaz**/St-Saphorin	4	2×	2×
5	Vevey-**Plan CGN**	4	2×	2×
6	**Vevey**-Marché	1	4×	4×
7	Vevey-**La Tour**	3	3×	3×
8	**Clarens**	3	3×	3×
9	**Montreux**	1	4×	4×
10	**Territet**	3	3×	3×
11	**Chillon**	2	3×	3×
12	**Villeneuve**	2	3×	3×
13	**Le Bouveret**	1	cas spécial	4×
14	**St-Gingolph**	2	4×	cas spécial
15	**Meillerie**	4	2×	2×
16	**Tourronde**	4	2×	2×
17	**Evian**-les-Bains	1	4×	4×
18	**Amphion**-les-Bains	4	2×	2×
19	**Thonon**-les-Bains	1	4×	4×
20	**Margencel**/Anthy/Séchex	4	2×	2×
21	**Sciez**	5	1×	1×
22	**Excenevex**	5	1×	1×
23	**Yvoire**	1	4×	4×
24	**Nernier**	3	3×	3×
25	**Chens**-sur-Léman/Tougues	4	2×	2×
26	**Hermance**	3	3×	3×
27	**Anières**	4	2×	2×
28	**Corsier**	4	2×	2×
29	**Genève**	1	4× (pas de direction)	
30	**Bellevue**	4	2×	2×
31	**Versoix**	3	3×	3×
32	**Coppet**	3	3×	3×
33	**Céligny**	4	2×	2×
34	**Nyon**	1	4×	4×
35	**Rolle**	2	3×	3×
36	**St-Prex**	3	3×	3×
37	**Morges**	2	3×	3×
38	**St-Sulpice**	3	3×	3×

TAB. 3.2 -

Les ports desservis par la CGN.

La classification des ports en différentes catégories a été effectuée par M. Aegler, fondé de pouvoir et chef du service commercial de la CGN. Elle correspond à l'importance accordée aux différents ports par la CGN. Elle peut s'interpréter de la manière suivante:

Type 1: Ports importants, normalement tous les bateaux s'y arrêtent, doivent être visités quotidiennement au moins 4 fois dans chaque sens.

Type 2: Ports de moyenne importance, presque tous les bateaux s'y arrêtent, trois visites journalières dans les deux sens doivent être effectuées au minimum.

Type 3: Ports secondaires, seuls certains bateaux s'y arrêtent, au moins trois passages quotidiens dans les deux directions sont nécessaires.

Type 4: Petits ports, peu de bateaux s'y arrêtent, deux visites par jour au minimum doivent être faites dans chacun des sens positif et négatif.

Type 5: Ports mineurs de la côte française, les bateaux ne s'y arrêtent que très rarement, une visite dans les deux sens au minimum doit se faire quotidiennement.

Afin de répartir dans la journée les différents passages dans un même port, nous avons introduit des tranches horaires (ou fenêtres de temps) à l'intérieur desquelles les ports doivent être visités. Elles sont présentées dans le tableau 3.3.

4 × par jour	3 × par jour	2 × par jour	1 × par jour
9h00 - 13h00	9h00 - 14h00	9h00 - 15h00	9h00 - 19h30
11h00 - 15h00	11h00 - 17h00	12h00 - 19h30	
13h00 - 17h00	14h00 - 19h30		
15h00 - 19h30			

TAB. 3.3 –

Heures de passages dans les ports.

Les fenêtres de temps ci-dessus sont valables pour les passages dans les ports suivant le sens positif ainsi que pour ceux qui sont effectués dans le sens négatif. Nous avons également défini d'autres tranches horaires, mais sauf mention contraire, ce sont celles du tableau 3.3 que nous utiliserons.

Exemple:

Considérons le port de Pully. Ce port doit être visité 3 fois par jour dans chacun des deux sens positif et négatif. Le premier passage dans le sens positif ainsi que le premier passage dans le sens négatif devront avoir lieu entre 9h00 et 14h00. Entre 11h00 et 17h00, deux nouveaux passages (un dans chaque sens) devront être effectués. Finalement, le dernier passage dans chacune des directions doit être accompli entre 14h00 et 19h30.

3.1.3.3 Trajets imposés

En plus des visites qui doivent être faites dans chaque port, un certain nombre de trajets doivent obligatoirement être effectués. Cela concerne principalement des liaisons transfrontalières entre la France et la Suisse. A chaque trajet imposé est associée une fenêtre de temps sur l'heure à laquelle il doit débuter. Ces fenêtres de temps servent à imposer un certain nombre de liaisons à des périodes clefs de la journée: tôt le matin et en début de soirée pour le transport des frontaliers, un peu avant et un peu après midi, etc. Deux types de trajets peuvent être imposés, les trajets directs (liaisons directes entre deux ports) et

les trajets indirects (liaisons entre deux ports passant par des ports intermédiaires).

Les tableaux ci-dessous contiennent la description de l'ensemble des trajets directs imposés.

13 traversées Lausanne ⟶ Evian	
5h00 - 6h00	1×
6h00 - 7h00	1×
10h30 - 11h30	1×
9h00 - 14h00	2×
11h00 - 16h00	2×
17h00 - 18h00	1×
18h00 - 19h30	1×
9h00 - 19h30	4×

11 traversées Evian ⟶ Lausanne	
5h00 - 6h00	1×
6h00 - 7h00	1×
9h00 - 14h00	2×
11h00 - 16h00	2×
17h00 - 18h00	1×
18h00 - 19h30	1×
9h00 - 19h30	3×

10 traversées Nyon ⟶ Yvoire	
9h00 - 14h00	2×
10h30 - 11h30	1×
17h00 - 19h30	1×
9h00 - 19h30	6×

10 traversées Yvoire ⟶ Nyon	
9h00 - 14h00	2×
10h30 - 11h30	1×
17h00 - 19h30	1×
9h00 - 19h30	6×

4 traversées Montreux ⟶ Le Bouveret	
9h00 - 13h00	1×
11h00 - 15h00	1×
13h00 - 17h00	1×
15h00 - 19h30	1×

4 traversées Le Bouveret ⟶ Montreux	
9h00 - 13h00	1×
11h00 - 15h00	1×
13h00 - 17h00	1×
15h00 - 19h30	1×

4 liaisons directes St-Gingolph ⟶ Le Bouveret	
9h00 - 13h00	1×
11h00 - 15h00	1×
13h00 - 17h00	1×
15h00 - 19h30	1×

4 liaisons directes Le Bouveret ⟶ St-Gingolph	
9h00 - 13h00	1×
11h00 - 15h00	1×
13h00 - 17h00	1×
15h00 - 19h30	1×

TAB. 3.4 –
Trajets directs imposés.

Deux aller-retours, Genève - Le Bouveret - Genève et Le Bouveret - Genève - Le Bouveret doivent être réalisés tous les jours en plus des liaisons directes décrites ci-dessus. Ces trajets sont appelés transversales. Ils relient les deux extrémités du lac. Le départ de Genève ou du Bouveret peut se faire au plus tôt à 5h00. L'autre extrémité du lac doit être atteinte entre 11h00 et 15h00. Le retour au point de départ doit être effectué au plus tard à 23h30. L'itinéraire suivi par les bateaux qui accomplissent les 2 transversales n'est pas fixé. Les services assurés au cours des transversales (visites obligatoires de ports ou liaisons directes imposées) sont pris en compte. Deux bateaux sont affectés aux transversales, l'un passe la nuit au Bouveret, l'autre à Genève.

3.1.3.4 Contraintes prises en compte et contraintes négligées

Parmi les contraintes liées aux bateaux, nous avons laissé tomber les contraintes de capacité. Plusieurs raisons motivent ce choix. La CGN dispose de suffisamment de bateaux de grande capacité. La fréquentation des bateaux sur un trajet est très difficile à évaluer. La météo par exemple influence de manière très importante le nombre de passagers convoyés par la CGN. Nous allons donc supposer que tous les bateaux disposent d'une capacité suffisante.

Nous avons également négligé la contrainte des vidanges. Le nombre de vidanges devant être effectuées sur un bateau dépend en partie de sa fréquentation: plus il y a de monde, plus la quantité des eaux usées générée est importante. Ce nombre est donc difficile à évaluer. Nous essayerons toutefois de voir, une fois les tournées construites, s'il est possible d'y insérer un arrêt pour effectuer les vidanges.
Finalement, les seules contraintes liées aux bateaux que nous allons prendre en compte sont celles qui concernent la durée de service et le stationnement de nuit. Cela revient à supposer que nous disposons d'une flotte homogène et que tous les bateaux sont interchangeables. Nous allons donc nous atteler à la construction de tournées et laisser le soin à l'utilisateur de régler le problème de l'affectation des bateaux aux tournées.

Les visites obligatoires des ports dans les deux directions que nous avons définies ainsi que les trajets imposés sont les contraintes avec lesquelles nous allons travailler. Nous nous efforcerons également de respecter les différentes fenêtres de temps que nous avons introduites et qui sont associées aux deux types de contraintes que nous venons de mentionner.

Quelques contraintes particulières que nous avons négligées

Nous n'avons que brièvement mentionné le problème des correspondances entre les bateaux en signalant que seuls quelques ports étaient équipés de plusieurs points d'amarrage (Lausanne, Thonon et Yvoire). D'autres types de correspondances peuvent également exister notamment avec d'autres modes de transport, trains ou bus par exemple. Même si le fait de pouvoir établir des correspondances permet de renforcer l'attractivité d'un horaire, ce type de contraintes revêt une importance secondaire par rapport à celles concernant les trajets et les visites de ports imposés. Nous n'allons donc pas en tenir compte.

En raison de phénomènes de basses eaux, certains bateaux ne peuvent pas visiter tous les ports. Ces phénomènes n'interviennent qu'à certaines périodes de l'année. Nous laisserons le soin à l'utilisateur de modifier lui-même l'affectation des bateaux aux tournées quand de tels phénomènes se produiront. Nous ne disposons d'ailleurs d'aucune information précise concernant ce problème, nous savons simplement qu'il peut exister.

Finalement, nous ne nous occuperons pas non plus des questions concernant d'éventuelles rotations des équipages. Ces dernières peuvent permettre d'allonger le temps de service journalier des bateaux. En revanche elles soulèvent de nombreuses questions qui compliquent passablement le problème: dans quels ports effectuer ces rotations, quand, sur quels bateaux, etc. Nous n'avons donc pas tenu compte de la possibilité d'effectuer des rotations entre les équipages et nous avons fixé la durée maximum de service d'un bateau à 13 heures. Cette durée correspond au temps de travail journalier maximum du personnel naviguant.

3.2 Modélisation

Le fait d'avoir, parmi les contraintes à satisfaire, des liaisons à accomplir obligatoirement, nous a conduit à envisager une modélisation de notre problème en termes de PTA (Problème de Tournées sur les Arcs). Pour cela, un réseau doit tout d'abord être construit, puis le problème qu'il s'agira de résoudre dans ce réseau devra être clairement défini.

3.2.1 Le réseau

3.2.1.1 Généralités

La façon la plus naturelle de construire un réseau qui correspond à notre problème consiste à représenter les ports à l'aide des sommets du réseau. Nous devons être capables de distinguer le sens de parcours (positif ou négatif) des bateaux circulant sur le lac, nous allons donc travailler avec un réseau orienté. Les liaisons entre les ports seront représentées par les arcs du réseau. Des fenêtres de temps seront associées à certains arcs pour indiquer que le début d'un trajet doit s'effectuer dans une certaine tranche horaire. A ce stade, nous n'avons qu'une idée très vague de ce que sera notre réseau. Il nous faut encore définir quelles sont les liaisons (les arcs) qui vont le constituer. En effet, si en théorie toutes les liaisons sont possibles, certaines d'entre elles sont complètement irréalistes et ne doivent donc pas figurer dans le réseau. Il faut également prendre en compte dans la modélisation l'importance des différents ports et faire en sorte, là aussi, que certains comportements théoriquement possibles mais inacceptables en pratique, puissent être évités.

3.2.1.2 Les liaisons entre les ports

Toutes les liaisons entre les 39 ports desservis par la CGN ne sont pas réalistes. Un trajet direct reliant par exemple le port de Meillerie à celui de Bellevue (voir figure 3.1) n'a aucun sens dans la pratique. En effet, cela reviendrait à traverser la quasi totalité du lac pour relier entre eux, de manière directe, deux petits ports (ports de type 4). Des restrictions au niveau des liaisons directes ont donc été effectuées et cela avec l'aide de la CGN. Finalement, les seules liaisons directes que nous avons conservées sont celles qui relient entre eux des ports "proches" les uns des autres. Cette notion de proximité est élargie lorsque les ports sont importants. Les ports les moins importants seront reliés exclusivement à des ports très proches situés sur la même rive du lac. Pour les ports de plus grande importance, les traversées du lac sont autorisées, mais là encore, pour autant que la liaison effectuée ne soit pas trop longue (il n'y a par exemple pas de trajet direct Lausanne - Genève). Une liste détaillée des liaisons directes prises en compte est disponible dans [Bar99]. Elle contient 186 trajets (93 aller-retours).

3.2.1.3 Les temps de parcours

A chaque arc du réseau est associé un temps de parcours. Ces temps de parcours sont indépendants des bateaux. Nous supposons que quel que soit le bateau utilisé, sa vitesse est la même. Nous avons supposé également que les temps de parcours étaient les mêmes dans les deux sens. Un trajet Lausanne - Evian a la même durée qu'un trajet Evian - Lausanne. Cela n'est pas forcément vrai en pratique où les temps de parcours peuvent dépendre des courants existant dans le lac ainsi que de la direction du vent. Ces facteurs sont extrêmement difficiles à prendre en compte aussi avons nous supposé que les temps

de parcours étaient symétriques. Lorsqu'ils ne le sont pas, les techniques de résolution que nous avons utilisées fonctionnent également.

Les temps de parcours prennent en compte la durée du trajet et les temps d'accostage, de débarquement et d'embarquement. Le temps de parcours entre deux ports correspond donc au temps écoulé entre le départ du premier port et le départ du second. Les temps de parcours ont été calculés en minutes en se basant sur l'horaire actuel, et arrondis de manière à ce qu'ils soient des multiples de 5. Les temps des trajets ne figurant pas dans l'horaire actuel mais apparaissant dans notre réseau ont été obtenus de manière approximative en se basant sur les distances séparant les ports. Ces distances nous ont été communiquées par la CGN.

Il n'y a pas de longueur associée aux arcs du réseau. Lorsque nous parlerons par la suite de plus courts chemins, il s'agira de plus courts chemins obtenus en se basant sur les temps de parcours de chaque arc, c'est-à-dire de chemins les plus rapides.

3.2.1.4 Les arcs obligatoires et facultatifs

Lorsqu'une ou plusieurs liaisons directes doivent être accomplies entre deux mêmes ports (trajets imposés), un arc est associé à chacune d'elles et il est ajouté au réseau. Il faut par exemple effectuer 13 trajets Lausanne - Evian. Treize arcs sont alors ajoutés au réseau. Ces arcs ont tous les mêmes extrémités. Ils relient entre eux deux sommets associés l'un au port de Lausanne, l'autre au port d'Evian. Les arcs associés à des trajets imposés (pour la liste de ces trajets voir la section 3.1.3.3) seront considérés comme des arcs obligatoires, c'est-à-dire des arcs devant être desservis au moins une fois. Les autres arcs seront des arcs facultatifs par lesquels il est possible de passer, mais sans obligation aucune de le faire. A chaque arc obligatoire est associée une fenêtre de temps. Celle-ci définit une plage horaire à l'intérieur de laquelle il faut que débute le trajet représenté par l'arc. La fenêtre de temps est composée d'une borne inférieure (heure du début du trajet au plus tôt) et d'une borne supérieure (heure du début du trajet au plus tard). Ces bornes sont définies en minutes écoulées depuis 0h00. Ainsi, si un trajet doit débuter entre 7h00 et 7h30, la borne inférieure de sa fenêtre de temps vaudra 420 et la borne supérieure 450.

3.2.1.5 Les sommets

Afin de tenir compte des contraintes imposant un nombre de visites minimum dans chaque port selon un sens fixé, ces derniers seront représentés dans notre réseau à l'aide de plusieurs sommets. Rappelons que lorsque nous nous trouvons dans un port, trois directions peuvent être prises:

le sens positif: le long de la rive du lac dans le sens des aiguilles d'une montre

le sens négatif: le long de la rive du lac dans le sens inverse des aiguilles d'une montre

la direction frontale: traversée du lac pour rejoindre l'autre rive

De manière générale, la modélisation que nous avons faite d'un port dépend de son importance. Dans certains cas, sa situation géographique peut jouer un rôle (c'est le cas de Genève). Finalement, quelques ports ont des modélisations spéciales qui sont imposées par le type de service devant s'y effectuer. Nous allons décrire la façon dont chacun des

39 ports a été représenté dans notre réseau.

Ports importants avec liaisons frontales

Pour distinguer les trois types de directions pouvant être prises lorsque nous nous trouvons dans ce type de port, quatre sommets seront utilisés pour les représenter. Les arcs reliant entre eux les sommets associés à un même port seront appelés arcs internes par opposition aux arcs externes qui relient deux sommets associés à des ports différents. La figure 3.2 représente le port de Lausanne. Les ports de Vevey, Evian, Thonon,Yvoire et Nyon sont modélisés de la même manière.

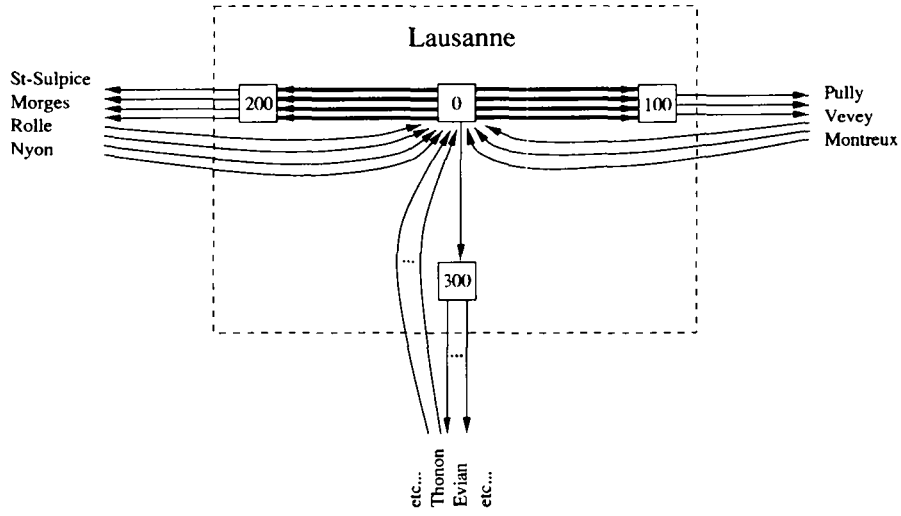


FIG. 3.2 -

Modélisation des ports importants avec liaisons frontales, exemple de Lausanne.

Dans la figure 3.2, le sommet 0 reçoit tous les arcs arrivant depuis les autres ports. A partir de ce sommet, 3 options sont possibles: partir dans le sens positif (vers le sommet 100), partir dans le sens négatif (vers le sommet 200) et partir dans la direction frontale (vers le sommet 300). Les contraintes indiquant qu'il faut visiter 4 fois Lausanne dans chacun des sens positif et négatif sont prises en compte en introduisant des arcs internes obligatoires. Quatre de ces arcs relient le sommet 0 au sommet 200 et quatre autres connectent le sommet 0 au sommet 100. Ils sont représentés en gras dans la figure. Ces arcs ont des temps de parcours nuls. A chacun d'eux est associée une fenêtre de temps indiquant la tranche horaire à l'intérieur de laquelle une visite de Lausanne doit être effectuée (voir tableau 3.3). Les autres arcs internes sont facultatifs; leur temps de parcours est nul également. Les arcs externes correspondent aux différents trajets directs existant entre le port de Lausanne et les autres ports. Ces trajets directs ont été définis dans la section 3.2.1.2. Même s'ils ne sont pas représentés en gras dans la figure, les arcs externes peuvent être obligatoires.

Dans les ports importants avec liaisons frontales, il est toujours possible, d'où que l'on vienne, de repartir dans n'importe quelle direction. Il est ainsi possible de faire demi-tour et de se diriger vers l'endroit d'où nous sommes venus.

Ports importants sans liaison frontale

Seules les directions positive et négative doivent être prises en compte dans ces ports. Nous pouvons donc les modéliser à l'aide de trois sommets seulement. La figure 3.3 représente le port de Montreux. Il s'agit du seul port modélisé de cette manière.

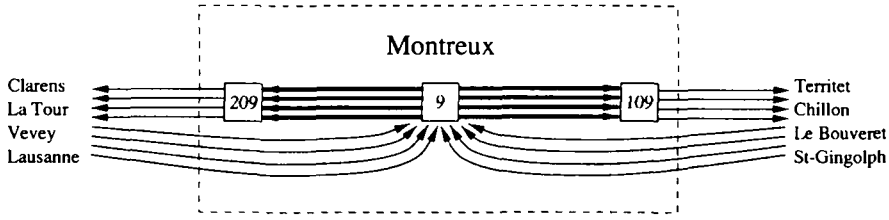


FIG. 3.3 –

Modélisation des ports importants sans liaison frontale, exemple de Montreux.

La modélisation du port de Montreux est identique à celle de Lausanne si ce n'est qu'aucune liaison frontale n'existe. Dans ce cas également, il est possible de repartir dans toutes les directions disponibles (i.e. sens positif et sens négatif) quel que soit l'endroit d'où l'on vient.

Port de Genève

Nous avons déjà plusieurs fois mentionné la particularité du port de Genève. Il se trouve au bout du lac et aucune direction positive, négative ou frontale n'est spécifiée pour ce port. Ce port est composé de quatre débarcadères (Mont-Blanc, Jardin Anglais, Eaux-Vives et Pâquis) entre lesquels les bateaux effectuent des liaisons. Nous aurions pu représenter chacun des débarcadères comme un port à part entière (c'est ce qui est fait par exemple pour Vevey où les trois débarcadères Vevey-Plan CGN, Vevey-Marché et Vevey-La Tour sont considérés comme des ports). Cependant, la navigation dans la rade de Genève est tellement complexe (forts courants, densité du trafic importante, etc.) que nous avons préféré les englober en un seul port. Afin de tenir compte de la simplification que nous avons faite, nous avons introduit un temps de parcours fictif de 20 minutes dans la modélisation du port de Genève. Ce temps de parcours permet de prendre en compte les éventuelles liaisons entre les débarcadères de la rade de Genève effectuées par un bateau entre son arrivée et son départ de Genève. La figure 3.4 représente la modélisation du port de Genève.

Les quatre visites minimum devant être effectuées à Genève sont modélisées à l'aide d'arcs internes obligatoires comme dans le cas des autres ports. Ces arcs cependant ont un temps de parcours qui n'est pas nul mais qui vaut 20 minutes.

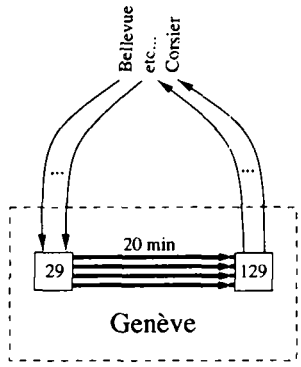


FIG. 3.4 -

Modélisation du port de Genève.

Ports secondaires avec liaisons frontales

Dans les ports secondaires avec liaisons frontales, nous voulons éviter qu'un bateau longeant la rive du lac puisse faire demi-tour et repartir dans la direction d'où il est venu. Ainsi lorsqu'on arrive dans un port en suivant le sens positif (resp. négatif), il est impossible de faire demi-tour et de repartir dans le sens négatif (resp. positif). La figure 3.5 représente le port de Morges. Les ports suivants sont modélisés de manière identique: Nernier, Hermance, Versoix, Coppet, Rolle.

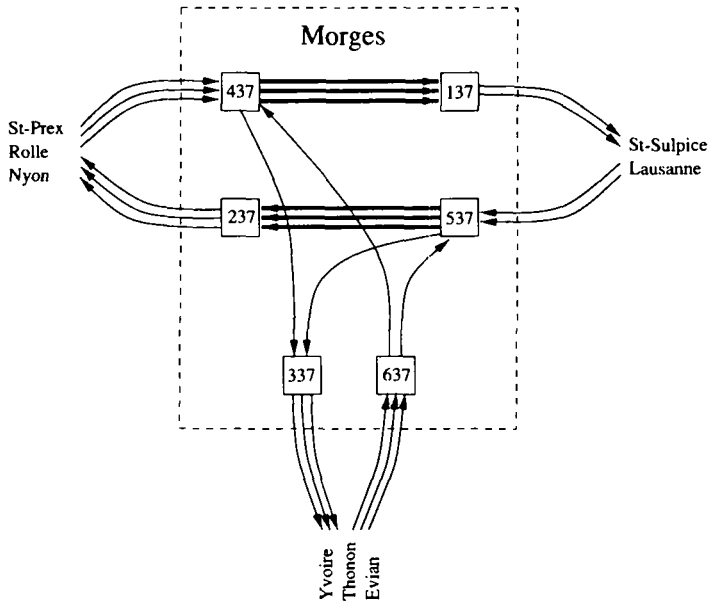


FIG. 3.5 -

Modélisation des ports secondaires avec liaisons frontales, exemple de Morges.

Six sommets sont nécessaires pour modéliser ce type de port. Comme précédemment, le nombre de visites imposées dans chaque port est pris en compte en introduisant des arcs internes obligatoires. Lorsqu'on arrive à Morges en suivant la rive du lac dans le sens positif, deux directions peuvent être prises, la direction frontale (arc (437, 337)) ou la direction positive (arc (437, 137)). De manière plus générale, dans ce type de port, il n'est jamais possible de faire demi-tour lorsqu'on longe la rive du lac.

Le but d'une telle modélisation est d'empêcher les aller-retours entre deux ports secondaires voisins. Sans cela, il serait par exemple possible d'effectuer le trajet St-Prex - Morges (St-Prex est alors visité une fois dans le sens positif), puis Morges - St-Prex (Morges est visité une fois dans le sens négatif), puis à nouveau St-Prex - Morges (nouvelle visite de St-Prex dans le sens positif), etc. Il est clair que de tels itinéraires ne sont absolument pas acceptables pour la CGN. La modélisation que nous utilisons pour les ports secondaires permet de les éviter.

Ports secondaires sans liaison frontale

La modélisation est identique à la précédente, la seule différence est l'absence de direction frontale. Les bateaux arrivant dans ces ports doivent continuer de longer la rive du lac dans la même direction. La figure 3.6 représente le port de Pully. Les ports de Lutry, Cully, Rivaz, Plan CGN, La Tour, Clarens, Territet, Chillon, Villeneuve, Meillerie, Tourronde, Amphion, Sciez, Excenevex, Chens, Anières, Corsier, Bellevue, Céligny, St-Prex et St-Sulpice sont modélisés de façon identique.

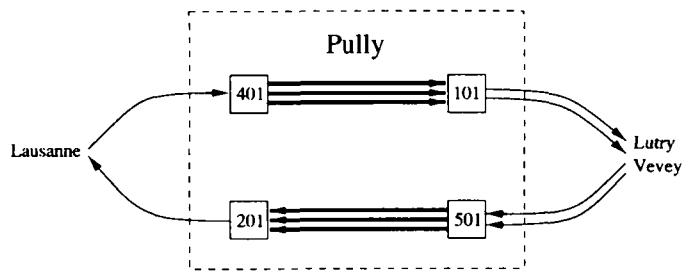


FIG. 3.6 –

Modélisation des ports secondaires sans liaison frontale, exemple de Pully.

Port de Margencel

Il s'agit d'un cas particulier de port secondaire sans liaison frontale. En effet, pour des raisons géographiques, nous devons pouvoir faire demi-tour dans ce port lorsque nous y arrivons en suivant le sens négatif. Un arc interne facultatif est donc ajouté pour permettre ce demi-tour. La figure 3.7 représente le port de Margencel.

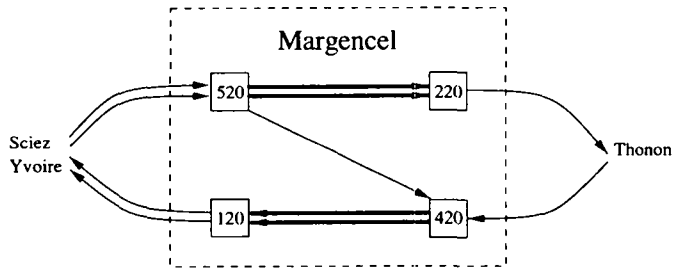


FIG. 3.7 –

Modélisation du port de Margencel.

Ports du Bouveret et de St-Gingolph

Ces deux ports constituent des cas très particuliers. Il faut pouvoir faire demi-tour dans ces ports quel que soit l'endroit d'où l'on vient, tout en empêchant la formation d'aller-retours: Bouveret - St-Gingolph - Bouveret - St-Gingolph, etc. La modélisation de ces deux ports est décrite dans la figure 3.8.

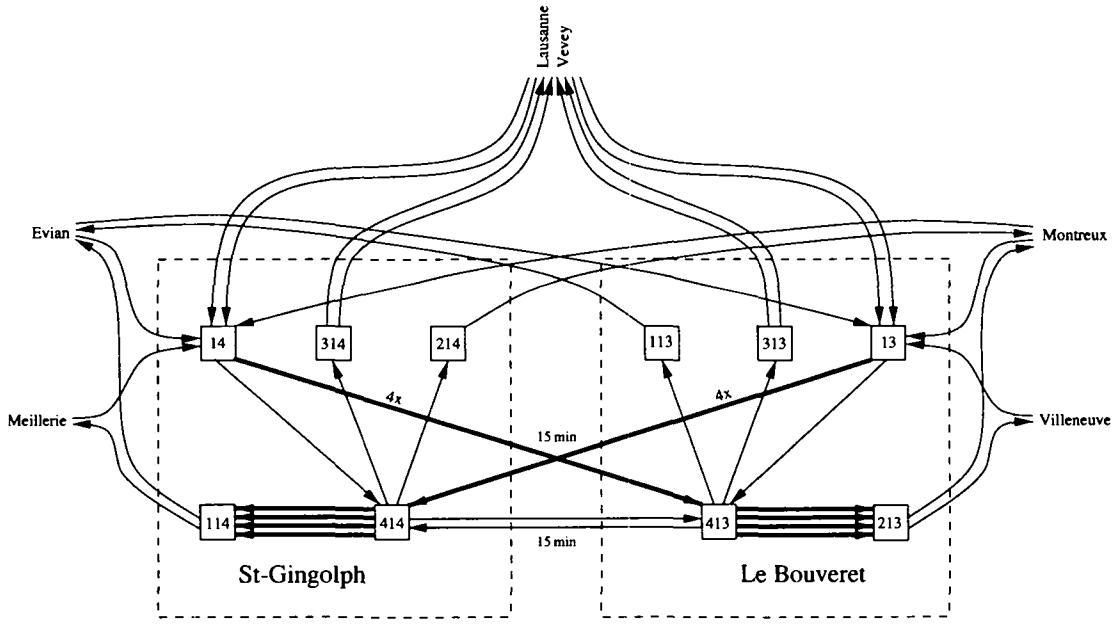


FIG. 3.8 –

Modélisation des ports du Bouveret et de St-Gingolph.

Tous les trajets aboutissant au Bouveret arrivent au sommet 13, ceux qui se terminent à St-Gingolph au sommet 14. A partir du sommet 13 (ou 14), il est possible de repartir dans n'importe quelle direction: direction frontale en passant par les sommets 413 et 313 (414 et 314 pour St-Gingolph), direction positive et direction négative.

Lorsque l'on va dans le sens positif (sens négatif dans le cas de St-Gingolph) en passant par les sommets 13, 413, 113 (14, 414, 214), ce trajet n'est pas comptabilisé. Il s'agit du défaut de cette modélisation. Les seuls trajets comptabilisés dans le sens positif pour le Bouveret sont les trajets Bouveret - St-Gingolph (13 - 414). De même pour St-Gingolph, les seuls trajets dans le sens négatif qui sont pris en compte sont les trajets St-Gingolph - Bouveret (14 - 413). Ces trajets doivent être effectués quatre fois et ils durent 15 minutes. Lorsque l'itinéraire Bouveret - St-Gingolph - Bouveret est effectué, au plus un des trajets Bouveret - St-Gingolph ou St-Gingolph - Bouveret dessert un arc interne obligatoire. Il est en effet impossible d'effectuer un tel parcours en passant par les sommets 13, 414, 14, 413.
Aucun aller-retour utilisant les sommets 413 et 414 n'est effectué car comme nous le verrons plus loin, les arcs facultatifs ne sont utilisés que dans des plus courts chemins et un plus court chemin ne comporte jamais les suites 413 - 414 - 413 ou 414 - 413 - 414.
La modélisation des visites devant être effectuées dans le sens négatif pour le Bouveret et positif pour St-Gingolph correspond à la modélisation classique que nous avons utilisée jusqu'à présent.

3.2.1.6 Caractéristiques du réseau

Le réseau est composé de 167 sommets représentant 39 ports. Il comporte 495 arcs. Les liaisons entre les ports sont représentées à l'aide de 186 arcs (93 dans chaque sens). Les trajets imposés sont modélisés à l'aide de 60 arcs multiples. La modélisation des ports nécessite l'emploi de 249 arcs internes. Parmi ces 495 arcs, 272 sont obligatoires (les 60 associés aux trajets imposés ainsi que 212 arcs internes), et 223 sont des arcs facultatifs (les arcs correspondant aux 186 trajets entre les ports plus 37 arcs internes).
Les temps de parcours associés aux arcs satisfont l'inégalité triangulaire. Cela est conforme à la réalité dans la mesure où un trajet direct entre deux ports est plus rapide qu'un trajet joignant les mêmes ports mais passant par un ou plusieurs ports intermédiaires.

3.2.2 Le problème à résoudre

Le réseau étant défini, il reste à préciser quel problème nous allons tenter de résoudre. Nous appellerons solution admissible un ensemble de tournées telles que:

I) chaque arc obligatoire est desservi exactement une fois et son service débute à l'intérieur de sa fenêtre de temps

II) le nombre maximum de bateaux pouvant stationner dans chaque port est respecté

III) il existe une transversale partant de Genève et passant par Le Bouveret avant de revenir à Genève et une transversale Le Bouveret - Genève - Le Bouveret.

IV) la durée de chaque tournée est inférieure à 13 heures (=780 minutes)

V) les bateaux quittent un port au plus tôt à 5h00 et rentrent au plus tard à 23h30.

Lors de la construction du réseau, nous n'avons pas mentionné les contraintes III) et V). Il en sera question plus loin dans la section 3.3.1. La contrainte II) est gérée par l'utilisateur. C'est lui qui indique le nombre de bateaux stationnant initialement dans chacun des ports de Lausanne, Genève, Yvoire et du Bouveret.

3.2.2.1 Modélisation des tournées et temps d'attente

Une tournée est modélisée comme une succession d'arcs obligatoires reliés par des chemins composés d'arcs facultatifs ou d'arcs obligatoires non desservis. Seuls des plus courts chemins sont utilisés pour relier entre eux deux arcs desservis successifs.

Afin de satisfaire les contraintes liées aux fenêtres de temps, il est nécessaire de permettre aux bateaux d'attendre dans un port qu'une fenêtre de temps "s'ouvre" avant de desservir l'arc associé à cette fenêtre. Une fenêtre de temps est "ouverte" lorsque l'heure courante se situe entre la borne inférieure et la borne supérieure définissant la fenêtre. Nous parlerons de fenêtre fermée dans le cas contraire. Le temps d'attente au départ d'une tournée n'est pas pris en compte dans le calcul de sa durée. Un bateau entre en service une fois qu'il débute son premier trajet.

Exemple:

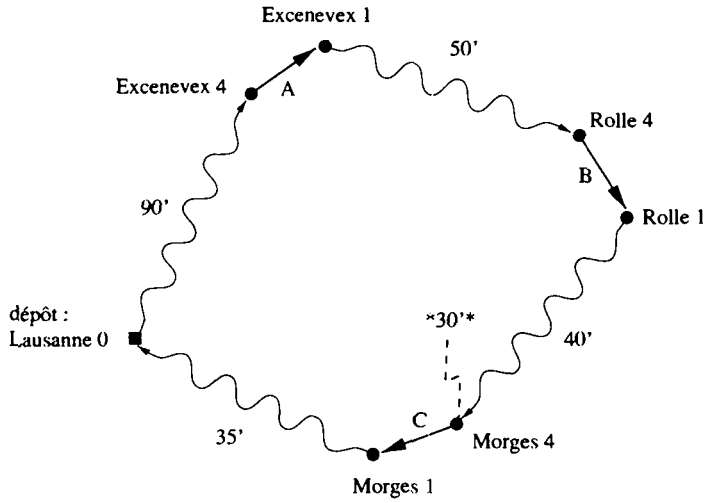


FIG. 3.9 -

Un exemple de tournée.

La figure ci-dessus représente une tournée effectuée depuis le port de Lausanne. Trois arcs obligatoires (en gras) sont desservis dans cette tournée; il s'agit d'arcs internes aux ports d'Excenevex, de Rolle et de Morges. Les arcs ondulés représentent les plus courts chemins reliant les arcs desservis. Les temps de parcours sont indiqués en minutes. Le départ de Lausanne a lieu à 9h00. Les départs des arcs A et B doivent se faire entre 9h00 et 12h00, le départ de l'arc C doit avoir lieu entre 12h30 et 16h30. En partant à 9h00 de Lausanne, le bateau arrive à Morges 4 à 12h00. Il doit donc attendre 30 minutes au port de Morges avant de pouvoir desservir l'arc C. Ce temps d'attente est indiqué entre les astérisques sur la figure 3.9.
Le nombre apparaissant après le nom d'un port permet de savoir sur quel type de sommet associé au port nous nous trouvons. La signification de ces nombres est la suivante:

0: sommet où arrivent tous les arcs externes
1: sommet d'où partent les arcs externes dans le sens positif

2: sommet d'où partent les arcs externes dans le sens négatif
3: sommet d'où partent les arcs externes dans la direction frontale
4: sommet où arrivent les arcs externes orientés dans le sens positif
5: sommet où arrivent les arcs externes orientés dans le sens négatif
6: sommet où arrivent les arcs externes en provenance de l'autre rive du lac

Ces nombres dépendent de la modélisation du port. Il y a des exceptions à la signification de ces nombres, elles concernent les ports de Genève, de St-Gingolph et du Bouveret qui comme nous l'avons vu ont des modélisations particulières.

3.2.2.2 La fonction objectif

Nous allons chercher à minimiser le temps total de service de l'ensemble des bateaux. Ce temps est composé du temps total de parcours des bateaux et du temps total d'attente de ces derniers. Le temps total de parcours est le temps pendant lequel les bateaux circulent. Le temps total d'attente est le temps durant lequel les bateaux stationnent dans un port après avoir effectué leur premier trajet.

3.2.3 Remarques finales

Le problème que nous allons résoudre est un problème de tournées sur les arcs avec capacité et fenêtres de temps. Par rapport au PTAC standard présenté aux chapitres 1 et 2, nous avons, en plus des fenêtres de temps, plusieurs dépôts de capacité limitée. La durée maximum de service des bateaux correspond à la capacité des véhicules du PTAC standard. Dans ce problème, la capacité résiduelle des véhicules ne diminue pas lorsque des arcs sont traversés sans être desservis. La situation est différente dans notre problème. Traverser un arc, même sans le desservir, réduit le temps pendant lequel un bateau peut encore être en service.

A notre connaissance, les seuls articles traitant de problèmes de tournées sur les arcs avec contraintes temporelles sont ceux de Eglese et Li [Egl96], Eglese et Letchford [Egl95], Dror et Stern et Trudeau [Dro87]. Il s'agit en général de contraintes de temps particulières: date limite avant laquelle un service doit être effectué (i.e. fenêtres de temps avec borne inférieure nulle), contraintes de précédences (des groupes d'arcs doivent être desservis avant d'autres groupes). Ces contraintes ne correspondent pas exactement à nos contraintes temporelles.
En revanche, de nombreux articles existent qui traitent de problèmes de tournées sur les sommets avec fenêtres de temps, citons par exemple Solomon [Sol87], et Rochat et Semet [Roc94]. Des heuristiques basées sur des méthodes d'insertion sont proposées dans [Sol87]. Nous allons nous en inspirer pour la résolution de notre problème.

3.3 Résolution

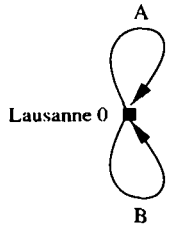
La résolution du problème de la confection des horaires de bateaux va s'effectuer en deux temps. Nous allons tout d'abord utiliser des heuristiques d'insertion pour tenter d'obtenir un ensemble de tournées admissibles satisfaisant les contraintes I) à V) décrites en 3.2.2. La solution que nous obtenons alors, même si elle satisfait bien les contraintes que

nous nous sommes fixées, n'est pas forcément réaliste du point de vue de la CGN. C'est pourquoi, dans un deuxième temps, nous présenterons des procédures de post-optimisation permettant à l'utilisateur d'apporter un certain nombre de modifications à la solution fournie par les heuristiques d'insertion.

3.3.1 Initialisation des tournées

Deux types de contraintes n'ont pas encore été prises en compte dans le modèle que nous avons décrit à la section 3.2. Il s'agit des contraintes III) et V) décrites en 3.2.2. Elles concernent les deux transversales quotidiennes devant être effectuées entre Le Bouveret et Genève ainsi que les heures de départ au plus tôt et les heures d'arrivée au plus tard des bateaux dans les ports. Une façon de prendre en compte ces types de contraintes consiste à initialiser les tournées que nous allons devoir effectuer en y insérant des trajets fictifs. Comme ces trajets fictifs dépendent du nombre de tournées à réaliser ainsi que du nombre de bateaux stationnant dans les ports de Lausanne, de Genève, du Bouveret et d'Yvoire (données entrées par l'utilisateur), ils sont ajoutés au réseau lors de la construction des tournées.

Pour tenir compte de l'heure de départ au plus tôt d'un bateau, une boucle est ajoutée sur un des sommets représentant le port où stationne le bateau (il s'agit du sommet associé au port où aboutissent tous les arcs externes). Cette boucle est un arc obligatoire devant être parcouru entre 5h00 et 23h30. Son temps de parcours est nul. Une deuxième boucle identique est également utilisée pour tenir compte de l'heure d'arrivée au plus tard du bateau dans le port. Les tournées partant de Lausanne ont donc l'allure suivante après initialisation:



300

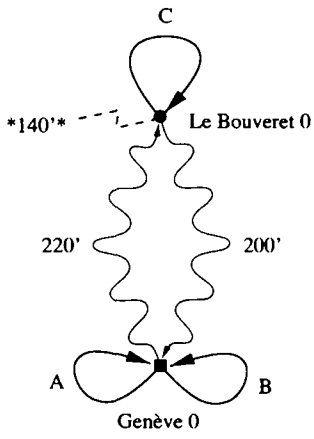
	départ			arrivée
5h00	Lausanne 0	$\xrightarrow{A}$	5h00	Lausanne 0
5h00	Lausanne 0	$\xrightarrow{B}$	5h00	Lausanne 0

FIG. 3.10 -

Initialisation des tournées, exemple de Lausanne.

La figure 3.10 représente les deux boucles insérées pour initialiser la tournée d'un bateau basé à Lausanne. Le tableau de droite indique les trajets effectués (ce sont des trajets fictifs). Un temps d'attente de 300 minutes est créé par l'insertion de la boucle A car le bateau doit attendre de 0h00 à 5h00 avant de pouvoir la desservir. Ce temps d'attente n'influence pas la durée de la tournée car il se situe au début de la tournée, avant le début du premier trajet. Les arcs obligatoires qui seront desservis par cette tournée seront toujours insérés entre les arcs A et B.

Le même principe d'arcs fictifs ajoutés au réseau est utilisé pour tenir compte des contraintes concernant les deux transversales. Si nous prenons le cas de la transversale Genève - Le Bouveret - Genève, deux arcs fictifs (les boucles A et B dans la figure 3.11) sont insérés à Genève pour tenir compte de l'heure de départ au plus tôt et de l'heure d'arrivée au plus tard. Une troisième boucle (boucle C) est insérée au Bouveret. Cette boucle permet de prendre en compte la contrainte disant qu'il faut arriver au Bouveret entre 11h00 et 15h00 (voir 3.1.3.3). Finalement, comme il n'existe pas de liaison directe Genève - Le Bouveret, un plus court chemin reliant Genève au Bouveret est inséré entre la boucle A et la boucle C. Un deuxième plus court chemin reliant cette fois Le Bouveret à Genève est inséré entre la boucle C et la boucle B. Cette initialisation est représentée dans la figure 3.11. Le principe est le même pour la transversale Le Bouveret - Genève - Le Bouveret.



	départ			arrivée
300	5h00	Genève 0	$\xrightarrow{A}$ 5h00	Genève 0
	5h00	Genève 0	$\longrightarrow$ 5h20	Genève 1
	5h20	Genève 1	$\longrightarrow$ 6h30	Yvoire 0
	6h30	Yvoire 0	$\longrightarrow$ 6h30	Yvoire 2
	6h30	Yvoire 2	$\longrightarrow$ 7h45	Evian 0
	7h45	Evian 0	$\longrightarrow$ 7h45	Evian 2
	7h45	Evian 2	$\longrightarrow$ 8h40	Le Bouveret 0
140	11h00	Le Bouveret 0	$\xrightarrow{C}$ 11h00	Le Bouveret 0
	11h00	Le Bouveret 0	$\longrightarrow$ 11h00	Le Bouveret 4
	11h00	Le Bouveret 4	$\longrightarrow$ 11h00	Le Bouveret 1
	11h00	Le Bouveret 1	$\longrightarrow$ 12h05	Evian 0
	12h05	Evian 0	$\longrightarrow$ 12h05	Evian 1
	12h05	Evian 1	$\longrightarrow$ 13h10	Yvoire 0
	13h10	Yvoire 0	$\longrightarrow$ 13h10	Yvoire 1
	13h10	Yvoire 1	$\longrightarrow$ 14h20	Genève 0
	14h20	Genève 0	$\xrightarrow{B}$ 14h20	Genève 0

FIG. 3.11 -

Initialisation de la transversale Genève - Le Bouveret - Genève.

Le tableau de la figure 3.11 représente le détail, après initialisation, de la tournée effectuant la transversale Genève - Le Bouveret - Genève . Nous avons mentionné tous les arcs traversés, même les arcs internes aux ports ayant un temps de parcours nul. Nous pouvons remarquer qu'au Bouveret le bateau attend 140 minutes avant de pouvoir desservir la boucle C. Cela vient du fait que nous faisons partir les tournées dès que possible. En repoussant ce départ, nous aurions pu éliminer ce temps d'attente en le transférant au début de la tournée. Suivant la suggestion de Solomon [Sol87], nous continuerons à faire partir les bateaux dès que possible et ce n'est qu'une fois les tournées construites que nous ajusterons les heures de départ selon des critères que nous spécifierons.
Une fois les transversales initialisées, des arcs peuvent y être insérés soit entre les boucles A et C, soit entre les boucles C et B.

3.3.2 Les heuristiques d'insertion

Dans le cadre de notre problème, une heuristique d'insertion consiste à sélectionner un arc obligatoire non traité et à l'insérer dans une tournée. Le choix de l'arc à insérer, la

manière de déterminer la tournée dans laquelle il sera desservi et l'endroit de cette tournée où il va prendre place sont autant de critères sur lesquels il est possible de jouer et qui permettent d'obtenir différentes méthodes d'insertion.

3.3.2.1 Notations

Une tournée T est un ensemble d'arcs obligatoires reliés par des plus courts chemins (voir 3.2.2.1). Nous noterons $T = (A_1, A_2, ..., A_{|T|})$ la tournée desservant les arcs obligatoires $A_1, A_2, ..., A_{|T|}$. Les arcs A_1 et $A_{|T|}$ sont les boucles insérées dans T lors de l'initialisation des tournées. Le temps de parcours de ces arcs est nul et aucun arc ne sera inséré avant A_1 ni après $A_{|T|}$. Nous noterons $[l_{A_i}, b_{A_i}]$ la fenêtre de temps de l'arc $A_i, i = 1, ..., |T|$ et h_{A_i} l'heure à laquelle débute son service. Une tournée T est dite admissible si et seulement si:

1) $l_{A_i} \leq h_{A_i} \leq b_{A_i} \ \forall \ i = 1, ..., |T|$

2) $h_{A_{|T|}} - h_{A_1} \leq 780$

La condition 1) indique que l'heure du début du service d'un arc A_i s'effectue à l'intérieur de la fenêtre de temps qui lui est associée. La condition 2) signifie que la durée de la tournée ne dépasse pas 13 heures (=780 minutes).

Nous noterons par R^- l'ensemble des arcs obligatoires qui ne sont pas encore traités. Soit $B \in R^-$, soit $[l_B, b_B]$ sa fenêtre de temps et soit une tournée $T = (A_1, ..., A_{|T|})$. Nous noterons par (T, B, A_j) l'insertion de B dans T entre les arcs A_{j-1} et A_j $(j = 2, ..., |T|)$. L'heure du début du service de l'arc A_i $(i = 1, ..., |T|)$ lorsque l'insertion (T, B, A_j) est effectuée sera notée $h_{A_i}(B, A_j)$. L'insertion d'un arc dans T influence uniquement l'heure de service des arcs obligatoires qui suivent l'arc inséré. Nous avons donc $h_{A_i}(B, A_j) = h_{A_i}$ pour $i = 1, ..., j-1$. Lorsque l'arc B est inséré dans T, un temps d'attente positif est introduit, si nécessaire, de manière à ce que l'heure du début du service de B soit égale à l_B.

Une insertion (T, B, A_j) est admissible si et seulement si la tournée obtenue en insérant l'arc B dans T entre les arcs A_{j-1} et A_j est admissible.

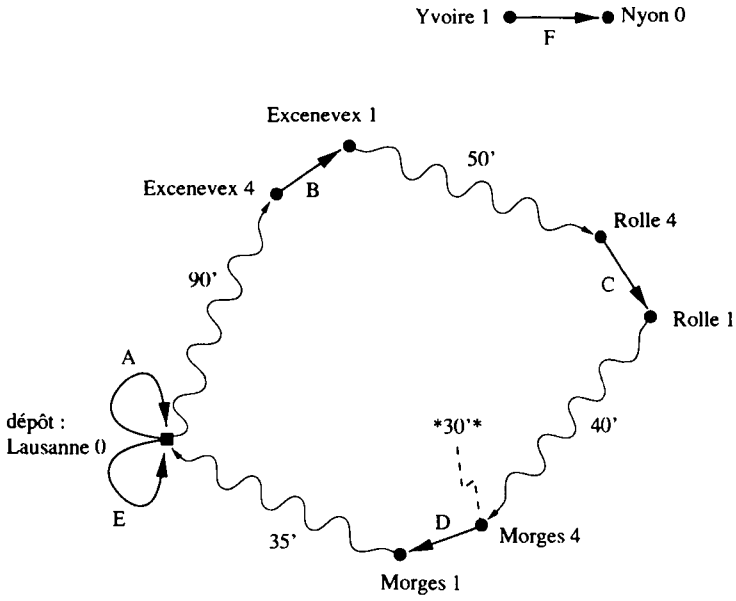
Nous appellerons décalage dû à l'insertion (T, B, A_j) la différence, notée $Dec(T, B, A_j)$, qui est définie par: $Dec(T, B, A_j) = h_{A_j}(B, A_j) - h_{A_j}$. Cette différence correspond au décalage temporel du début du service de A_j provoqué par l'insertion de l'arc B entre les arcs A_{j-1} et A_j. Ce décalage est toujours positif ou nul. Un décalage admissible est un décalage engendré par une insertion admissible. Le plus petit décalage admissible provoqué par l'insertion de B dans T sera appelé décalage admissible optimal de B dans T et il sera noté $Dec_{adm}(T, B)$. Une insertion admissible de B dans T provoquant un décalage égal à $Dec_{adm}(T, B)$ sera appelée insertion admissible optimale de B dans T.

Etant donné deux ports i et j, nous noterons par tp_{ij} le temps de parcours nécessaire pour relier i à j en passant par le chemin le plus court. Le temps d'attente au port i sera noté ta_i. Afin d'alléger les notations, un port sera représenté par ses deux premières lettres. Nous noterons par exemple, tp_{YvNy} le temps de parcours entre les ports d'Yvoire et de Nyon.

3.3.2.2 Insertion d'un arc dans une tournée

Soit $T = (A_1, ..., A_{|T|})$ une tournée admissible et soit B un arc de R^-. Si une insertion admissible de B dans T existe, alors l'arc B est inséré de telle sorte que le décalage engendré par cette insertion soit égal à $Dec_{adm}(T, B)$. Cela signifie qu'on effectue l'insertion admissible optmimale de B dans T. Si aucune insertion admissible de B dans T n'existe, B n'est pas introduit dans T , et il n'y a pas d'insertion admissible optimale possible.

Exemple:



Arc	Fenêtre de temps		début du
	borne inf.	borne sup.	service à
A	5h00	23h30	9h00
B	9h00	12h00	10h30
C	9h00	12h00	11h20
D	12h30	16h30	12h30
E	5h00	23h30	13h05
F	10h30	11h30	à trouver

FIG. 3.12 -

Insertion de l'arc F dans la tournée $T = (A, B, C, D, E)$.

	Yvoire	Nyon	Lausanne	Excenevex	Rolle	Morges
Yvoire	0	20	80	20	30	55
Nyon	20	0	95	40	40	75
Lausanne	80	95	0	90	70	35
Excenevex	20	40	90	0	50	75
Rolle	30	40	70	50	0	40
Morges	55	75	35	75	40	0

TAB. 3.5 -

Temps de parcours entre les ports.

La figure 3.12 représente une tournée dans laquelle on souhaite insérer l'arc obligatoire (Yvoire 1, Nyon 0) (arc F). Les arcs obligatoires A et E représentent les boucles introduites dans la tournée lors de la phase d'initialisation. Les autres arcs obligatoires correspondent à des arcs internes aux ports d'Excenevex, de Rolle et de Morges. La fenêtre de temps de chacun de ces arcs est définie dans le tableau de la figure 3.12. L'heure du début du service de chaque arc obligatoire se trouve également dans ce tableau. Le début du service de l'arc A a été fixé arbitrairement à 9h00.

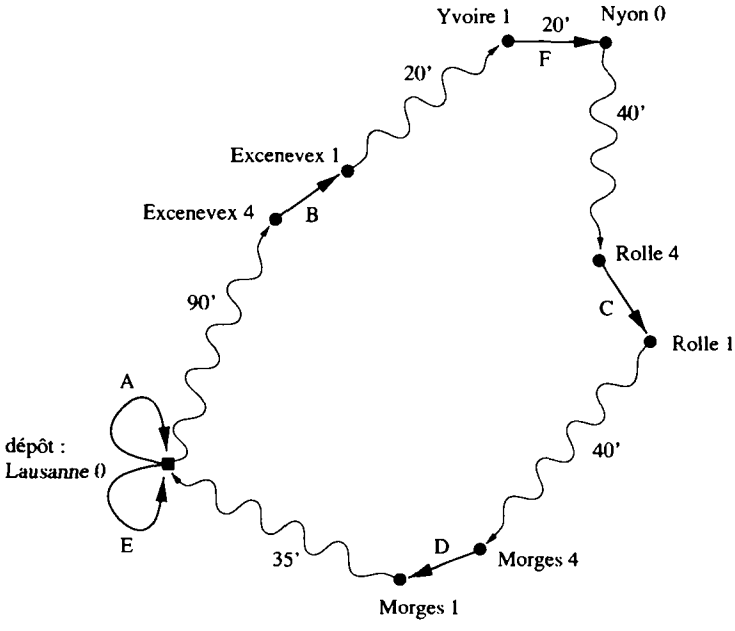
Le tableau 3.5 contient les temps de parcours des plus courts chemins reliant entre eux les ports d'Yvoire, Nyon, Lausanne, Excenevex, Rollle et Morges. Ces temps de parcours sont utilisés pour calculer la valeur des décalages provoqués par l'insertion de l'arc F dans la tournée $T = (A, B, C, D, E)$ de la figure 3.12. Les valeurs des décalages provoqués par l'insertion de l'arc F dans T sont les suivantes:

$$Dec(T, F, B) = tp_{LaYv} + tp_{YvNy} + tp_{NyEx} - tp_{LaEx} = 80 + 20 + 40 - 90 = 50$$
$$Dec(T, F, C) = tp_{ExYv} + tp_{YvNy} + tp_{NyRo} - tp_{ExRo} = 20 + 20 + 40 - 50 = 30$$
$$Dec(T, F, D) = tp_{RoYv} + tp_{YvNy} + tp_{NyMo} - (tp_{RoMo} + ta_{Mo}) = 30 + 20 + 75 - (40 + 30) = 55$$
$$Dec(T, F, E) = tp_{MoYy} + tp_{YvNy} + tp_{NyLa} - tp_{MoLa} = 55 + 20 + 95 - 35 = 135$$

L'insertion (T, F, B) n'est pas admissible. En effet, l'heure de service de l'arc C lorsque l'arc F est inséré immédiatement avant l'arc B est la suivante:

$$h_C(F, B) = h_C + Dec(T, F, B) = 11h20 + 50' = 12h10 > b_C = 12h00$$

Les insertions (T, F, D) et (T, F, E) ne sont pas admissibles non plus. Dans ces deux cas, le début du service de l'arc F s'effectue à respectivement 11h50 et 13h25, soit après 11h30 qui est la valeur de la borne supérieure de sa fenêtre de temps. La seule insertion admissible est l'insertion (T, F, C). Il faut donc insérer l'arc F dans la tournée immédiatement avant l'arc C. Le décalage admissible optimal engendré par l'insertion de F dans T est alors de 30 minutes (i.e. $Dec_{adm}(T, F) = 30$). La nouvelle tournée que nous obtenons en effectuant l'insertion (T, F, C) est représentée à la figure 3.13.



Arc	Fenêtre de temps		début du service à
	borne inf.	borne sup.	
A	5h00	23h30	9h00
B	9h00	12h00	10h30
C	9h00	12h00	11h50
D	12h30	16h30	12h30
E	5h00	23h30	13h05
F	10h30	11h30	10h50

FIG. 3.13 -

Tournée obtenue en effectuant l'insertion (T, F, C).

La tournée de la figure 3.13 s'achève à la même heure que la tournée décrite à la figure 3.12. Cela vient du fait que le décalage d'une demi-heure provoqué par l'insertion de l'arc F entre les arcs B et C a permis de supprimer la demi-heure d'attente au port de Morges.

3.3.2.3 Choix de l'arc à insérer

Etant donné l'ensemble R^- des arcs obligatoires restant à desservir et une tournée admissible T, l'arc B^* de R^- choisi pour être inséré dans T est celui qui, parmi tous les arcs de R^-, minimise le décalage admissible optimal. Lorsque plusieurs arcs de R^- permettent d'atteindre la plus petite valeur du décalage admissible optimal, B^* est choisi aléatoirement parmi eux. Si aucun arc de R^- ne peut être introduit dans T de manière à obtenir une tournée admissible, aucune insertion n'est effectuée dans T.

3.3.2.4 Méthode séquentielle d'insertion

La première méthode d'insertion que nous avons testée consiste à insérer successivement les arcs de R^- dans une même tournée. Lorsqu'aucun arc de R^- ne peut plus être introduit dans la tournée considérée, une nouvelle tournée est construite pour autant que le nombre de tournées, K, introduit par l'utilisateur ne soit pas déjà atteint. Si la construction de K tournées est achevée et qu'il reste encore des arcs obligatoires non desservis, nous dirons que l'algorithme a échoué. Une telle stratégie d'insertion revient à construire les itinéraires de chaque bateau les uns après les autres.

Algorithme: INSERTION_SEQUENTIELLE

ENTREE: Le réseau décrit en 3.2.1, K tournées déjà initialisées, un ordre $T_{O_1},...,T_{O_K}$ de ces tournées.

SORTIE: Une solution admissible au sens défini en 3.2.2 ou une indication signifiant que l'algorithme a échoué.

Etape 0. Poser $i := 1$ et mettre dans R^- l'ensemble des arcs obligatoires non encore desservis.

Etape 1. Déterminer l'arc B^* à insérer dans la tournée T_{O_i} selon le critère défini en 3.3.2.3.
Si B^* existe:

> Poser $R^- := R^- \setminus B^*$.
> Effectuer l'insertion admissible optimale de B^* dans T_{O_i}.
> Si $R^- \neq \emptyset$ recommencer l'étape 1, sinon aller à l'étape 3.

Sinon aller à l'étape 2.

Etape 2. Si $i = K$, STOP. L'algorithme a échoué. Sinon poser $i := i + 1$ et retourner à l'étape 1.

Etape 3. Une solution admissible utilisant i tournées a été trouvée, STOP.

L'utilisateur a la possibilité de définir l'ordre dans lequel les K tournées sont construites. Il peut également faire en sorte que cet ordre soit généré aléatoirement. En cas d'échec de l'algorithme, le nombre d'arcs n'ayant pas pu être insérés est indiqué à l'utilisateur.

3.3.2.5 Méthode d'insertion en parallèle

La méthode d'insertion en parallèle consiste à tester toutes les insertions admissibles des arcs de R^- dans chacune des K tournées. La meilleure insertion (celle qui minimise le décalage admissible optimal) est effectuée. Ce procédé est répété jusqu'à ce qu'il ne soit plus possible d'insérer, de manière admissible, un arc de R^- dans aucune des K tournées, ou jusqu'à ce que tous les arcs de R^- aient pu trouver une place dans une tournée.

Algorithme: INSERTION_PARALLELE

ENTREE: Le réseau décrit en 3.2.1, K tournées déjà initialisées: $T_1, ..., T_K$.

SORTIE: Une solution admissible au sens défini en 3.2.2 ou une indication signifiant que l'algorithme a échoué.

Etape 0. Mettre dans R^- l'ensemble des arcs obligatoires non encore desservis.

Etape 1. Déterminer l'arc B^* et la tournée T_{k^*} tels que:

$$Dec_{adm}(B^*, T_{k^*}) \leq Dec_{adm}(B, T_k) \ \forall \ B \in R^-, \ \forall \ k = 1, ..., K.$$

Si B^* et T_{k^*} existent:

Poser $R^- := R^- \setminus B^*$.
Effectuer l'insertion admissible optimale de B^* dans T_{k^*}.
Si $R^- \neq \emptyset$ recommencer l'étape 1, sinon aller à l'étape 2.

Sinon STOP. L'algorithme a échoué.

Etape 2. Une solution admissible utilisant au plus K tournées a été trouvée, STOP.

3.3.2.6 Traitement des temps d'attente

Lorsque nous avons réussi à construire une solution admissible, nous tentons de réduire le temps de service de chaque tournée en transférant les éventuels temps d'attente au début de la tournée. Rappelons que le temps d'attente qui précède le premier trajet de chaque tournée n'est pas pris en compte dans le calcul de la durée de la tournée (voir 3.2.2.1). Le transfert des temps d'attente au début d'une tournée se fait en retardant l'heure de départ du premier trajet de la tournée. Cette opération est réalisée de manière à réduire autant que possible les temps d'attente qui suivent le premier trajet d'une tournée tout en préservant l'admissibilité de cette dernière.

Soit $T = (A_1, ..., A_{|T|})$ une tournée admissible. Une partie (la plus grande possible) des

temps d'attente ta_{A_i} $(i = 2, ..., |T|)$ sont transférés en début de tournée, c'est-à-dire dans ta_{A_1}. La valeur de ta_{A_1} va augmenter ce qui va retarder l'heure de départ h_{A_1} du premier trajet de la tournée. Comme l'admissibilité de T doit être préservée, ce retard sera d'au plus t_{min1} minutes, où t_{min1} est défini de la façon suivante:

$$t_{min1} = \min \left\{ \min_{i=2,...,|T|} \{ \sum_{j=2}^{i} (ta_{A_j}) + b_{A_i} - h_{A_i} \},\ b_{A_1} - h_{A_1} \right\}$$

Comme nous voulons réduire les temps d'attente ta_{A_i} $(i = 2, ..., |T|)$, l'heure de départ de T ne pourra pas non plus être retardée de plus de t_{min2} minutes, où t_{min2} est défini par:

$$t_{min2} = \sum_{i=2}^{|T|} ta_{A_i}$$

A la place de débuter à l'heure h_{A_1}, la tournée T va commencer à l'heure $h_{A_1} + Ret(T)$ où $Ret(T)$ est défini de la manière suivante:

$$Ret(T) = \min \{ t_{min1},\ t_{min2} \}$$

Si les temps d'attente ta_{A_i} $(i = 2, ..., |T|)$ sont tous nuls, l'heure à laquelle le premier trajet de T est effectué n'est pas modifiée $(Ret(T) = 0)$.

Exemple:

Nous allons considérer la tournée $T = (A, B, C, D, E)$ représentée à la figure 3.12. Le tableau 3.6 contient les arcs desservis dans cette tournée, l'heure à laquelle débute leur service, leur temps d'attente ainsi que leur fenêtre de temps. Nous supposons toujours que la journée débute à 0h00. Les heures de départ, les temps d'attente et les bornes des fenêtres de temps sont donnés en minutes, les heures correspondantes figurent entre parenthèses.

Arc	Début du service à	Temps d'attente	Fenêtre de temps	
			borne inf.	borne sup.
A	540 (9h00)	540	300 (5h00)	1410 (23h30)
B	630 (10h30)	0	540 (9h00)	720 (12h00)
C	680 (11h20)	0	540 (9h00)	720 (12h00)
D	750 (12h30)	30	750 (12h30)	990 (16h30)
E	785 (13h05)	0	300 (5h00)	1410 (23h30)

TAB. 3.6 -

Pour transférer les temps d'attente au début de la tournée, nous allons calculer les valeurs de t_{min1} et t_{min2} associées à T.

$t_{min1} = \min\{0+720-630, 0+720-680, 30+990-750, 30+1410-785, 1410-540\}$
$t_{min1} = \min\{90, 40, 270, 655, 840\} = 40$ minutes
$t_{min2} = 30$ minutes

Nous avons donc $Ret(T) = 30$ minutes. Le début de la tournée T sera donc retardé de 30 minutes. Le service de l'arc A débutera à 9h30 au lieu de 9h00. Le tableau 3.7 contient les heures de service et les temps d'attente des arcs lorsque T commence à 9h30. Le temps d'attente de 30 minutes qui existait avant de desservir l'arc D a disparu. La tournée T débute maintenant à 9h30 et s'achève à 13h05, elle dure 3 heures et 35 minutes. Sa durée était avant de 4 heures et 5 minutes.

Arc	Début du service à	Temps d'attente	Fenêtre de temps	
			borne inf.	borne sup.
A	570 (9h30)	570	300 (5h00)	1410 (23h30)
B	660 (11h00)	0	540 (9h00)	720 (12h00)
C	680 (11h50)	0	540 (9h00)	720 (12h00)
D	750 (12h30)	0	750 (12h30)	990 (16h30)
E	785 (13h05)	0	300 (5h00)	1410 (23h30)

TAB. 3.7 -

3.3.3 Post-optimisation des tournées

Nous avons développé des procédures permettant à l'utilisateur de modifier des tournées pour tenir compte de certaines contraintes que nous avons jusque-là négligées.

3.3.3.1 Elimination de trajets redondants

Il peut arriver que deux bateaux effectuent le même trajet dans un laps de temps très faible. Ces trajets seront alors appelés trajets redondants. Nous ne traiterons que les trajets redondants directs, c'est-à-dire ceux qui sont représentés par un arc dans le réseau défini en 3.2.1. Un trajet redondant composé de plusieurs arcs sera considéré comme plusieurs trajets directs redondants composés d'un arc uniquement. Dès à présent lorsque nous utiliserons le terme trajet dans cette section, ce sera pour désigner une liaison directe entre deux ports. L'impact des trajets redondants sur la qualité de l'horaire est difficile à évaluer. Il dépend de l'origine et de la destination des bateaux impliqués ainsi que de l'écart qui les sépare dans le temps. Ainsi, deux bateaux effectuant le trajet Evian - Lausanne à 10 minutes d'intervalle, l'un venant de Thonon et transitant par Evian et Lausanne pour aller à Morges et l'autre venant de Tourronde et ayant pour but Pully, peut être concevable. En effet, la ligne Evian - Lausanne est une des principales liaisons desservies par la CGN, et le fait que deux bateaux se suivent de près sur ce parcours ne constitue pas forcément un inconvénient (surtout si cela se passe aux heures de pointe). En revanche, les trajets redondants apparaissant sur des liaisons secondaires ne sont pas acceptables pour la CGN, même si l'intervalle de temps qui les sépare est de l'ordre d'une demi-heure. Seul un utilisateur capable d'évaluer correctement les effets négatifs des trajets redondants est à même de modifier l'horaire en vue d'atténuer (ou de supprimer) ces effets. Les trajets redondants peuvent être des arcs desservis, des arcs obligatoires non desservis ou des arcs non obligatoires.

Deux méthodes sont proposées pour tenter de réduire le nombre de trajets redondants. La première méthode ne modifie que les heures auxquelles les arcs sont parcourus. Les itinéraires empruntés par les tournées ne changent pas. La deuxième méthode consiste à supprimer certains arcs d'une tournée afin de les réinsérer soit dans la même tournée, soit dans une autre.

3.3.3.1.1 Modifications temporelles

Nous allons examiner le cas où deux tournées $T = (A_1, ..., A_{|T|})$ et $T' = (A'_1, ..., A'_{|T'|})$ effectuent un trajet identique, $Traj_{red}$ à très peu d'intervalle. Si $Traj_{red}$ est un arc desservi dans T (resp. T'), nous désignerons cet arc par A_k (resp. A'_q). Sinon A_k (resp. A'_q) est l'arc de T (resp. T') desservi immédiatement avant $Traj_{red}$. Pour augmenter l'intervalle de temps entre le parcours de $Traj_{red}$ dans T et T', nous allons envisager une des opérations suivantes: avancer ou retarder l'heure de service de l'arc A_k dans T, retarder ou avancer l'heure de service de l'arc A'_q dans T'. Ces opérations sont effectuées de manière à préserver l'admissibilité (au sens défini en 3.3.2.1) des tournées T et T'.

Considérons le cas de l'arc A_k dans T (le traitement de A'_q dans T' est similaire). Nous pouvons retarder son heure de service d'au plus $Ret_{A_k}(T)$ minutes où le terme $Ret_{A_k}(T)$ est défini par:

$$Ret_{A_k}(T) = \min \left\{ \min_{i=k+1,...,|T|} \{ \sum_{j=k+1}^{i} (ta_{A_j}) + b_{A_i} - h_{A_i} \},\ b_{A_k} - h_{A_k} \right\}$$

La définition de $Ret_{A_k}(T)$ est très semblable à celle de t_{min1} présentée en 3.3.2.6. En fait nous avons $t_{min1} = Ret_{A_1}(T)$. Au lieu de retarder l'heure du début de la tournée, nous retardons l'heure de service des arcs à partir de l'arc A_k. Le service de A_k peut être repoussé d'au plus $Ret_{A_k}(T)$ minutes sans briser l'admissibilité de T. Il est retardé en ajoutant $Ret_{A_k}(T)$ au temps d'attente ta_{A_k}.

Avancer l'heure de service de l'arc A_k est un peu plus compliqué. Soit arr_{A_i} l'heure d'arrivée de la tournée T en A_i. L'heure de service de l'arc A_k peut être avancée d'au plus $Av_{A_k}(T)$ minutes où $Av_{A_k}(T)$ est calculée à l'aide de la procédure décrite ci-dessous:

Algorithme: CALCULER_AVANCE

ENTREE: Le réseau décrit en 3.2.1, une tournée $T = (A_1, ..., A_{|T|})$ et un arc A_k desservi dans T.

SORTIE: La valeur maximum, $Av_{A_k}(T)$ dont le service de A_k peut être avancé.

Etape 0. Poser $i := 1$, et $Av_{A_k}(T) = h_{A_i} - l_{A_i}$.

Etape 1. Poser $i := i + 1$.
Si $Av_{A_k}(T) \geq arr_{A_i} - l_{A_i}$ alors poser $Av_{A_k}(T) = arr_{A_i} - l_{A_i} + ta_{A_i}$
Sinon poser $Av_{A_k}(T) = Av_{A_k}(T) + ta_{A_i}$.
Si $i = k$ STOP, sinon recommencer l'étape 1.

L'exemple qui va suivre illustre le fonctionnement de la procédure CALCULER_AVANCE.

Exemple:

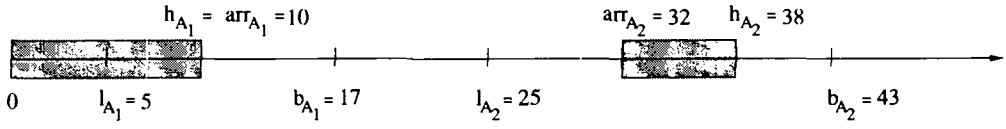


FIG. 3.14 -

Nous allons considérer une tournée $T = (A_1, A_2)$, et supposer que nous voulons avancer l'heure du début du service de l'arc A_2 de la plus grande valeur possible. La figure 3.14 représente T avec les heures de service, d'arrivée, et les fenêtres de temps des arcs A_1 et A_2. Les temps d'attente sont représentés en gris. La valeur initiale de $Av_{A_2}(T)$ est égale à $h_{A_1} - l_{A_1} = 5$. A l'itération suivante (itération 2), nous avons $arr_{A_2} - l_{A_2} > Av_{A_2}(T)$. La valeur de $Av_{A_2}(T)$ est donc augmentée de ta_{A_2}. Elle est alors égale à 11. Nous pouvons donc avancer l'heure du service de A_2 de 11 minutes. La procédure ci-dessous décrit comment le faire.

Algorithme: AVANCER_SERVICE

ENTREE: Le réseau décrit en 3.2.1, une tournée $T = (A_1, ..., A_{|T|})$, un arc A_k desservi dans T et la valeur $Av_{A_k}(T)$.

SORTIE: La tournée T à l'intérieur de laquelle l'heure du service de A_k est avancée de $Av_{A_k}(T)$ minutes.

Etape 0. Si $k < |T|$ poser $ta_{A_{k+1}} := ta_{A_{k+1}} + Av_{A_k}(T)$. Poser $i := k$.

Etape 1. Si $ta_{A_i} > 0$ alors

Si $ta_{A_i} > Av_{A_k}(T)$ poser $ta_{A_i} := ta_{A_i} - Av_{A_k}(T)$ et $Av_{A_k}(T) := 0$
Sinon poser $ta_{A_i} := 0$ et $Av_{A_k}(T) := Av_{A_k}(T) - ta_{A_i}$

Si $Av_{A_k}(T) = 0$ STOP, sinon poser $i := i - 1$ et répéter l'étape 1.

En reprenant l'exemple ci-dessus, nous devons faire avancer l'heure du service de l'arc A_2 de 11 minutes dans $T = (A_1, A_2)$. La valeur de ta_{A_2} est positive mais inférieure à celle de $Av_{A_2}(T)$. Le temps d'attente en A_2 est donc supprimé ($ta_{A_2} = 0$) et la valeur de $Av_{A_2}(T)$ passe de 11 à 5 minutes (voir figure 3.15). La valeur de ta_{A_1} est supérieure à 5, $Av_{A_2}(T)$ devient nul et ta_{A_1} passe de 10 à 5 minutes. Nous nous trouvons alors dans la situation décrite à la figure 3.16. Le début du service de l'arc A_2 de T a été retardé de 11 minutes tout en préservant l'admissibilité de T.

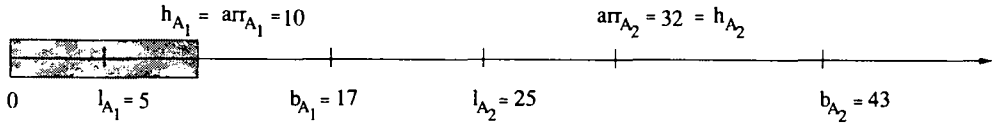


FIG. 3.15 -

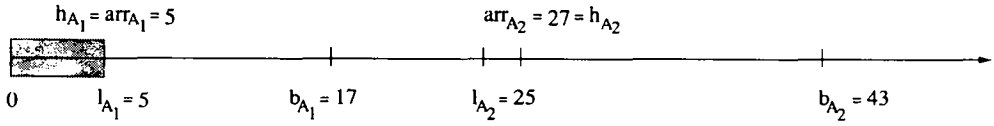


FIG. 3.16 -

Les termes $Ret_{A'_q}(T')$ et $Av_{A'_q}(T')$ sont définis de façon semblable aux termes $Ret_{A_k}(T)$ et $Av_{A_k}(T)$. L'utilisateur a alors la possibilité de fixer l'heure du service de A_k dans l'intervalle $[h_{A_k} - Av_{A_k}(T), h_{A_k} + Ret_{A_k}(T)]$ et l'heure du début du service de A'_q dans l'intervalle $[h_{A'_q} - Av_{A'_q}(T'), h_{A'_q} + Ret_{A'_q}(T')]$. Il peut ainsi tenter d'augmenter l'écart entre l'heure de parcours de $Traj_{red}$ dans T et T'.

Nous avons présenté un cas où deux tournées effectuaient pratiquement en même temps un trajet identique. Il peut exister des situations dans lesquelles plus de deux tournées parcourent le même trajet redondant. L'utilisateur peut alors modifier les heures de passage sur ce trajet de chacune des tournées en utilisant la même technique que celle que nous venons de présenter.
Afin de repérer les trajets redondants, l'utilisateur doit entrer une valeur t_{seuil}. La liste des tournées impliquées dans tous les trajets redondants s'effectuant à moins de t_{seuil} minutes d'intervalle est affichée. L'utilisateur choisit ensuite la tournée qu'il souhaite modifier. Une indication sur sa marge de manoeuvre (intervalle de temps sur lequel il peut jouer pour que la tournée reste admissible) lui est donnée.

Le fait de décaler l'heure de service d'un arc dans une tournée peut permettre de supprimer des trajets redondants, mais cela peut également en créer de nouveaux. Après chaque modification temporelle effectuée par l'utilisateur, la liste des tournées engendrant des trajets redondants est donc remise à jour. Afin d'éviter que lors de l'application successive de modifications temporelles l'utilisateur ne tourne en rond (il pourrait par exemple retarder l'heure d'un trajet qu'il avait avancé précédemment), des modifications sont effectuées sur certains arcs de la tournée qui vient d'être traitée. La nature de ces modifications varie suivant que le trajet redondant est desservi ou non dans la tournée:

a) Si le trajet redondant est représenté par un arc desservi:
La borne supérieure de la fenêtre de temps associée à cet arc est modifiée. Elle est égale à l'heure de départ du trajet si cette dernière a été avancée. De cette façon, il n'est plus possible de retarder ce trajet.
Lorsque le trajet est retardé, la borne inférieure de la fenêtre de temps qui lui est associée est modifiée. Elle est alors égale à l'heure du début du trajet. Celui-ci ne peut alors plus être avancé.

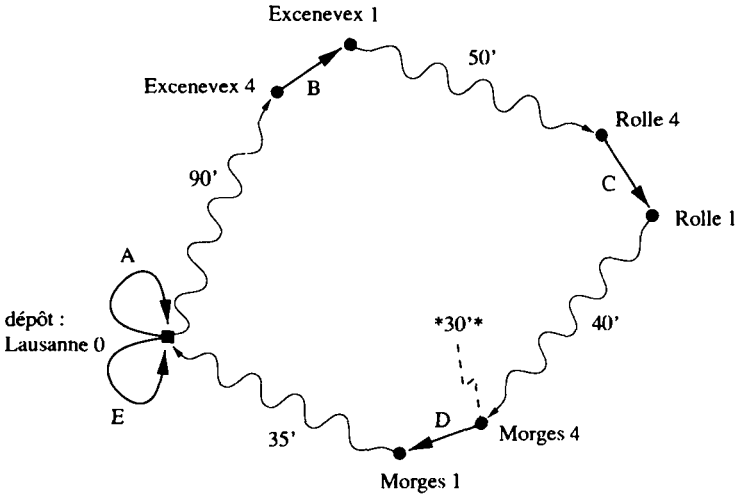
b) Si le trajet redondant est représenté par un arc non desservi:
Le statut de l'arc correspondant au trajet est modifié. Il devient un arc desservi dans la tournée auquel une fenêtre de temps est associée. Cette fenêtre de temps est: [5h00, heure de départ du trajet] si l'heure du début du trajet a été avancée. Si le trajet a été retardé, la fenêtre de temps est: [heure de début du trajet, 23h30].

L'exemple ci-dessous décrit la façon dont s'effectue une modification temporelle.

Exemple:

Nous allons considérer la tournée $T = (A, B, C, D, E)$ déjà décrite en 3.3.2.2. Elle est représentée à la figure 3.17 avec un tableau contenant les arcs qu'elle dessert. Un deuxième tableau (tableau 3.8) contient le détail de tous les trajets effectués au cours de la tournée. Nous allons supposer que le trajet marqué en gras dans ce tableau (trajet Yvoire 3 - Rolle 4) est un trajet redondant qu'il faut décaler d'une demi-heure au moins. Ce trajet n'est pas desservi dans la tournée, l'arc B est l'arc desservi qui le précède immédiatement. Dans les calculs que nous effectuons, les heures de départ, les bornes des fenêtres de temps et les temps d'attente sont exprimés en minutes. Nous pouvons calculer la valeur de $Ret_B(T)$ ainsi que celle de $Av_B(T)$.

$$\begin{aligned} Ret_B(T) &= \min\{0 + b_C - h_C, 30 + b_D - h_D, 30 + b_E - h_E, b_B - h_B\} \\ &= \min\{720 - 680, 30 + 990 - 750, 30 + 1410 - 785, 720 - 630\} = 40 \text{ minutes} \\ Av_B(T) &= 90 \text{ minutes (valeur fournie par CALCULER\_AVANCE)} \end{aligned}$$



Arc	Fenêtre de temps		début du
	borne inf.	borne sup.	service à
A	5h00	23h30	9h00
B	9h00	12h00	10h30
C	9h00	12h00	11h20
D	12h30	16h30	12h30
E	5h00	23h30	13h05

FIG. 3.17 --

Tournée $T = (A, B, C, D, E)$.

	départ			arrivée
540	9h00	Lausanne 0	$\xrightarrow{A}$	9h00	Lausanne 0
	9h00	Lausanne 0	$\longrightarrow$	9h00	Lausanne 3
	9h00	Lausanne 3	$\longrightarrow$	9h55	Thonon 0
	9h55	Thonon 0	$\longrightarrow$	9h55	Thonon 1
	9h55	Thonon 1	$\longrightarrow$	10h15	Margencel 4
	10h15	Margencel 4	$\longrightarrow$	10h15	Margencel 1
	10h15	Margencel 1	$\longrightarrow$	10h20	Sciez 4
	10h20	Sciez 4	$\longrightarrow$	10h20	Sciez 1
	10h20	Sciez 1	$\longrightarrow$	10h30	Excenevex 4
	10h30	Excenevex 4	$\xrightarrow{B}$	10h30	Excenevex 1
	10h30	Excenevex 1	$\longrightarrow$	10h50	Yvoire 0
	10h50	Yvoire 0	$\longrightarrow$	10h50	Yvoire 3
	10h50	**Yvoire 3**	$\longrightarrow$	**11h20**	**Rolle 4**
	11h20	Rolle 4	$\xrightarrow{C}$	11h20	Rolle 1
	11h20	Rolle 1	$\longrightarrow$	12h00	Morges 4
30	12h30	Morges 4	$\xrightarrow{D}$	12h30	Morges 1
	12h30	Morges 1	$\longrightarrow$	13h05	Lausanne 0
	13h05	Lausanne 0	$\xrightarrow{E}$	13h05	Lausanne 0

TAB. 3.8 -

Ensemble des trajets de T.

Il est possible de servir l'arc B entre 9h00 ($= h_B - Av_B(T)$) et 11h10 ($= h_B + Ret_B(T)$). L'horaire de la tournée T que nous obtenons en avançant l'heure de service de B de 30 minutes est présenté dans le tableau 3.9. Un temps d'attente de 30 minutes est introduit sur le premier arc obligatoire desservi suivant immédiatement le trajet redondant (arc C) afin de conserver l'admissibilité de la tournée. Le trajet (Yvoire 3 - Rolle 4) est maintenant effectué à 10h20 (au lieu de 10h50).

	départ			arrivée
510	8h30	Lausanne 0	$\xrightarrow{A}$	8h30	Lausanne 0
	8h30	Lausanne 0	$\longrightarrow$	8h30	Lausanne 3
	8h30	Lausanne 3	$\longrightarrow$	9h25	Thonon 0
	9h25	Thonon 0	$\longrightarrow$	9h25	Thonon 1
	9h25	Thonon 1	$\longrightarrow$	9h45	Margencel 4
	9h45	Margencel 4	$\longrightarrow$	9h45	Margencel 1
	9h45	Margencel 1	$\longrightarrow$	9h50	Sciez 4
	9h50	Sciez 4	$\longrightarrow$	9h50	Sciez 1
	9h50	Sciez 1	$\longrightarrow$	10h00	Excenevex 4
	10h00	Excenevex 4	$\xrightarrow{B}$	10h00	Excenevex 1
	10h00	Excenevex 1	$\longrightarrow$	10h20	Yvoire 0
	10h20	Yvoire 0	$\longrightarrow$	10h20	Yvoire 3
	10h20	**Yvoire 3**	$\longrightarrow$	**10h50**	**Rolle 4**
30	11h20	Rolle 4	$\xrightarrow{C}$	11h20	Rolle 1
	11h20	Rolle 1	$\longrightarrow$	12h00	Morges 4
30	12h30	Morges 4	$\xrightarrow{D}$	12h30	Morges 1
	12h30	Morges 1	$\longrightarrow$	13h05	Lausanne 0
	13h05	Lausanne 0	$\xrightarrow{E}$	13h05	Lausanne 0

TAB. 3.9 -

Tournée T obtenue en avançant le début du service de l'arc B de 30 minutes.

Pour déplacer l'heure de service du trajet (Yvoire 3 - Rolle 4), nous pouvons également retarder l'heure de service de l'arc B d'une demi-heure. L'horaire ainsi obtenu est présenté dans le tableau 3.10.

	départ			arrivée
540	9h00	Lausanne 0	$\xrightarrow{A}$	9h00	Lausanne 0
	9h00	Lausanne 0	$\longrightarrow$	9h00	Lausanne 3
	9h00	Lausanne 3	$\longrightarrow$	9h55	Thonon 0
	9h55	Thonon 0	$\longrightarrow$	9h55	Thonon 1
	9h55	Thonon 1	$\longrightarrow$	10h15	Margencel 4
	10h15	Margencel 4	$\longrightarrow$	10h15	Margencel 1
	10h15	Margencel 1	$\longrightarrow$	10h20	Sciez 4
	10h20	Sciez 4	$\longrightarrow$	10h20	Sciez 1
	10h20	Sciez 1	$\longrightarrow$	10h30	Excenevex 4
30	11h00	Excenevex 4	$\xrightarrow{B}$	11h00	Excenevex 1
	11h00	Excenevex 1	$\longrightarrow$	11h20	Yvoire 0
	11h20	Yvoire 0	$\longrightarrow$	11h20	Yvoire 3
	11h20	**Yvoire 3**	$\longrightarrow$	**11h50**	**Rolle 4**
	11h50	Rolle 4	$\xrightarrow{C}$	11h50	Rolle 1
	11h50	Rolle 1	$\longrightarrow$	12h30	Morges 4
	12h30	Morges 4	$\xrightarrow{D}$	12h30	Morges 1
	12h30	Morges 1	$\longrightarrow$	13h05	Lausanne 0
	13h05	Lausanne 0	$\xrightarrow{E}$	13h05	Lausanne 0

TAB. 3.10 -

Tournée T obtenue en retardant le début du service de l'arc B de 30 minutes.

Le début du service de l'arc B est retardé par l'introduction d'un temps d'attente de 30 minutes. Le trajet (Yvoire 3 - Rolle 4) s'effectue ainsi une demi-heure plus tard, ce qui a pour cause de supprimer le temps d'attente qui existait avant le service de l'arc C.

L'itinéraire parcouru par une tournée n'est pas modifié lorsque nous effectuons des décalages temporels. Cela permet un bon contrôle des modifications effectuées. En revanche, ce type de modifications est souvent insuffisant pour permettre de régler tous les problèmes liés à l'existence de trajets redondants.

3.3.3.1.2 Modifications des itinéraires des tournées

Contrairement à ce qui se passe lorsque des modifications temporelles sont effectuées, la méthode que nous allons décrire ici va permettre de modifier l'itinéraire des tournées. L'objectif est toujours de diminuer le nombre de trajets redondants. Le principe sur lequel repose cette méthode consiste à ôter des tournées un ensemble d'arcs desservis liés aux trajets redondants. On essaye ensuite de réinsérer ces arcs dans les tournées, en espérant qu'en effectuant cette opération un certain nombre de trajets redondants disparaîtront.

Soit $T = (A_1, ..., A_{|T|})$ une tournée et soit $Traj_{red}$ un trajet redondant appartenant à T. Un ou deux arcs desservis dans T sont supprimés afin d'éliminer $Traj_{red}$ de T. Les arcs qui sont ôtés de T sont différents suivant que $Traj_{red}$ est desservi ou non dans T:

a) si $Traj_{red}$ est desservi dans T, alors seul l'arc correspondant à $Traj_{red}$ dans T est supprimé.

b) si $Traj_{red}$ n'est pas desservi dans T, les deux arcs desservis dans T qui suivent et qui précèdent immédiatement $Traj_{red}$ sont ôtés de T.

La suppression d'un arc A_k de T se fait en remplaçant par un plus court chemin le chemin reliant dans T l'extrémité de l'arc A_{k-1} à l'origine de l'arc A_{k+1}.

Exemple:

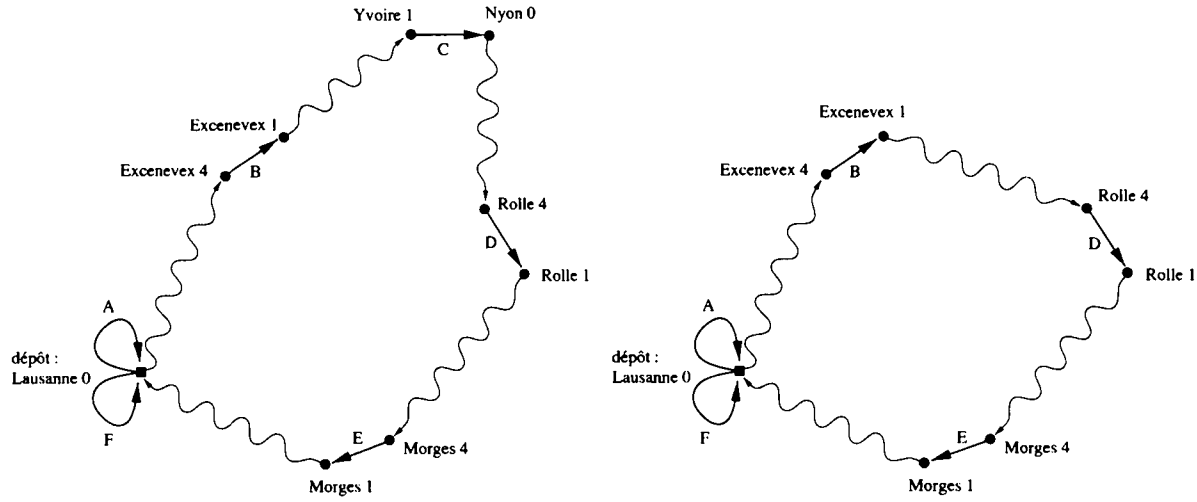


FIG. 3.18 –

Suppression de l'arc C.

Pour supprimer l'arc C, le chemin de la tournée entre Excenevex 1 et Rolle 4 est remplacé par un plus court chemin reliant ces mêmes sommets. La figure 3.18 représente la tournée avant et après la suppression de C.
La figure 3.19 illustre la suppression d'un trajet redondant entre les ports de Nyon et de Rolle. Ce trajet est l'arc non desservi G, joignant Nyon 1 à Rolle 4. Pour faire disparaître ce trajet, les arcs C et D sont ôtés de la tournée.

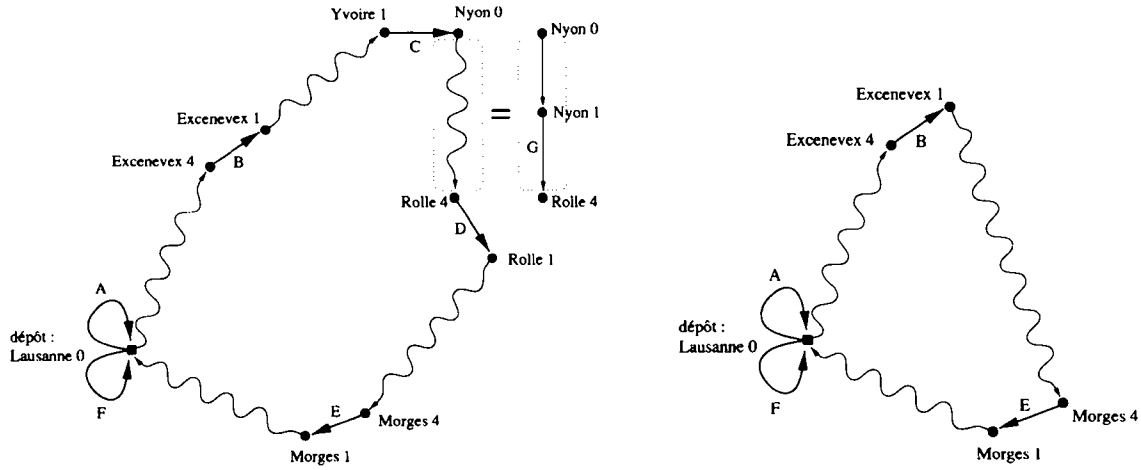


FIG. 3.19 –

Suppression du trajet redondant G.

Pour réinsérer les arcs, nous utilisons la méthode d'insertion parallèle décrite en 3.3.2.5. Cela ne va pas sans poser quelques problèmes. Tout comme dans le cas des modifications temporelles, il est possible qu'après avoir réinséré tous les arcs, de nouveaux trajets redondants apparaissent ou que les anciens n'aient pas pu être supprimés. Un autre inconvénient aux conséquences bien plus graves vient du fait que tous les arcs supprimés ne peuvent pas forcément être réinsérés. Nous nous retrouvons alors avec un horaire partiel dans lequel certains services ne sont pas effectués.
Malgré ces problèmes, la méthode que nous venons de décrire permet à l'utilisateur de traiter plusieurs trajets redondants à la fois. Elle permet également d'apporter à l'horaire des modifications plus profondes que de simples modifications temporelles.

3.3.4 Variables définies par l'utilisateur

Avant le début de chacune des méthodes d'insertion, l'utilisateur doit entrer le nombre de bateaux disponibles pour effectuer les tournées. Nous noterons ce nombre par n. Il doit également indiquer la manière dont les bateaux sont répartis dans les ports de Genève, Yvoire, Le Bouveret et Lausanne. Pour définir cette répartition, il lui suffit d'indiquer le nombre de bateaux basés à Genève et le nombre de bateaux se trouvant à Yvoire. Le nombre de bateaux stationnant au Bouveret est connu. En effet, la capacité de ce port est d'une unité et il faut obligatoirement un bateau au départ du Bouveret pour pouvoir effectuer la transversale Bouveret-Genève-Bouveret. Le nombre de bateaux se trouvant à Lausanne est égal au nombre de bateaux qui ne sont pas encore affectés à un port. Nous noterons n_L, n_G, n_Y et n_B le nombre de bateaux basés à Lausanne, à Genève, à Yvoire et au Bouveret respectivement.
Finalement, l'utilisateur doit définir la valeur d'un germe pour le générateur de nombres aléatoires. Ce générateur est utilisé pour choisir aléatoirement un ordre des tournées (dans INSERTION_SEQUENTIELLE) et pour choisir un arc à insérer lorsque plusieurs arcs ont un coût d'insertion équivalent. Une fois toutes ces valeurs introduites, une heuristique d'insertion est appliquée, puis, si une solution admissible est trouvée, l'utilisateur peut la modifier à l'aide des procédures de post-optimisation.

Pour un nombre de bateaux fixé, il faut parfois tester plusieurs germes différents ainsi que plusieurs répartitions des bateaux dans les ports avant d'obtenir une solution admissible. L'utilisateur a la possibilité de tester différents germes et diverses répartitions de manière automatique. Pour un nombre de bateaux fixé, et pour un germe donné, toutes les répartitions possibles des bateaux dans les ports (compte tenu de leur capacité d'accueil) sont envisagées. Si aucune solution admissible n'est trouvée, ce procédé est répété avec une autre valeur du germe. Il est interrompu dès qu'une solution admissible est trouvée ou lorsqu'un certain nombre de germes différents ont été testés sans succès. Ce nombre est de 20 pour l'algorithme INSERTION_PARALLELE et de 100 pour INSERTION_SEQUENTIELLE.

3.4 Les horaires

Nous présentons dans cette section les horaires que nous avons obtenus avec l'aide de la modélisation et des techniques de résolution présentées en 3.2 et en 3.3. Plusieurs tests ont été effectués en faisant varier le choix des fenêtres de temps. Une comparaison avec un horaire actuellement en vigueur sera également présentée.

3.4.1 Les fenêtres de temps

Nous appellerons fenêtres de temps standard les fenêtres de temps définies aux sections 3.1.3.2 et 3.1.3.3. Elles ont pour but de répartir tout au long d'une journée la visite des différents ports ainsi que le service accompli sur certains trajets. La manière dont elles ont été définies reste cependant arbitraire, c'est pourquoi nous avons également testé d'autres fenêtres de temps.

3.4.1.1 Fenêtres de temps relâchées

Afin de voir si les fenêtres de temps standard ne sont pas trop contraignantes, nous allons travailler avec des fenêtres de temps beaucoup plus souples. Ces fenêtres de temps sont définies de la manière la moins restrictive possible. Ainsi les visites de chaque port (dans une direction donnée) peuvent avoir lieu entre 9h00 et 19h30 et les trajets obligatoires doivent débuter entre 5h00 et 23h30. L'initialisation des tournées n'est pas concernée par ces modifications, seules les fenêtres de temps des arcs obligatoires du réseau défini en 3.2 sont modifiées.

3.4.1.2 Fenêtres de temps strictes

Afin de tenter de réduire le nombre de trajets redondants, des fenêtres de temps plus strictes (sans chevauchement) que les fenêtres de temps standard sont introduites. Elles concernent les visites devant être effectuées dans les ports ainsi que les trajets imposés entre Le Bouveret et Montreux et entre Le Bouveret et St-Gingolph. Les autres trajets obligatoires ont les mêmes fenêtres de temps que celles définies en 3.1.3.3. Les tableaux 3.11 et 3.12 contiennent les valeurs des fenêtres de temps strictes.

4 × par jour	3 × par jour	2 × par jour	1 × par jour
9h00 - 11h30	9h00 - 12h30	9h00 - 14h00	9h00 - 19h30
11h30 - 14h00	12h30 - 16h00	14h00 - 19h30	
14h00 - 16h30	16h00 - 19h30		
16h30 - 19h30			

TAB. 3.11 –

Heures de passages dans les ports.

Le tableau 3.11 contient les fenêtres de temps à l'intérieur desquelles les visites imposées (dans les sens positif et négatif) des différents ports peuvent avoir lieu. Ces fenêtres de temps varient en fonction du nombre de passages exigés dans chaque port.

4 traversées Montreux ⟶ Le Bouveret	
9h00 - 11h30	1×
11h30 - 14h00	1×
14h00 - 16h30	1×
16h30 - 19h30	1×

4 traversées Le Bouveret ⟶ Montreux	
9h00 - 11h30	1×
11h30 - 14h00	1×
14h00 - 16h30	1×
16h30 - 19h30	1×

4 liaisons directes St-Gingolph ⟶ Le Bouveret	
9h00 - 11h30	1×
11h30 - 14h00	1×
14h00 - 16h30	1×
16h30 - 19h30	1×

4 liaisons directes Le Bouveret ⟶ St-Gingolph	
9h00 - 11h30	1×
11h30 - 14h00	1×
14h00 - 16h30	1×
16h30 - 19h30	1×

TAB. 3.12 –

Fenêtres de temps strictes sur les trajets Le Bouveret Montreux et Le Bouveret St-Gingolph.

Comme dans le cas des fenêtres de temps relâchées, les arcs obligatoires introduits lors de l'initialisation des tournées conservent les fenêtres de temps que nous avons définies en 3.3.1.

3.4.2 Résultats

Nous présentons dans le tableau 3.13 les résultats obtenus à l'aide des méthodes d'insertion séquentielle et parallèle. Nous avons appliqué chaque méthode en utilisant les trois types de fenêtre de temps que nous venons de définir (standard, relâchées et strictes). Pour chacun des algorithmes, nous avons considéré plusieurs germes différents pour le générateur de nombres aléatoires. Pour une valeur fixée du germe, toutes les répartitions des bateaux dans les ports (compte tenu de leur capacité d'accueil) sont envisagées. Cent germes sont testés avec l'algorithme d'insertion séquentielle et vingt lorsque la méthode d'insertion parallèle est utilisée. Cela signifie que 800 solutions sont générées par INSERTION_SEQUENTIELLE et 160 par INSERTION_PARALLELE (il y a en effet 8 répartitions possibles des bateaux dans les ports).

Le tableau 3.13 contient un ensemble de valeurs moyennes qui sont calculées à partir de toutes les solutions admissibles générées lors de l'exécution de chaque algorithme. Le nombre de solutions admissibles trouvées par une heuristique est noté nb_{adm}, et $pour_{adm}$ représente le pourcentage de ces solutions par rapport au nombre total de solutions générées au cours d'une exécution. La valeur moyenne (en minutes) du temps total de parcours d'une solution admissible est notée $\overline{t}_{par}$, et celle du temps total d'attente (en minutes également) est désignée par $\overline{t}_{att}$. Rappelons que le temps total d'attente d'une solution admissible ne prend pas en compte le temps d'attente écoulé avant le premier trajet effectué par une tournée. La durée moyenne d'une tournée (en minutes) est représentée par la variable d_{moy}. Cette durée est égale à la somme de $\overline{t}_{par}$ et de $\overline{t}_{att}$ divisée par le nombre de bateaux utilisés n. La durée d'une tournée ne peut pas dépasser 13 heures (780 minutes). Le nombre moyen, dans une solution admissible, de trajets directs identiques effectués dans un laps de temps inférieur ou égal à t minutes est noté $\overline{nb}_{red}(t)$.

Le nombre de bateaux disponibles, n, est le nombre minimum de bateaux avec lequel une méthode a réussi à trouver une solution admissible. Si par exemple il faut 10 bateaux à INSERTION_PARALLELE lorsque nous utilisons des fenêtres de temps standard, cela signifie qu'aucune des 160 solutions générées en utilisant 9 bateaux n'est admissible.

Dans le tableau 3.13 figurent encore les caractéristiques de la solution admissible, rencontrée au cours de l'exécution d'un algorithme, qui a le temps de parcours le plus faible. Ce dernier est représenté par t^*_{par}, et le temps d'attente de cette solution est désigné par t^*_{att}. Ces temps sont donnés en minutes. Le nombre de trajets redondants effectués dans un intervalle d'au plus t minutes est noté $nb^*_{red}(t)$. Nous indiquons également la répartition des bateaux dans les ports associée à cette solution. Cette répartition est représentée à l'aide d'un vecteur (n_L, n_G, n_Y, n_B) où les variables n_L, n_G, n_Y et n_B correspondent au nombre de bateaux stationnant respectivement à Lausanne, Genève, Yvoire et au Bouveret. Finalement, nous indiquons aussi la valeur du germe qui a permis d'obtenir notre solution ainsi que le temps de calcul (en secondes) nécessaire pour la trouver lorsque le germe et la répartition des bateaux dans les ports sont connus. Ce temps ne prend donc pas en compte la durée nécessaire à la recherche du germe et de la répartition qui permettent d'aboutir à notre solution.

	INSERTION_SEQUENTIELLE			INSERTION_PARALLELE		
	Fenêtres de temps			Fenêtres de temps		
	relâchées	standard	strictes	relâchées	standard	strictes
n	9	10	11	9	10	10
nb_{adm}	526	4	262	155	5	4
$pour_{adm}$	65.75%	0.5%	32.75%	96.88%	3.13%	2.5%
$\bar{t}_{par}$	6646	7169	7620	6576	6985	7065
$\bar{t}_{att}$	0	0	17	0	4	30
$\bar{d}_{moy}$	738	717	694	731	699	710
$\overline{nb}_{red}(15)$	16.19	10.00	10.33	17.63	9.40	9.00
t^*_{par}	6225	7115	7015	6180	6870	6740
t^*_{att}	0	0	45	0	0	5
$nb^*_{red}(15)$	14	12	7	16	14	8
germe	25	12	48	16	12	5
répartition des bateaux	(5,3,01)	(6,2,1,1)	(6,3,1,1)	(6,2,0,1)	(7,2,0,1)	(7,2,0,1)
temps de calcul [s]	20	21	16	119	138	93

TAB. 3.13 -

Résultats obtenus avec différents types de fenêtres de temps.

Plus les contraintes temporelles sont strictes, plus la valeur de $\bar{t}_{par}$ augmente. Le nombre de bateaux utilisés croît lui aussi lorsque les fenêtres de temps deviennent plus restrictives. Même avec un nombre de bateaux en service inférieur, il est beaucoup plus facile de trouver des solutions admissibles avec les fenêtres de temps relâchées. Le seul inconvénient des fenêtres de temps relâchées est que la bonne répartition des visites d'un même port au cours d'une journée ne peut pas être véritablement contrôlée. Seule la manière dont nous avons modélisé les ports (demi-tours interdits dans la plupart des ports secondaires) permet d'étaler dans le temps les visites d'un même port. Bien qu'étant plus important en moyenne avec les fenêtres de temps relâchées, le nombre de trajets redondants reste du même ordre de grandeur qu'avec les autres types de fenêtre de temps. Par contre, comme la durée moyenne d'une tournée est plus importante avec les fenêtres de temps relâchées,

nous disposons d'une marge de manoeuvre plus faible pour tenter de supprimer les trajets redondants dans la phase de post-optimisation.

En utilisant des fenêtres de temps plus contraignantes (standard ou strictes), il devient plus difficile de trouver une solution admissible à l'aide des heuristiques d'insertion. Des bateaux supplémentaires sont nécessaires et il existe toujours en moyenne une dizaine de trajets redondants. Des temps d'attente apparaissent avec ce type de fenêtre de temps principalement lorsque celles-ci sont les plus restrictives. Ces temps d'attente restent néanmoins très faibles par rapport au temps total de parcours des tournées composant une solution.

La méthode d'insertion parallèle est plus efficace que la méthode d'insertion séquentielle. Quel que soit le type de fenêtre de temps utilisées, c'est elle qui, en moyenne, nécessite le moins de bateaux et possède le temps total de parcours le plus faible. Le pourcentage de solutions admissibles trouvées à l'aide de cette méthode est également plus important que celui obtenu en utilisant INSERTION_SEQUENTIELLE. En revanche, lorsque le germe du générateur de nombres aléatoires et la répartition des bateaux dans les ports sont donnés, l'heuristique d'insertion parallèle est 5 à 6 fois plus lente que la méthode d'insertion séquentielle.

3.4.3 Comparaison avec l'horaire actuel

Afin de se faire une idée de la qualité des solutions obtenues à l'aide des méthodes d'insertion décrites en 3.3, nous allons les appliquer sur un réseau construit à partir des prestations actuelles de la CGN. Ce réseau sera appelé réseau actuel par opposition au réseau standard décrit en 3.2.1. L'horaire sur lequel nous nous basons est celui de la période juillet-août 1999. Il est en vigueur le lundi, le mardi, le mercredi et le vendredi (l'horaire du jeudi est très légèrement différent). Le tableau 3.14 contient les services qui sont assurés lorsque cet horaire est en place. Entre parenthèses figurent, à titre indicatif, les services obligatoires définis en 3.1.3.2.

De façon générale, la côte suisse du lac située entre Lausanne et St-Gingolph est mieux desservie actuellement que ce qui est imposé en 3.1.3.2. Le réseau standard en revanche, impose un plus grand nombre de visites dans les ports situés sur la rive française du lac. Certains trajets qui sont effectués dans l'horaire actuel ne sont pas autorisés dans le réseau standard, il s'agit des liaisons directes: Villeneuve-St-Gingolph, St-Gingolph-Villeneuve, Villeneuve-Territet et Territet-Villeneuve.

Nous avons imposé en 3.2.1.5 que les visites du Bouveret dans le sens positif se fassent uniquement en reliant Le Bouveret à St-Gingolph. Les 6 liaisons effectuées dans ce sens actuellement satisfont cette contrainte. Une restriction similaire concerne les visites de St-Gingolph dans le sens négatif. Celles-ci ne peuvent se faire qu'en reliant St-Gingolph au Bouveret. Parmi les 5 visites de St-Gingolph effectuées actuellement dans cette direction, 4 seulement sont des liaisons St-Gingolph-Le Bouveret. La cinquième relie St-Gingolph à Villeneuve.

Nom du port	Nombre de départs par jour		
	Total	sens positif	sens négatif
Lausanne-Ouchy	25	6× (4×)	3× (4×)
Pully	10	4× (3×)	6× (3×)
Lutry	8	4× (3×)	4× (3×)
Cully	8	4× (3×)	4× (3×)
Rivaz/St-Saphorin	7	4× (2×)	3× (2×)
Vevey-**Plan CGN**	4	2× (2×)	2× (2×)
Vevey-Marché	15	9× (4×)	6× (4×)
Vevey-**La Tour**	9	6× (3×)	3× (3×)
Clarens	9	6× (3×)	3× (3×)
Montreux	15	6× (4×)	9× (4×)
Territet	10	5× (3×)	5× (3×)
Chillon	10	6× (3×)	4× (3×)
Villeneuve	12	7× (3×)	5× (3×)
Le Bouveret	11	6× (4× destination St-Gingolph)	5× (4×)
St-Gingolph	9	1× (4×)	5× (4× destination Le Bouveret)
Meillerie	2	1× (2×)	1× (2×)
Tourronde	2	1× (2×)	1× (2×)
Evian-les-Bains	20	4× (4×)	1× (4×)
Amphion-les-Bains	7	4× (2×)	3× (2×)
Thonon-les-Bains	7	3× (4×)	3× (4×)
Margencel/Anthy/Séchex	1	1× (2×)	0× (2×)
Sciez	1	1× (1×)	0× (1×)
Excenevex	1	1× (1×)	0× (1×)
Yvoire	13	5× (4×)	3× (4×)
Nernier	9	0× (3×)	4× (3×)
Chens-sur-Léman/Tougues	2	1× (2×)	1× (2×)
Hermance	3	1× (3×)	1× (3×)
Anières	2	1× (2×)	1× (2×)
Corsier	2	1× (2×)	1× (2×)
Genève	7 (4×)	(pas de direction)	
Bellevue	3	2× (2×)	1× (2×)
Versoix	7	4× (3×)	3× (3×)
Coppet	7	4× (3×)	3× (3×)
Céligny	3	2× (2×)	1× (2×)
Nyon	13	3× (4×)	6× (4×)
Rolle	6	3× (3×)	0× (3×)
St-Prex	6	3× (3×)	3× (3×)
Morges	6	3× (3×)	3× (3×)
St-Sulpice	6	3× (3×)	3× (3×)

TAB. 3.14 –

Visites des ports effectuées en semaine (sauf le jeudi) au cours de la période juillet-août 1999.

Au niveau des trajets imposés entre les ports, des différences apparaissent également entre l'horaire actuel et les exigences spécifiées en 3.1.3.3. Le tableau 3.15 permet de s'en faire une idée. Entre parenthèses figure le nombre de trajets imposés que nous avons définis en 3.1.3.3.

Les principales différences se rencontrent sur les trajets Nyon-Yvoire et Yvoire-Nyon. Actuellement, la plupart des liaisons entre ces deux ports se font en transitant par Nernier. La liaison Nernier-Nyon est effectuée 5 fois et le trajet Nyon-Nernier est parcouru 4 fois dans l'horaire actuel. Aucune obligation ne concerne ces deux trajets dans le réseau standard.

Liaison	Nombre
Lausanne ⟶ Evian	16 (13)
Evian ⟶ Lausanne	15 (12)
Nyon ⟶ Yvoire	3 (10)
Yvoire ⟶ Nyon	2 (10)
Montreux ⟶ Le Bouveret	2 (4)
Le Bouveret ⟶ Montreux	1 (4)
St-Gingolph ⟶ Le Bouveret	4 (4)
Le Bouveret ⟶ St-Gingolph	6 (4)

TAB. 3.15 –

Trajets effectués en semaine (sauf le jeudi) entre certains ports (période juillet-août 1999).

3.4.3.1 Construction du réseau actuel

La construction du réseau actuel se fait de manière similaire à celle du réseau standard. Simplement, au lieu de prendre en compte les trajets imposés et les visites obligatoires de ports décrits en 3.1.3.2 et 3.1.3.3, nous nous basons sur les contraintes contenues dans les tableaux 3.14 et 3.15. Ainsi, par exemple, le réseau actuel comporte 16 arcs obligatoires reliant le port de Lausanne à celui d'Evian alors que le nombre de ces arcs est de 13 dans le réseau standard. Les arcs Villeneuve-St-Gingolph, St-Gingolph-Villeneuve, Villeneuve-Territet et Territet-Villeneuve qui n'existent pas dans le réseau standard sont également introduits dans le réseau actuel. La modélisation des ports ne change pratiquement pas, seul le nombre d'arcs obligatoires internes à un port est modifié. L'exemple qui suit illustre la façon dont un port peut être modifié en passant du réseau standard au réseau actuel. La figure 3.20 représente la modélisation du port de Meillerie dans le réseau standard. Deux visites dans ce port sont obligatoires dans chacun des sens positif et négatif. Elles sont représentées par deux arcs en gras reliant les sommets 415 et 115 (sens positif) et les sommets 515 et 215 (sens négatif). Dans le réseau actuel, une seule visite dans chaque direction suffit (voir figure 3.21).

Exemple:

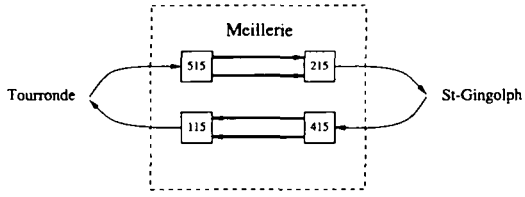


FIG. 3.20 –

Port de Meillerie dans le réseau standard.

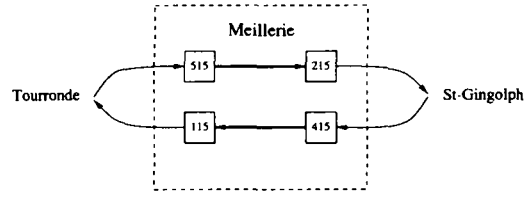


FIG. 3.21 –

Port de Meillerie dans le réseau actuel.

3.4.3.2 Les tournées effectuées actuellement

L'horaire de semaine (jeudi excepté) de la période juillet-août 1999 nécessite l'utilisation de 11 bateaux. Pour calculer le temps total de parcours de ces bateaux, nous avons

utilisé les temps de liaison entre les ports que nous employons dans notre programme. Rappelons que ces temps, fournis par la CGN, tiennent compte de la durée nécessaire au débarquement et à l'embarquement des passagers dans les ports (cette durée est estimée à 5 minutes). Ils sont donc légèrement plus longs que les temps de liaison figurant sur l'horaire officiel de la CGN qui, eux, ne prennent pas en compte le transfert des passagers dans les ports. Pour tenir compte de ce transfert, les horaires officiels de la CGN prévoient des temps d'attente dans les ports. Ces temps d'attente sont plus ou moins grands suivant l'importance du port visité. Ils peuvent varier pour un même port. Comme la durée nécessaire à l'embarquement et au débarquement des passagers est comprise dans les temps de parcours que nous utilisons, nous avons considéré des temps d'attente inférieurs de 5 minutes à ceux figurant dans l'horaire officiel (les temps d'attente inférieurs ou égaux à 5 minutes dans l'horaire officiel ne sont pas comptabilisés).
Parmi les 11 tournées qui constituent l'horaire actuel, six débutent et s'achèvent à Lausanne, une au Bouveret et deux à Genève. Parmi les deux tournées restantes, une débute à Lausanne et se termine à Genève et l'autre part de Genève et arrive à Lausanne. Tous les bateaux sont en service moins de 13 heures. Certaines tournées ne respectent pas les contraintes liées aux heures de début et de fin des trajets qui nous ont été imposées. Une tournée débute à 4h55, une autre se termine à 1h40. De plus, certaines visites obligatoires de port sont effectuées après 19h30.

3.4.3.3 Comparaison

Nous avons effectué une première série de tests en appliquant les heuristiques d'insertion au réseau actuel et en utilisant les fenêtres de temps relâchées décrites en 3.4.1.1. Comme nous l'avons fait à la section 3.4.2, 160 solutions sont générées lorsque INSERTION_PARALLELE est utilisé et 800 solutions avec INSERTION_SEQUENTIELLE. La solution admissible ayant le temps de parcours le plus faible est celle qui est fournie par nos algorithmes.

	Horaire actuel	INSERTION_SEQUENTIELLE			INSERTION_PARALLELE		
n	11	9	10	11	9	10	11
t_{par}	5810	6590	6310	6670	6565	6590	6560
t_{att}	640	0	0	0	0	0	0
d_{moy}	586	732	631	606	729	659	596
nb_{red}	1	25	21	15	18	28	22
germe		93	25	6	17	19	14
répartition des bateaux		(5,3,0,1)	(5,4,0,1)	(6,4,0,1)	(5,2,1,1)	(8,1,0,1)	(8,1,1,1)
$pour_{adm}$		1.9%	91.50%	100%	33.13%	99.38%	100%

TAB. 3.16 -

Comparaison entre les heuristiques d'insertion et l'horaire actuel.

Le tableau 3.16 contient les résultats que nous avons obtenus en appliquant nos méthodes d'insertion au réseau actuel. Il contient également les caractéristiques des tournées effectuées par la CGN en semaine (jeudi excepté) au cours de la période juillet-août 1999. Figurent dans ce tableau le nombre de bateaux utilisés, n, le temps total de parcours, t_{par}, le temps total d'attente, t_{att}, la durée moyenne d'une tournée, d_{moy} ainsi que le nombre de trajets directs identiques effectués dans un laps de temps inférieur ou égal à 15 minutes,

$nb_{red}(15)$. Le germe du générateur de nombres aléatoires, la répartition des bateaux dans les ports ainsi que le pourcentage, $pour_{adm}$, de solutions admissibles trouvées (par rapport au nombre de solutions générées) sont également mentionnés dans le tableau. Nous avons effectué des tests avec 9, 10 et 11 bateaux. Aucune solution admissible utilisant un nombre de bateaux inférieur à 9 n'a pu être trouvée à l'aide de nos algorithmes.

Nous avons pu obtenir des solutions admissibles qui utilisent moins de bateaux que ceux en service dans l'horaire actuel. Nos heuristiques trouvent facilement des solutions admissibles composées de 10 tournées. Ces dernières représentent en effet plus de 90% des solutions générées par nos algorithmes. Ceux-ci ont plus de difficultés à produire des solutions admissibles utilisant 9 bateaux.

Une des différences frappantes entre l'horaire actuel et ceux construits à l'aide des heuristiques d'insertion se situe au niveau du temps total d'attente. L'addition des temps d'attente apparaissant dans l'horaire actuel nous donne une durée totale supérieure à 10 heures. Cette durée est nulle avec les solutions fournies par nos heuristiques. Cette différence s'explique par le critère d'insertion que nous utilisons. Celui-ci a pour but de minimiser le décalage temporel provoqué par l'insertion d'un nouvel arc dans une tournée. Ce décalage est défini par le détour qu'il faut effectuer pour desservir l'arc inséré ainsi que par le temps d'attente qui est éventuellement engendré par cette insertion. Nos heuristiques ont ainsi tendance à éviter d'insérer un arc aux endroits où cela provoque la création de temps d'attente. Le fait d'avoir négligé les contraintes liées aux vidanges contribue également à l'obtention de temps d'attente de valeur nulle.

L'horaire actuel ne possède qu'un seul trajet redondant. Il s'agit de deux liaisons Evian-Lausanne effectuées à 5 minutes d'intervalle. Les solutions que fournissent nos heuristiques ont un nombre de trajets redondants bien plus élevé. Ce nombre peut toutefois être diminué lors de la phase de post-optimisation. Nous avons pu ainsi, dans le cas de la solution obtenue par la méthode d'insertion séquentielle utilisant 11 bateaux, éliminer tous les trajets redondants. La solution a alors un temps total de parcours de 6675 minutes et un temps total d'attente de 110 minutes.

Le temps de parcours de l'horaire actuel est inférieur à celui que nous obtenons avec les heuristiques d'insertion. La différence est importante, puisqu'elle varie entre 500 et 860 minutes. Plusieurs raisons peuvent expliquer cela. La première est que, dans l'horaire actuel, certaines visites imposées dans les ports ont lieu avant 9h00 et après 19h30. Cela n'arrive pas dans les solutions admissibles que nous générons car nous devons respecter les fenêtres de temps relâchées. Ainsi toutes les visites obligatoires de ports se font entre 9h00 et 19h30. Les solutions que nous construisons doivent ainsi satisfaire des contraintes plus strictes que celles qui sont effectivement respectées dans l'horaire actuel. Une deuxième explication peut provenir du fait que dans chacune de nos solutions, un bateau rentre au port d'où il est parti. Dans l'horaire actuel, cela n'est pas le cas, il existe deux bateaux qui terminent leur course dans des ports différents de leur port d'origine.

Afin de voir si les deux arguments que nous venons d'exposer sont valables, nous avons effectué une nouvelle série de tests. Ils ont été réalisés à partir du réseau actuel. Les fenêtres de temps associées aux visites obligatoires des ports ont été élargies. Ces visites doivent maintenant débuter entre 5h00 et 23h30. Afin de faciliter la comparaison avec

l'horaire actuel, seules des solutions utilisant 11 bateaux sont générées. De plus, les 11 tournées de chaque solution sont initialisées de manière à ce que les ports de départ et d'arrivée de chaque bateau soient identiques à ceux de l'horaire actuel. Les solutions que nous obtenons sont ainsi composées d'une transversale Genève - Le Bouveret - Genève, d'une transversale Le Bouveret - Genève - Le Bouveret, d'une tournée partant de Genève et s'achevant à Lausanne, d'une tournée quittant Lausanne et se terminant à Genève, de 6 tournées ayant Lausanne comme origine et comme destination et d'une tournée débutant et arrivant Genève. La répartition des bateaux dans les ports étant fixée, 100 solutions sont générées avec la méthode d'insertion séquentielle (en utilisant 100 valeurs différentes pour le germe) et 20 pour l'insertion parallèle. Le tableau 3.17 contient les résultats que nous avons obtenus.

	Horaire actuel	INSERTION_SEQUENTIELLE	INSERTION_PARALLELE
n	11	11	11
t_{par}	5810	6065	5965
t_{att}	640	0	0
d_{moy}	586	551	542
nb_{red}	1	15	22
germe		32	3
$pour_{adm}$		100%	100%

TAB. 3.17 -

Comparaison entre les heuristiques d'insertion et l'horaire actuel.

Nous pouvons constater que le temps total de parcours a diminué de manière significative et qu'il est maintenant voisin de celui de l'horaire actuel. En examinant les deux solutions que nous avons obtenues, nous avons remarqué que seuls 9 bateaux assurent des services, les deux qui restent ne font rien. La solution obtenue par INSERTION_SEQUENTIELLE est composée des deux transversales, d'une tournée partant de Lausanne et arrivant à Genève, d'une tournée quittant Genève et s'achevant à Lausanne, d'une tournée partant et revenant à Genève et de 5 tournées ayant Lausanne pour origine et pour destination. La solution fournie par INSERTION_PARALLELE est un peu différente. Les deux transversales sont toujours présentes. Parmi les 7 tournées restantes, 6 partent et reviennent à Lausanne et une débute et s'achève à Genève. Il n'existe pas de bateau partant de Genève et terminant sa course à Lausanne ni de bateau quittant Lausanne pour aboutir à Genève et cela malgré l'initialisation des tournées que nous avons effectuée.
Toutes les tournées de nos solutions débutent à 5h00 car tous les arcs obligatoires ont la même fenêtre de temps qui est [5h00,23h30]. Cela n'est pas réaliste du tout. En revanche, nous pouvons ainsi disposer d'une marge de manoeuvre importante pour supprimer une partie des trajets redondants en jouant par exemple sur l'heure à laquelle les tournées débutent. Le fait d'avoir élargi certaines fenêtres de temps rend plus facile la suppression des trajets redondants lors de la phase de post-optimisation. Nous avons ainsi pu les éliminer de la solution produite par INSERTION_PARALLELE. Le temps total de parcours de cette solution reste inchangé, en revanche le temps total d'attente passe alors de 0 à 170 minutes. De même, nous avons pu également supprimer de la solution obtenue à l'aide d' INSERTION_SEQUENTIELLE tous les trajets redondants effectués à 15 minutes d'intervalle ou moins. Le temps total de parcours de cette solution n'est pas modifié et son temps total d'attente vaut alors 95 minutes.

Les tournées composant l'horaire actuel et celles des solutions obtenues à l'aide des heuristiques d'insertion peuvent être très différentes. Nous avons comparé la transversale Genève-Le Bouveret-Genève de l'horaire actuel à celle de la solution obtenue par INSERTION_PARALLELE présentée dans le tableau 3.17.
Les figurent 3.22 et 3.23 représentent les liaisons effectuées au cours de cette transversale. Lorsqu'un trajet n'est effectué que dans un sens, il est dessiné en pointillés et une flèche indique la direction dans laquelle il est accompli. Les liaisons parcourues dans les deux sens sont représentées sans aucune flèche. Par rapport à la transversale de l'horaire actuel, celle que nous avons construite assure de nombreux services le long de la côte française du lac. Elle effectue également plusieurs circuits entre Le Bouveret et Montreux avant de repartir sur Genève. Sa durée est de 12 heures et 55 minutes, ce qui correspond, à 5 minutes près, à la durée maximum de service d'une tournée. Les trajets composant la transversale de l'horaire actuel sont beaucoup plus directs. Seuls les ports les plus importants de la côte française sont reliés. La durée de cette transversale est de 12 heures et 10 minutes.

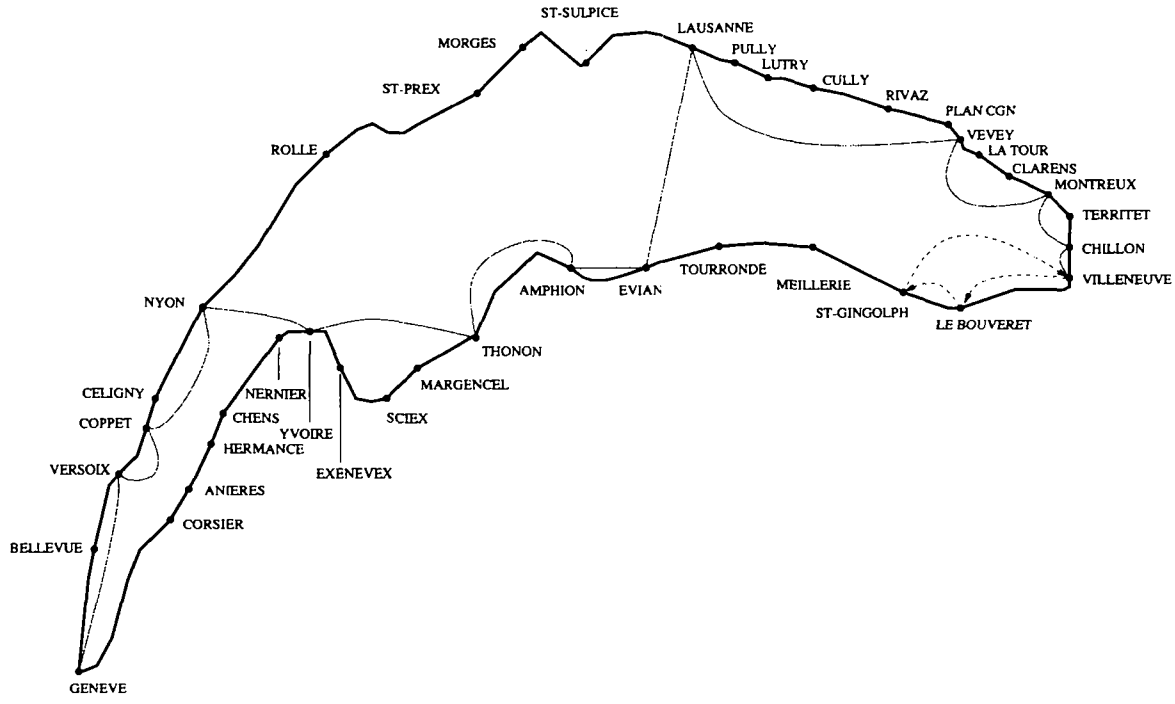


FIG. 3.22 --

Transversale Genève - Le Bouveret - Genève de l'horaire actuel.

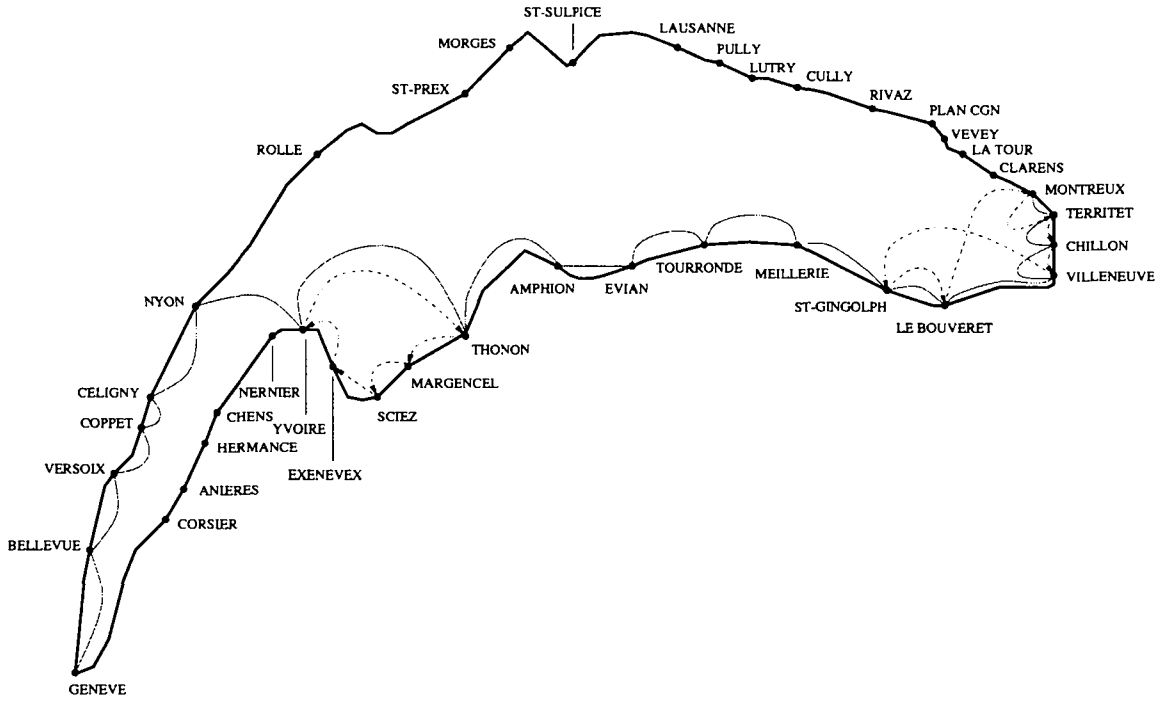


FIG. 3.23 -

Transversale Genève - Le Bouveret - Genève obtenue par INSERTION_PARALLELE.

3.4.4 Un exemple d'horaire

Nous présentons dans cette section de manière complète une solution que nous avons obtenue à l'aide de la méthode d'insertion parallèle. Il s'agit de la solution figurant dans le tableau 3.13 obtenue à partir du réseau standard. Les fenêtres de temps considérées sont les fenêtres de temps relâchées et le nombre de bateaux utilisés est égal à 9.

Les temps d'attente sont indiqués à la gauche du tableau entre les astérisques. Les trajets Genève-Genève correspondent aux éventuelles liaisons pouvant s'effectuer entre les débarcadères qui composent le port de Genève (voir 3.2).

	Tournée 1				
	départ				arrivée
475	7h55	Genève	⟶	8h15	Genève
	8h15	Genève	⟶	8h40	Versoix
	8h40	Versoix	⟶	9h00	Hermance
	9h00	Hermance	⟶	9h40	Genève

460

Tournée 2				
	départ			arrivée
7h40	Lausanne	→	8h20	Evian
8h20	Evian	→	9h00	Lausanne
9h00	Lausanne	→	9h20	St-Sulpice
9h20	St-Sulpice	→	9h35	St-Prex
9h35	St-Prex	→	10h15	Rolle
10h15	Rolle	→	10h45	Yvoire
10h45	Yvoire	→	11h15	Rolle
11h15	Rolle	→	11h40	St-Prex
11h40	St-Prex	→	11h55	Morges
11h55	Morges	→	12h30	Lausanne
12h30	Lausanne	→	12h50	St-Sulpice
12h50	St-Sulpice	→	13h05	Morges
13h05	Morges	→	13h20	St-Prex
13h20	St-Prex	→	13h45	Rolle
13h45	Rolle	→	14h25	Nyon
14h25	Nyon	→	14h50	Coppet
14h50	Coppet	→	15h05	Versoix
15h05	Versoix	→	15h30	Genève
15h30	Genève	→	15h50	Genève
15h50	Genève	→	16h15	Corsier
16h15	Corsier	→	16h20	Anières
16h20	Anières	→	16h30	Hermance
16h30	Hermance	→	17h05	Yvoire
17h05	Yvoire	→	17h35	Rolle
17h35	Rolle	→	18h00	St-Prex
18h00	St-Prex	→	18h15	Morges
18h15	Morges	→	18h30	St-Sulpice
18h30	St-Sulpice	→	18h50	Lausanne

380

Tournée 3				
	départ			arrivée
6h20	Lausanne	→	7h00	Evian
9h20	St-Sulpice	→	7h40	Lausanne
7h40	Lausanne	→	8h20	Evian
8h20	Evian	→	9h00	Morges
9h00	Morges	→	9h15	St-Sulpice
9h15	St-Sulpice	→	9h35	Lausanne
9h35	Lausanne	→	9h55	St-Sulpice
9h55	St-Sulpice	→	10h10	Morges
10h10	Morges	→	10h25	St-Prex
10h25	St-Prex	→	10h50	Rolle
10h50	Rolle	→	11h20	Yvoire
11h20	Yvoire	→	11h30	Nernier
11h30	Nernier	→	12h05	Coppet
12h05	Coppet	→	12h30	Nyon
12h30	Nyon	→	12h55	Chens
12h55	Chens	→	13h05	Hermance
13h05	Hermance	→	14h05	Chens
14h05	Chens	→	14h30	Nernier
14h30	Nernier	→	14h40	Yvoire
14h40	Yvoire	→	14h50	Nernier
14h50	Nernier	→	15h15	Chens
15h15	Chens	→	15h25	Hermance
15h25	Hermance	→	15h45	Versoix
15h45	Versoix	→	16h00	Coppet
16h00	Coppet	→	16h15	Hermance
16h15	Hermance	→	16h25	Chens
16h25	Chens	→	16h50	Nernier
16h50	Nernier	→	17h00	Yvoire
17h00	Yvoire	→	17h30	Rolle
17h30	Rolle	→	17h55	St-Prex
17h55	St-Prex	→	18h10	Morges
18h10	Morges	→	18h25	St-Sulpice
18h25	St-Sulpice	→	18h45	Lausanne

	Tournée 4				
	départ				arrivée
390	6h30	Lausanne	⟶	7h10	Evian
	7h10	Evian	⟶	7h50	Lausanne
	7h50	Lausanne	⟶	8h30	Evian
	8h30	Evian	⟶	8h45	Tourronde
	8h45	Tourronde	⟶	9h00	Meillerie
	9h00	Meillerie	⟶	9h25	St-Gingolph
	9h25	St-Gingolph	⟶	9h40	Le Bouveret
	9h40	Le Bouveret	⟶	10h05	Montreux
	10h05	Montreux	⟶	10h30	St-Gingolph
	10h30	St-Gingolph	⟶	10h45	Le Bouveret
	10h45	Le Bouveret	⟶	11h05	Villeneuve
	11h05	Villeneuve	⟶	11h15	Chillon
	11h15	Chillon	⟶	11h20	Territet
	11h20	Territet	⟶	11h30	Montreux
	11h30	Montreux	⟶	11h55	Vevey
	11h55	Vevey	⟶	12h00	La Tour
	12h00	La Tour	⟶	12h15	Clarens
	12h15	Clarens	⟶	12h25	Montreux
	12h25	Montreux	⟶	12h35	Clarens
	12h35	Clarens	⟶	12h50	La Tour
	12h50	La Tour	⟶	12h55	Vevey
	12h55	Vevey	⟶	13h00	La Tour
	13h00	La Tour	⟶	13h15	Clarens
	13h15	Clarens	⟶	13h25	Montreux
	13h25	Montreux	⟶	13h35	Clarens
	13h35	Clarens	⟶	13h50	La Tour
	13h50	La Tour	⟶	13h55	Vevey
	13h55	Vevey	⟶	14h20	Montreux
	14h20	Montreux	⟶	14h45	St-Gingolph
	14h45	St-Gingolph	⟶	15h00	Le Bouveret
	15h00	Le Bouveret	⟶	15h25	Montreux
	15h25	Montreux	⟶	15h50	Le Bouveret
	15h50	Le Bouveret	⟶	16h05	St-Gingolph
	16h05	St-Gingolph	⟶	16h20	Le Bouveret
	16h20	Le Bouveret	⟶	16h45	Montreux
	16h45	Montreux	⟶	17h10	Le Bouveret
	17h10	Le Bouveret	⟶	17h25	St-Gingolph
	17h25	St-Gingolph	⟶	17h40	Le Bouveret
	17h40	Le Bouveret	⟶	18h05	Montreux
	18h05	Montreux	⟶	18h15	Clarens
	18h15	Clarens	⟶	18h30	La Tour
	18h30	La Tour	⟶	18h35	Vevey
	18h35	Vevey	⟶	18h40	Plan CGN
	18h40	Plan CGN	⟶	18h50	Rivaz
	18h50	Rivaz	⟶	19h00	Cully
	19h00	Cully	⟶	19h15	Lutry
	19h15	Lutry	⟶	19h20	Pully
	19h20	Pully	⟶	19h30	Lausanne

400

Tournée 5				
	départ			arrivée
6h40	Lausanne	→	7h20	Evian
7h20	Evian	→	8h25	Yvoire
8h25	Yvoire	→	9h00	Hermance
9h00	Hermance	→	9h10	Anières
9h10	Anières	→	9h15	Corsier
9h15	Corsier	→	9h40	Genève
9h40	Genève	→	10h00	Genève
10h00	Genève	→	10h20	Bellevue
10h20	Bellevue	→	10h30	Versoix
10h30	Versoix	→	10h50	Hermance
10h50	Hermance	→	11h00	Anières
11h00	Anières	→	11h05	Corsier
11h05	Corsier	→	11h30	Genève
11h30	Genève	→	11h50	Genève
11h50	Genève	→	12h15	Corsier
12h15	Corsier	→	12h20	Anières
12h20	Anières	→	13h05	Yvoire
13h05	Yvoire	→	14h25	Lausanne
14h25	Lausanne	→	15h00	Morges
15h00	Morges	→	15h40	Evian
15h40	Evian	→	16h30	St-Gingolph
16h30	St-Gingolph	→	17h20	Evian
17h20	Evian	→	18h10	St-Gingolph
18h10	St-Gingolph	→	19h00	Evian
19h00	Evian	→	19h40	Lausanne

300

Tournée 6				
	départ			arrivée
5h00	Lausanne	→	5h40	Evian
5h40	Evian	→	6h20	Lausanne
6h20	Lausanne	→	7h00	Evian
7h00	Evian	→	7h40	Lausanne
7h40	Lausanne	→	8h20	Evian
8h20	Evian	→	9h00	Lausanne
9h00	Lausanne	→	9h40	Evian
9h40	Evian	→	10h10	Thonon
10h10	Thonon	→	10h40	Evian
10h40	Evian	→	11h00	Amphion
11h00	Amphion	→	11h20	Thonon
11h20	Thonon	→	11h55	Yvoire
11h55	Yvoire	→	12h30	Thonon
12h30	Thonon	→	12h50	Margencel
12h50	Margencel	→	13h15	Yvoire
13h15	Yvoire	→	13h25	Nernier
13h25	Nernier	→	14h00	Coppet
14h00	Coppet	→	14h15	Céligny
14h15	Céligny	→	14h30	Nyon
14h30	Nyon	→	14h45	Nernier
14h45	Nernier	→	14h55	Yvoire
14h55	Yvoire	→	15h15	Nyon
15h15	Nyon	→	15h35	Yvoire
15h35	Yvoire	→	15h55	Nyon
15h55	Nyon	→	16h15	Yvoire
16h15	Yvoire	→	16h50	Thonon
16h50	Thonon	→	17h20	Evian
17h20	Evian	→	18h00	Lausanne

420

Tournée 7				
	départ			arrivée
7h00	Lausanne	→	7h40	Evian
7h40	Evian	→	8h20	Lausanne
8h20	Lausanne	→	9h00	Evian
9h00	Evian	→	9h15	Tourronde
9h15	Tourronde	→	9h30	Meillerie
9h30	Meillerie	→	9h55	St-Gingolph
9h55	St-Gingolph	→	10h10	Le Bouveret
10h10	Le Bouveret	→	10h35	Montreux
10h35	Montreux	→	10h45	Territet
10h45	Territet	→	10h50	Chillon
10h50	Chillon	→	11h00	Villeneuve
11h00	Villeneuve	→	11h20	Le Bouveret
11h20	Le Bouveret	→	11h40	Villeneuve
11h40	Villeneuve	→	11h50	Chillon
11h50	Chillon	→	11h55	Territet
11h55	Territet	→	12h05	Montreux
12h05	Montreux	→	12h15	Territet
12h15	Territet	→	12h20	Chillon
12h20	Chillon	→	12h30	Villeneuve
12h30	Villeneuve	→	12h50	Le Bouveret
12h50	Le Bouveret	→	13h10	Villeneuve
13h10	Villeneuve	→	13h20	Chillon
13h20	Chillon	→	13h25	Territet
13h25	Territet	→	13h35	Montreux
13h35	Montreux	→	13h45	Territet
13h45	Territet	→	13h50	Chillon
13h50	Chillon	→	14h00	Villeneuve
14h00	Villeneuve	→	14h20	Le Bouveret
14h20	Le Bouveret	→	14h35	St-Gingolph
14h35	St-Gingolph	→	15h00	Meilleric
15h00	Meillerie	→	15h15	Tourronde
15h15	Tourronde	→	15h30	Evian
15h30	Evian	→	16h10	Lausanne
16h10	Lausanne	→	17h00	Vevey
17h00	Vevey	→	17h50	Lausanne
17h50	Lausanne	→	18h00	Pully
18h00	Pully	→	18h05	Lutry
18h05	Lutry	→	18h20	Cully
18h20	Cully	→	18h45	Vevey
18h45	Vevey	→	19h10	Cully
19h10	Cully	→	19h25	Lutry
19h25	Lutry	→	19h30	Pully
19h30	Pully	→	19h40	Lausanne

	Tournée 8				
		départ			arrivée
365	6h05	Genève	→	6h25	Genève
	6h25	Genève	→	7h20	Nyon
	7h20	Nyon	→	7h40	Yvoire
	7h40	Yvoire	→	8h00	Nyon
	8h00	Nyon	→	8h20	Yvoire
	8h20	Yvoire	→	8h40	Nyon
	8h40	Nyon	→	9h00	Yvoire
	9h00	Yvoire	→	9h25	Margencel
	9h25	Margencel	→	9h45	Thonon
	9h45	Thonon	→	10h05	Amphion
	10h05	Amphion	→	10h25	Evian
	10h25	Evian	→	11h05	Lausanne
	11h05	Lausanne	→	11h15	Pully
	11h15	Pully	→	11h20	Lutry
	11h20	Lutry	→	11h35	Cully
	11h35	Cully	→	11h45	Rivaz
	11h45	Rivaz	→	11h55	Plan CGN
	11h55	Plan CGN	→	12h00	Vevey
	12h00	Vevey	→	12h05	Plan CGN
	12h05	Plan CGN	→	12h15	Rivaz
	12h15	Rivaz	→	12h25	Cully
	12h25	Cully	→	12h40	Lutry
	12h40	Lutry	→	12h45	Pully
	12h45	Pully	→	12h55	Lausanne
	12h55	Lausanne	→	13h05	Pully
	13h05	Pully	→	13h10	Lutry
	13h10	Lutry	→	13h25	Cully
	13h25	Cully	→	13h35	Rivaz
	13h35	Rivaz	→	13h45	Plan CGN
	13h45	Plan CGN	→	13h50	Vevey
	13h50	Vevey	→	13h55	La Tour
	13h55	La Tour	→	14h10	Clarens
	14h10	Clarens	→	14h20	Montreux
	14h20	Montreux	→	14h45	Le Bouveret
	14h45	Le Bouveret	→	15h00	St-Gingolph
	15h00	St-Gingolph	→	15h25	Meillerie
	15h25	Meillerie	→	15h40	Tourronde
	15h40	Tourronde	→	15h55	Evian
	15h55	Evian	→	16h15	Amphion
	16h15	Amphion	→	16h35	Thonon
	16h35	Thonon	→	16h55	Margencel
	16h55	Margencel	→	17h00	Sciez
	17h00	Sciez	→	17h10	Excenevex
	17h10	Excenevex	→	17h30	Yvoire
	17h30	Yvoire	→	17h50	Nyon
	17h50	Nyon	→	18h05	Céligny
	18h05	Céligny	→	18h20	Coppet
	18h20	Coppet	→	18h35	Versoix
	18h35	Versoix	→	18h45	Bellevue
	18h45	Bellevue	→	19h05	Genève

	Tournée 9				
	départ				arrivée
395	6h35	Le Bouveret	→	6h50	St-Gingolph
	6h50	St-Gingolph	→	7h40	Evian
	7h40	Evian	→	8h20	Lausanne
	8h20	Lausanne	→	9h00	Evian
	9h00	Evian	→	9h30	Thonon
	9h30	Thonon	→	10h05	Yvoire
	10h05	Yvoire	→	10h25	Nyon
	10h25	Nyon	→	10h45	Yvoire
	10h45	Yvoire	→	11h05	Nyon
	11h05	Nyon	→	11h25	Yvoire
	11h25	Yvoire	→	11h45	Nyon
	11h45	Nyon	→	12h05	Yvoire
	12h05	Yvoire	→	13h05	Nyon
	13h05	Nyon	→	13h20	Céligny
	13h20	Céligny	→	13h35	Coppet
	13h35	Coppet	→	13h50	Versoix
	13h50	Versoix	→	14h00	Bellevue
	14h00	Bellevue	→	14h20	Genève
	14h20	Genève	→	14h40	Genève
	14h40	Genève	→	15h00	Bellevue
	15h00	Bellevue	→	15h10	Versoix
	15h10	Versoix	→	15h25	Coppet
	15h25	Coppet	→	15h40	Céligny
	15h40	Céligny	→	15h55	Nyon
	15h55	Nyon	→	16h15	Yvoire
	16h15	Yvoire	→	16h35	Excenevex
	16h35	Excenevex	→	16h45	Sciez
	16h45	Sciez	→	16h50	Margencel
	16h50	Margencel	→	17h10	Thonon
	17h10	Thonon	→	17h30	Amphion
	17h30	Amphion	→	17h50	Evian
	17h50	Evian	→	18h05	Tourronde
	18h05	Tourronde	→	18h20	Meillerie
	18h20	Meillerie	→	18h45	St-Gingolph
	18h45	St-Gingolph	→	19h10	Montreux
	19h10	Montreux	→	19h35	Le Bouveret

3.5 Améliorations possibles

Les horaires que nous obtenons actuellement ne sont pas directement utilisables par la CGN. Certaines contraintes que nous avons jusqu'ici négligées doivent être prises en compte. Des améliorations doivent également être apportées afin d'étendre les modifications pouvant être effectuées par l'utilisateur sur les horaires. Nous présentons dans cette section quelques pistes permettant de tenir compte de ces considérations.

3.5.1 Le problème des vidanges

Les principales contraintes que nous avons négligées sont celles liées aux vidanges des réservoirs d'eaux usées des bateaux. Ces vidanges peuvent être introduites avant la construction des tournées lors de la phase d'initialisation (voir figure 3.24).

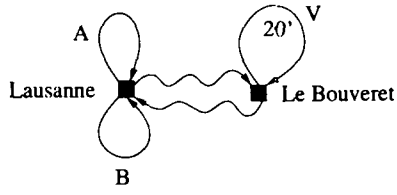


FIG. 3.24 -

Introduction d'une vidange au Bouveret dans une tournée débutant à Lausanne.

La figure 3.24 représente l'introduction d'une vidange lors de la phase d'initialisation d'une tournée. La tournée représentée débute à Lausanne. Les arcs A et B sont les arcs obligatoires fictifs introduits pour garantir que l'ensemble des trajets de la tournée s'effectuent entre 5h00 et 23h30 (voir section 3.3.1). L'arc V représente une vidange effectuée au Bouveret. Son temps de parcours de 20 minutes correspond à la durée maximum d'une vidange (cf tableau 3.1). Cet arc est obligatoire. Une fenêtre de temps peut lui être associée précisant l'heure à laquelle la vidange sera faite. Un plus court chemin entre Lausanne et Le Bouveret relie l'arc A à V et un plus court chemin entre Le Bouveret et Lausanne relie l'arc V à B. Une fois cette initialisation effectuée, les arcs sont insérés dans la tournée entre les arcs A et V ou entre les arcs V et B. L'utilisation de cette technique pour introduire les vidanges dans une tournée suppose que le nombre, le lieu et l'heure approximative des vidanges soient fixés pour chaque tournée avant même leur construction.

Les vidanges peuvent aussi être introduites une fois que toutes les tournées sont construites. Le fait que seuls certains ports disposent des installations nécessaires aux vidanges ne représente pas un problème. Toutes les tournées que nous avons générées passent par un de ces ports à un moment ou à un autre de la journée. En général, elles les visitent à plusieurs reprises. Les vidanges sont alors intégrées à une tournée en introduisant un temps d'attente correspondant à leur durée dans un des ports possédant l'équipement approprié et appartenant à la tournée. Cette solution n'est pas idéale. Le choix des ports et du moment où la vidange s'effectue est conditionné par l'itinéraire emprunté par la tournée. Celui-ci peut également limiter le nombre de vidanges pouvant avoir lieu. De plus, l'introduction d'un temps d'attente n'est pas toujours possible. Celui-ci ne doit pas provoquer

un dépassement de la durée maximum de service autorisée. Une solution pour éliminer ce dernier problème consiste à réduire la durée maximum de service de 20 minutes afin qu'une vidange au moins puisse être insérée dans chaque tournée. Cela ne résout cependant pas le problème des violations des fenêtres de temps qui peuvent être provoquées par l'introduction de vidanges dans une tournée déjà construite. Il faut alors, dans de tels cas, choisir le type de contraintes à privilégier: celles qui sont liées aux fenêtres de temps ou celles qui concernent les vidanges.

3.5.2 Le problème des trajets redondants

Les solutions obtenues à l'aide des heuristiques d'insertion ne sont pas totalement réalistes et nécessitent des modifications. Nous avons décrit à la section 3.3.3.1 des techniques permettant de réduire le nombre de trajets redondants apparaissant dans une solution. Cette phase de post-optimisation est faite manuellement par l'utilisateur. Elle peut prendre du temps, et il faut parfois compter sur la chance pour parvenir à réduire de manière significative le nombre de trajets redondants. Le développement d'un outil permettant d'effectuer automatiquement cette post-optimisation serait souhaitable. La possibilité offerte à l'utilisateur de modifier manuellement l'horaire serait néanmoins conservée. Des techniques différentes de celles proposées en 3.3.3.1 pourraient être également développées. Ces techniques pourraient consister, par exemple, à échanger des tronçons ayant mêmes extrémités et appartenant à des tournées différentes.

3.5.3 Equilibrage de la durée des tournées

Nous avons vu, dans l'exemple d'horaire que nous avons présenté, qu'une des tournées avait une durée inférieure à deux heures alors que plusieurs autres possédaient une durée égale à la durée maximum autorisée. Cela n'est pas concevable en pratique. Même s'il n'en a jamais été question jusqu'à présent, un certain équilibre sur la durée des tournées doit être respecté. La prise en compte de cet équilibre peut se faire lors de la construction des tournées, en privilégiant l'insertion de trajets obligatoires dans les tournées les moins longues. Elle peut également s'effectuer une fois l'ensemble des tournées construites en déplaçant un certain nombre de services des tournées les plus longues vers des tournées ayant une durée très faible.

3.5.4 Interface utilisateur

Actuellement, l'utilisateur dispose de deux moyens pour modifier un horaire. Il peut effectuer soit des décalages temporels en introduisant des temps d'attente dans les ports visités par les tournées, soit décider de modifier l'itinéraire des tournées. Dans cette deuxième situation, il n'a pratiquement aucun contrôle sur les nouveaux itinéraires qui peuvent être obtenus (voir 3.3.3.1). Des outils permettant à l'utilisateur de maîtriser l'ensemble des modifications apportées à un horaire doivent être développés. La gamme même des modifications possibles devrait aussi être étendue. Nous avons déjà mentionné en 3.5.2 la possibilité de pratiquer des échanges entre certains tronçons appartenant à des tournées différentes. Nous pourrions également offrir à l'utilisateur la possibilité de définir partiellement ou complètement l'itinéraire d'une tournée, de choisir son port de départ et son port d'arrivée, etc.
Le développement de tels outils va de pair avec une bonne visualisation des horaires et des

problèmes qui peuvent y apparaître. Il serait également souhaitable de pouvoir connaître à l'avance les conséquences des modifications d'un horaire avant que celles-ci soient validées.

3.6 Conclusion

La modélisation du problème de la conception des horaires de la CGN en termes de PTAC avec fenêtres de temps est bien adaptée. Elle permet de tenir compte des contraintes liées aux services minimum qui doivent être assurés, et offre la possibilité de contrôler la période de la journée durant laquelle ces services sont effectués. Elle a également l'avantage de pouvoir s'adapter facilement lorsque les exigences en matière de services sont modifiées.

Les premiers résultats que nous avons obtenus sont encourageants. Si les horaires que nous construisons ne sont pas, actuellement, directement utilisables par la CGN, ils peuvent néanmoins servir de base à l'élaboration d'un nouvel horaire.

Certaines contraintes que nous avons jusqu'à présent négligées doivent encore être prises en compte de manière à améliorer la qualité de nos horaires. Des développements doivent également être réalisés afin de fournir à l'utilisateur la possibilité de modifier l'ensemble des tournées. Dans cette optique, des outils permettant une bonne visualisation des horaires sont nécessaires. Ils permettront à l'utilisateur de saisir rapidement les problèmes à résoudre afin d'obtenir un horaire qui puisse être utilisé en pratique.

La mise en oeuvre des heuristiques d'insertion présentées dans ce chapitre a été effectuée par M. Frédéric Bardet lors de son travail pratique de diplôme [Bar99]. C'est suite à ce travail que la CGN a décidé de nous mandater afin de développer un logiciel d'aide à la décision pour la confection de ses horaires.

Chapitre 4

Optimisation d'itinéraires de câbles

4.1 Introduction

Lors de la construction d'une centrale nucléaire, de nombreux câbles doivent être tirés afin de relier deux à deux les divers équipements de la centrale. Ces câbles doivent passer à travers des éléments appelés "tablettes" et "trémies" qui sont de capacité limitée. A chaque câble est associée une section. La somme des sections des câbles traversant une tablette ou une trémie ne doit pas excéder sa capacité. Les coûts engendrés par l'achat et la pose des câbles sont très importants, d'où l'intérêt d'essayer de réduire autant que possible la longueur des chemins empruntés par les câbles. Le problème qui consiste à minimiser la longueur des itinéraires pris par les câbles tout en respectant les contraintes liées à la capacité des tablettes et des trémies sera appelé problème de plus courts chemins avec capacité (PPCCC).
Le PPCCC peut se modéliser à l'aide d'un graphe non orienté $G = (V, E)$ dont les sommets représentent les équipements à connecter et dont les arêtes correspondent aux tablettes (ou trémies). A chaque arête est associé un coût qui correspond au coût de liaison entre les équipements représentés par les extrémités de l'arête. On suppose que les coûts sont indépendants du type de câble utilisé. Des capacités correspondant à celles des tablettes et des trémies sont également associées aux arêtes. Le nombre de câbles à tracer, leur section ainsi que les équipements qu'ils doivent connecter sont connus.

Nous verrons que le PPCCC est un problème NP-dur. Nous allons nous intéresser dans ce chapitre au développement de méthodes heuristiques capables de produire de bonnes approximations de la solution optimale. Afin d'évaluer la qualité des solutions fournies par les heuristiques, nous allons étudier deux bornes inférieures obtenues à l'aide de techniques de relaxation lagrangienne à partir d'une formulation en termes de programmation linéaire 0_1 du PPCCC.

Le PPCCC trouve également des applications dans le domaine des télécommunications où il peut être vu comme un sous-problème du "Bandwidth Packing Problem" (BWP) introduit par Cox et al. [Cox91]. Nous présenterons les liens existant entre ces deux problèmes et décrirons une manière de résoudre le BWP en passant par la résolution du PPCCC.

4.2 Présentation du problème

4.2.1 Description et données du problème

Soit $G = (V, E)$ un graphe non orienté où $V = \{v_1, v_2, ..., v_n\}$ est l'ensemble des sommets et E l'ensemble des arêtes. A chaque arête (v_i, v_j) est associé un coût (ou longueur) c_{ij} non négatif et une capacité u_{ij} positive. Soit $C = \{(s_1, t_1), ..., (s_K, t_K)\}$ un ensemble contenant K paires de sommets de G. On suppose que s_k et t_k sont des sommets distincts de G $(k = 1, ..., K)$. A chaque paire de sommets (s_k, t_k) est associé un poids σ_k positif. Le couple de sommets (s_k, t_k) correspond aux extrémités que doit relier le câble k et le poids σ_k représente sa section. La section des câbles ainsi que la capacité des arêtes sont supposées entières.
Le graphe G, les coûts et les capacités associés aux arêtes de G, l'ensemble C ainsi que les poids σ_k, $k = 1, 2, ..., K$ constituent les données du problème du PPCCC. Ce problème consiste à déterminer les chemins des K câbles de telle sorte que:

a) chaque câble k relie s_k à t_k, $k = 1, 2, ..., K$,

b) la somme des sections des câbles traversant une arête (v_i, v_j) ne dépasse pas sa capacité u_{ij},

c) la longueur totale des K chemins tracés est minimum.

Exemple:

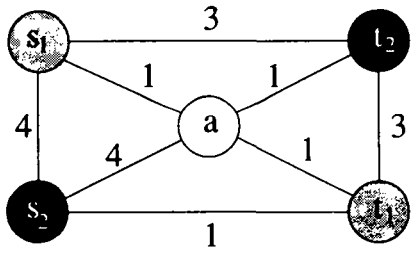


FIG. 4.1 -

Réseau G.

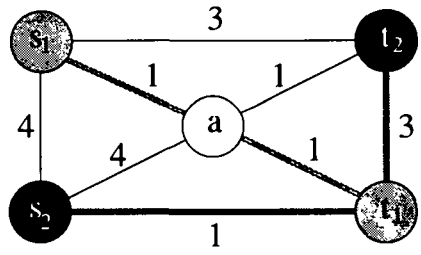


FIG. 4.2 -

Une solution du PPCCC dans G.

La figure 4.1 représente le graphe G dans lequel le PPCCC doit être résolu. Deux câbles doivent être tracés. Ces deux câbles ont une section de 1. La capacité de toutes les arêtes vaut 1 également. Les nombres situés au-dessus des arêtes correspondent à leur coût. La figure 4.2 représente une solution admissible du PPCCC dans G qui a un coût égal à 6. Le câble 1 emprunte le chemin $P_1 = (s_1, a, t_1)$ et le câble 2 suit le chemin $P_2 = (s_2, t_1, t_2)$.

Lorsque la capacité des arêtes et la section des câbles valent 1, le PPCCC revient alors à rechercher un ensemble de K chemins disjoints par les arêtes et de longueur totale minimum. On parle alors de problème de plus courts chemins disjoints par les arêtes (PPCCDA). Déterminer si oui ou non il existe une solution admissible (i.e. une solution satisfaisant **a)** et **b)**) pour le PPCCDA dans G est un problème NP-complet (voir [Mid93]

et [Vyg95]). Satisfaire les contraintes **a)**, **b)** et **c)** est au moins aussi difficile que de satisfaire seulement **a)** et **b)**, le PPCCDA est donc NP-dur. Le PPCCC est NP-dur également puisqu'il admet comme cas particulier le PPCCDA qui est NP-dur.

Le PPCCC est étroitement lié aux problèmes de multiflot compatible de coût minimum et à valeurs entières (voir [Min75]). En effet, nous pouvons faire correspondre à chaque câble k un type de flot k. La source et le puits de flot k sont respectivement les sommets s_k et t_k. La disponibilité en s_k est de σ_k unités de flot k. La demande en t_k est de σ_k également. Une unité de flot n'est pas divisible. On veut écouler les σ_k unités de flot k de s_k vers t_k et cela pour tous les types de flot k ($k = 1, ..., K$). La capacité u_{ij} d'une arête (v_i, v_j) définit le nombre maximum d'unités de flot qui peuvent traverser (v_i, v_j). Traverser une arête (v_i, v_j) avec une unité de flot k coûte alors $c_{ij}^k = \frac{c_{ij}}{\sigma_k}$. Le problème de multiflot que nous venons de définir ne correspond pas tout à fait au PPCCC. Il faut encore ajouter une contrainte supplémentaire qui oblige toutes les unités d'un même type de flot à parcourir le même chemin. Le PPCCC peut donc être vu comme un problème de multiflot compatible de coût minimum à valeurs entières et où toutes les unités d'un même type de flot suivent le même chemin. Lorsque toutes les paires de sommets de C sont identiques et que la section des câbles vaut 1, le PPCCC correspond alors à un problème de flot compatible de coût minimum et peut être résolu en temps polynomial.
Les problèmes de recherche d'un multiflot compatible de coût minimum font l'objet d'une abondante littérature. Le cas continu est le plus étudié (voir par exemple [Ken78], [Ahu93] pour des états de la recherche dans ce domaine). Il peut se résoudre à l'aide de l'algorithme du simplexe, mais pour des problèmes de taille importante, des algorithmes plus performants ont été développés (voir par exemple [McB97]). Trouver un multiflot compatible de coût minimum et à valeurs entières est un problème NP-dur. Des heuristiques ont été développées, notamment par Minoux [Min75] et par Aggarwal et al. [Agg95].

4.2.2 Formulation en termes de programmation linéaire 0_1

Pour décrire le PPCCC sous la forme d'un programme linéaire 0_1, nous allons introduire deux types de variables booléennes.

Pour chaque arête (v_i, v_j) de E et pour chaque paire (s_k, t_k) de C nous définissons une variable x_{ij}^k de la manière suivante:

$$x_{ij}^k = \begin{cases} 1 & \text{si le câble } k \text{ traverse l'arête } (v_i, v_j) \\ 0 & \text{sinon} \end{cases}$$

Comme le graphe G n'est pas orienté, nous avons $x_{ij}^k = x_{ji}^k$. Par la suite, nous ne considérerons donc que les variables x_{ij}^k avec $i < j$.

A chaque sommet v_i de V et à chaque paire (s_k, t_k) de C, on associe la variable y_i^k définie de la façon suivante:

$$y_i^k = \begin{cases} 1 & \text{si le câble } k \text{ passe par le sommet } v_i \\ 0 & \text{sinon} \end{cases}$$

En utilisant les deux types de variable que nous venons d'introduire, nous pouvons maintenant formuler le PPCCC à l'aide du programme linéaire suivant:

$$\textbf{MIN} \quad \sum_{k=1}^{K} \sum_{\substack{(v_i,v_j)\in E \\ i<j}} c_{ij}x_{ij}^k$$

$$\textbf{s.c.} \quad \sum_{k=1}^{K} \sigma_k x_{ij}^k \leq u_{ij} \quad \forall\,(v_i,v_j)\in E,\ i<j \qquad \textbf{(1)}$$

$$\sum_{\substack{(v_i,v_j)\in E \\ j<i}} x_{ji}^k + \sum_{\substack{(v_i,v_j)\in E \\ i<j}} x_{ij}^k = 2y_i^k \quad \forall\, k\in\{1,...,K\}, \forall\, v_i \neq s_k,\ t_k \qquad \textbf{(2)}$$

$$\sum_{\substack{(v_i,v_j)\in E \\ j<i}} x_{ji}^k + \sum_{\substack{(v_i,v_j)\in E \\ i<j}} x_{ij}^k = 1 \quad \forall\, k\in\{1,...,K\}, \forall\, v_i = s_k \text{ ou } t_k \qquad \textbf{(3)}$$

$$x_{ij}^k = 0 \text{ ou } 1 \quad \begin{array}{l}\forall\, k\in\{1,...,K\} \\ \forall\,(v_i,v_j)\in E,\ i<j\end{array} \qquad \textbf{(4)}$$

$$y_i^k = 0 \text{ ou } 1 \quad \forall\, k\in\{1,...,K\}, \forall\, v_i\in V \qquad \textbf{(5)}$$

Cinq types de contrainte apparaissent dans ce programme linéaire. Les contraintes de type **(1)** garantissent que la somme des sections des câbles est toujours inférieure ou égale à la capacité des arêtes traversées. Les contraintes de type **(2)** et **(3)** assurent que chaque câble k ($k = 1,...,K$) parcourt bien un chemin allant de s_k à t_k. Finalement, les contraintes de type **(4)** et **(5)** indiquent que les variables utilisées dans le programme linéaire sont des variables booléennes. Une solution admissible du PPCCC est définie comme étant une solution satisfaisant toutes les contraintes du programme linéaire ci-dessus. Lorsqu'une solution viole une de ces contraintes, on parlera alors de solution non admissible. Le programme linéaire ci-dessus sera appelé **PL**.

4.3 Bornes inférieures

Un des intérêts de la formulation en termes de programmation linéaire réside dans le fait qu'en relaxant certains types de contraintes, il est possible de calculer des bornes inférieures pour notre problème. Ces bornes sont importantes dans la mesure où elles permettent d'évaluer la qualité des solutions que nous allons obtenir. Nous avons étudié deux bornes inférieures différentes basées toutes deux sur la relaxation lagrangienne de certains types de contraintes. Pour plus de détails concernant les techniques de relaxation lagrangienne, le lecteur pourra se référer à [Fis81b].

4.3.1 Relaxation des contraintes de capacité

La première borne inférieure étudiée est obtenue en relaxant les contraintes de capacité, c'est-à-dire toutes les contraintes de type **(1)**. Après avoir introduit pour chaque arête $(v_i, v_j) \in E$ un multiplicateur de Lagrange λ_{ij} non négatif et relaxé les contraintes de type **(1)**, nous obtenons le problème suivant appelé aussi sous-problème de Lagrange:

$$\textbf{MIN} \quad \sum_{k=1}^{K} \sum_{\substack{(v_i,v_j) \in E \\ i<j}} (c_{ij} + \lambda_{ij}\sigma_k)x_{ij}^k \; - \sum_{\substack{(v_i,v_j) \in E \\ i<j}} \lambda_{ij}u_{ij}$$

$$\textbf{s.c.} \quad \textbf{(2), (3), (4), (5)}$$

Etant donné un vecteur λ contenant les valeurs des multiplicateurs de Lagrange, nous noterons $\mathbf{R1}(\lambda)$ le sous-problème de Lagrange obtenu en relaxant les contraintes de capacité. $L_1(\lambda)$ représentera la valeur de la solution optimale de $\mathbf{R1}(\lambda)$. Comme le deuxième terme de la fonction objectif ne dépend ni des variables x_{ij}^k ni des variables y_i^k, il ne joue aucun rôle dans la minimisation et peut donc être négligé durant cette phase. Nous pouvons également remarquer qu'aucune contrainte ne lie entre elles les variables associées à des câbles différents. Le problème $\mathbf{R1}(\lambda)$ peut donc se décomposer en K sous-problèmes indépendants. Le sous-problème associé au câble k est alors de la forme:

$$\textbf{MIN} \quad \sum_{\substack{(v_i,v_j) \in E \\ i<j}} (c_{ij} + \lambda_{ij}\sigma_k)x_{ij}^k$$

$$\textbf{s.c.} \quad \sum_{\substack{(v_i,v_j) \in E \\ j<i}} x_{ji}^k \; + \sum_{\substack{(v_i,v_j) \in E \\ i<j}} x_{ij}^k \; = 2y_i^k \quad \forall\, v_i \neq s_k,\, t_k$$

$$\sum_{\substack{(v_i,v_j) \in E \\ j<i}} x_{ji}^k \; + \sum_{\substack{(v_i,v_j) \in E \\ i<j}} x_{ij}^k \; = 1 \quad \forall\, v_i = s_k \text{ ou } t_k$$

$$x_{ij}^k = 0 \text{ ou } 1 \quad \forall\, (v_i, v_j) \in E,\; i < j$$

$$y_i^k = 0 \text{ ou } 1 \quad \forall\, v_i \in V$$

Le problème ci-dessus est un problème de plus court chemin dans le graphe initial G où le coût d'une arête (v_i, v_j) est égal à $c_{ij} + \lambda_{ij}\sigma_k$. Il peut être résolu au moyen de l'algorithme de Dijkstra. En effet nous avons $c_{ij} + \lambda_{ij}\sigma_k \geq 0$ car c_{ij} et λ_{ij} sont non négatifs et σ_k est positif. Nous pouvons alors calculer $L_1(\lambda)$ en résolvant K problèmes de plus court chemin indépendants les uns des autres.

Finalement, $Z_1 = \max_{\lambda \geq 0} L_1(\lambda)$ est une borne inférieure de notre problème pouvant être obtenue en utilisant la méthode standard des sous-gradients décrite dans [Hel74].

4.3.2 Relaxation des contraintes de chemin

La deuxième borne étudiée est basée sur la relaxation des contraintes de chemin (contraintes de type **(2)** et **(3)**). A chaque sommet v_i de V et à chaque câble k est associé le multipli-

cateur de Lagrange μ_i^k (μ_i^k est quelconque). Le sous-problème de Lagrange issu de cette relaxation est le suivant:

$$\textbf{MIN} \sum_{k=1}^{K} \sum_{\substack{(v_i,v_j)\in E \\ i<j}} (c_{ij} - \mu_i^k - \mu_j^k)x_{ij}^k + 2\sum_{k=1}^{K} \sum_{\substack{v_i \in V \\ v_i \neq s_k,\, t_k}} \mu_i^k y_i^k + \sum_{k=1}^{K} \sum_{\substack{v_i \in V \\ v_i \in \{s_k,\, t_k\}}} \mu_i^k$$

s.c. **(1), (4), (5)**

Etant donné un vecteur μ contenant les multiplicateurs de Lagrange, nous noterons $\mathbf{R2}(\mu)$ le sous-problème de Lagrange obtenu en relaxant les contraintes de chemin. La valeur de la solution optimale de $\mathbf{R2}(\mu)$ sera notée $L_2(\mu)$. Comme précédemment, le dernier terme de la fonction objectif du problème relaxé ne dépend ni des x_{ij}^k ni des y_i^k. Il sera donc négligé au cours de la phase de minimisation. Les variables y_i^k n'apparaissent pas dans les contraintes du problème relaxé, nous pouvons donc les fixer à 1 si le multiplicateur de Lagrange correspondant μ_i^k est négatif et à 0 sinon. Aucune contrainte ne lie les variables associées aux différentes arêtes de E dans le problème $\mathbf{R2}(\mu)$. Il peut donc être décomposé en $|E|$ sous-problèmes indépendants. Le sous-problème associé à l'arête (v_i, v_j) est alors de la forme:

$$\textbf{MIN} \quad \sum_{k=1}^{K} (c_{ij} - \mu_i^k - \mu_j^k)x_{ij}^k$$

$$\textbf{s.c.} \quad \sum_{k=1}^{K} \sigma_k x_{ij}^k \leq u_{ij}$$

$$x_{ij}^k = 0 \text{ ou } 1 \quad \forall k \in \{1, ..., K\}$$

Le problème ci-dessus est un problème de sac de montagne avec variables 0_1. Il s'agit d'un problème qui dans le cas général est NP_dur. Il peut cependant être facilement résolu si $\sigma_k = 1$, $\forall\, k \in \{1, ..., K\}$. En effet, soit $I^- = \{k \mid k \in \{1, ..., K\} \text{ et } c_{ij} - \mu_i^k - \mu_j^k < 0\}$. Si $|I^-| \leq u_{ij}$, alors la solution optimale consiste à poser $x_{ij}^k = 1$ si $k \in I^-$ et $x_{ij}^k = 0$ dans le cas contraire. Sinon lorsque $|I^-| > u_{ij}$, la solution optimale est obtenue en triant les éléments de I^- selon la valeur non décroissante de $c_{ij} - \mu_i^k - \mu_j^k$ et en posant $x_{ij}^k = 1$ si et seulement si k se trouve parmi les u_{ij} premiers éléments de I^-.
Pour le cas général du problème du sac de montagne avec variables 0_1, des méthodes exactes efficaces de type Branch and Bound existent. Nous avons utilisé l'algorithme proposé par Horowitz et Sahni décrit dans [Mar90] pour obtenir la solution optimale de ces sous-problèmes. Le calcul de $L_2(\mu)$ passe donc par la résolution de $|E|$ problèmes de sac de montagne indépendants les uns des autres.

Finalement, $Z_2 = \max_{\mu} L_2(\mu)$ est une borne inférieure de notre problème qui peut être obtenue en utilisant la méthode standard des sous-gradients.

4.3.3 Comparaison entre les bornes inférieures

Soit Z_{PL} la valeur de la solution optimale obtenue en résolvant le problème **PL** sans tenir compte des contraintes d'intégralité. Il existe un théorème qui dit que la borne inférieure

obtenue à l'aide des techniques de relaxation lagrangienne est toujours supérieure ou égale à Z_{PL} (voir [Geo74], [Fis81b], [Ahu93]). Nous avons donc $Z_1 \geq Z_{PL}$ et $Z_2 \geq Z_{PL}$. Il existe également une condition suffisante pour avoir l'égalité. Cette condition est appelée propriété d'intégralité. Un sous-problème de Lagrange satisfait la propriété d'intégralité si pour tout choix des coefficients de Lagrange, la valeur de sa solution optimale ne varie pas lorsque les contraintes d'intégralité sont supprimées. Nous appellerons cette propriété **(PI)**.

La valeur de la solution optimale de $\mathbf{R1}(\lambda)$ est obtenue en résolvant K problèmes de plus court chemin. La valeur de la solution optimale de tels problèmes peut être strictement supérieure à celle que l'on obtient en les résolvant sans tenir compte des contraintes d'intégralité comme le montre l'exemple ci-dessous.

Exemple:

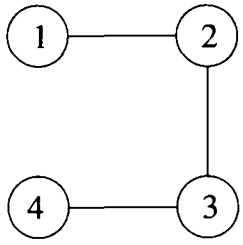


FIG. 4.3 -

Graphe G.

$$
\begin{array}{lllllll}
\textbf{MIN} & x_{12} & + & x_{23} & + & x_{34} & \\
\textbf{s.c.} & x_{12} & + & x_{23} & = & 2y_2 & \\
 & x_{23} & + & x_{34} & = & 2y_3 & \\
 & x_{12}{=}1 \text{ et } x_{34} = 1 & & & & & \\
 & x_{23},\, y_2,\, y_3 = 0 \text{ ou } 1 & & & & &
\end{array}
$$

FIG. 4.4 -

Un problème de plus court chemin dans G formulé en termes de programmation linéaire 0_1.

On considère un problème de plus court chemin dans le graphe représenté à la figure 4.3. Dans ce graphe, le coût de chaque arête est de 1. Il s'agit de déterminer le plus court chemin entre le sommet 1 et le sommet 4. La formulation de ce problème en termes de programmation linéaire 0_1 est décrite à la figure 4.4. La variable x_{ij} vaut 0 si le plus court chemin ne passe pas par l'arête (i,j) et 1 sinon, la variable y_i vaut 1 si le plus court chemin passe par le sommet i et 0 sinon. La solution optimale de ce problème vaut 3 ($x_{12} = x_{23} = x_{24} = y_2 = y_3 = 1$). En supprimant les contraintes d'intégralité, la solution optimale a une valeur de 2 ($x_{12} = x_{34} = 1, y_2 = y_3 = 0.5, x_{23} = 0$).
La valeur de la solution optimale d'un problème de plus court chemin peut donc diminuer lorsque les contraintes d'intégralité sont supprimées, il en va donc de même pour celle du problème $\mathbf{R1}(\lambda)$. Le sous-problème de Lagrange obtenu en relaxant les contraintes de capacité ne satisfait donc pas la propriété **(PI)**.

La valeur de la solution optimale de $\mathbf{R2}(\mu)$ s'obtient en résolvant $|E|$ problèmes indépendants. Ces problèmes sont des problèmes de sac de montagne à variables 0_1. Comme dans le cas des problèmes de plus court chemin, la valeur de la solution optimale de ce type de problème peut être strictement inférieure à celle obtenue en supprimant les contraintes d'intégralité. L'exemple suivant illustre un tel comportement.

Exemple:

$$\begin{array}{lrcrcl} \textbf{MIN} & -x_1 & - & x_2 & & \\ \textbf{s.c.} & 2x_1 & + & 2x_2 & \leq & 3 \\ & x_1, x_2 = 0 \text{ ou } 1 & & & & \end{array}$$

FIG. 4.5 -

Un problème de sac de montagne à variables 0_1.

La figure 4.5 représente la description d'un problème de sac de montagne à variables 0_1. La solution optimale de ce problème vaut -1 ($x_1 = 1, x_2 = 0$ ou $x_2 = 1, x_1 = 0$). En relaxant les contraintes d'intégralité, la valeur de la solution optimale est alors de -1.5. Elle peut être obtenue en posant par exemple $x_1 = 1$ et $x_2 = 0.5$ (d'autres solutions optimales existent). Le sous-problème de Lagrange $\mathbf{R2}(\mu)$ ne satisfait donc pas la propriété **(PI)**.

Lorsque la section de chaque câble vaut 1, la situation est un peu différente. Dans ce cas particulier, nous avons vu en 4.3.2 une manière de construire une solution optimale respectant les contraintes d'intégralité. Lorsque ces dernières sont supprimées, la même technique peut être utilisée. La solution optimale ainsi construite est alors identique à celle obtenue en tenant compte des contraintes d'intégralité. La propriété **(PI)** est donc satisfaite lorsque la section de chaque câble vaut 1. Nous avons alors $Z_1 \geq Z_2 = Z_{PL}$, ce qui signifie que la borne inférieure Z_1 est toujours au moins aussi bonne que la borne Z_2 lorsque $\sigma_k = 1 \ \forall k = 1, ..., K$.

4.4 Méthodes heuristiques

Deux méthodes heuristiques sont présentées dans cette section. La première proposée par Turki [Tur97] est basée sur une approche gloutonne alors que la seconde est une adaptation de la méthode tabou au PPCCC.

4.4.1 Notations

Nous noterons par P_k un chemin reliant s_k à t_k, $k = 1, ..., K$.

Une solution s du PPCCC est un ensemble de K chemins reliant les sommets s_k à t_k ($k = 1, 2, ..., K$): $s = \{P_1, ..., P_K\}$. Toutes les solutions dont il sera question à partir de maintenant satisferont les contraintes **(2)**, **(3)**, **(4)**, **(5)** du problème **PL**. Le terme "solution non admissible" désignera dès lors une solution ne satisfaisant pas les contraintes de capacité (contraintes de type **(1)** dans **PL**).

Une solution partielle est une solution (admissible ou non) composée de $J < K$ chemins $P_{k_1}, ..., P_{k_J}$ où $k_1, ..., k_J \in \{1, ..., K\}$ et $k_i \neq k_j \ \forall \ i \neq j$, $i, j = 1, ..., J$. Si s_{part} est une solution partielle composée des chemins $P_{k_1}, ..., P_{k_J}$, on définit $I(s_{part})$ de la manière suivante: $I(s_{part}) = \{k_1, ..., k_J\}$. $I(s_{part})$ est donc l'ensemble des indices des câbles dont les chemins sont tracés dans la solution partielle s_{part}. La capacité résiduelle d'une arête (v_i, v_j) associée à la solution partielle s_{part} sera notée $u_{ij}(s_{part})$. Elle est définie de la

manière suivante:

$$u_{ij}(s_{part}) = u_{ij} - \sum_{\substack{r \in I(s_{part}) \\ (v_i, v_j) \in P_r}} \sigma_r$$

Soit s_{part} une solution partielle, et soit (s_k, t_k) une paire de sommets de C non reliés par un chemin dans s_{part} ($k \notin I(s_{aprt})$). On définit le graphe $G(s_{part}, (s_k, t_k))$ comme étant le graphe obtenu en enlevant du graphe initial G toutes les arêtes dont la capacité résiduelle (par rapport à s_{part}) n'est plus suffisante pour accueillir le câble reliant s_k à t_k. De façon plus précise, une arête (v_i, v_j) de G est enlevée si: $\sigma_k > u_{ij}(s_{part})$.

Comme nous allons travailler avec des solutions admissibles et des solutions non admissibles, deux fonctions objectifs vont être utilisées. La première $F_1(s)$ est simplement la longueur totale des chemins parcourus par les câbles dans la solution s. La deuxième, $F_2(s)$ est définie de la manière suivante: $F_2(s) = F_1(s) + \alpha \triangle(s)$ où $\triangle(s)$ est la somme des dépassements de capacité engendrés par la solution s, et α est un coefficient de pénalité. Si s est une solution admissible alors $\triangle(s)$ est nul et $F_1(s) = F_2(s)$. Nous noterons par F_1^* et F_2^* les meilleures valeurs rencontrées de F_1 et F_2 respectivement.

Les notations que nous venons d'introduire sont illustrées dans l'exemple qui suit. La figure 4.6 représente le graphe G avec les deux couples de sommets (s_1, t_1) et (s_2, t_2) qui doivent être reliés par deux câbles. La section des câbles vaut 1 de même que la capacité des arêtes. Le coût de chaque arête est indiqué sur le dessin. La figure 4.7 représente une solution admissible s, du PPCCC dans G. La valeur de s est 6. La solution s est composée de deux chemins: $P_1 = (s_1, a, t_1)$ et $P_2 = (s_2, t_1, t_2)$.
La figure 4.8 correspond à une solution non admissible $\tilde{s}$, dans G. Les câbles 1 et 2 traversent l'arête (t_1, a) provoquant un dépassement de sa capacité. La valeur de $\tilde{s}$ est égale à $F_2(\tilde{s}) = F_1(\tilde{s}) + \alpha\Delta(\tilde{s}) = 5 + \alpha$. La dernière figure (figure 4.9) représente le graphe $G(s_{part}, (s_2, t_2))$ où s_{part} est la solution partielle composée uniquement du chemin $P_1 = (s_1, a, t_1)$ pris par le câble 1. Par rapport au graphe initial G, les arêtes (s_1, a) et (a, t_1) ont été supprimées car le câble 2 ne peut pas traverser ces arêtes sans entraîner un dépassement de leur capacité.

Exemple:

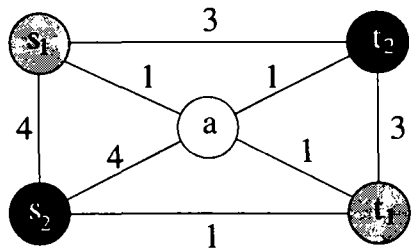


FIG. 4.6 -

Réseau G.

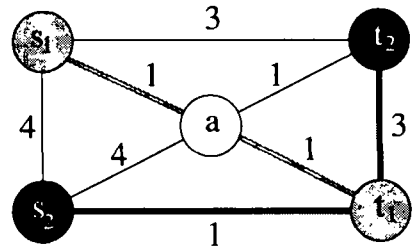


FIG. 4.7 -

Une solution du PPCCC dans G.

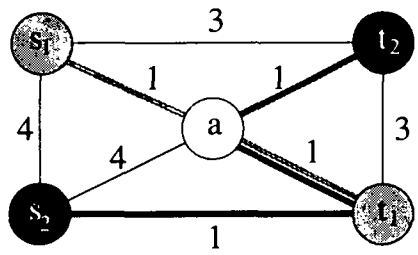


FIG. 4.8 –

Une solution $\tilde{s}$ non admissible dans G.

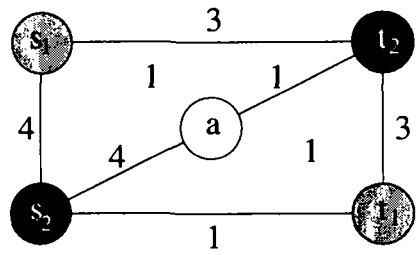


FIG. 4.9 –

Le graphe $G(s_{part}, (s_2, t_2))$.

4.4.2 Approche gloutonne

Etant donné un ordre des câbles, l'approche gloutonne que nous présentons consiste simplement à faire passer successivement chaque câble par le plus court chemin possible qui respecte la capacité des arêtes. Si tous les K câbles ont pu être tracés, la solution que l'on obtient est forcément admissible. Ce procédé est répété un nombre i_{max} de fois, en partant d'un ordre des câbles généré aléatoirement. La meilleure solution admissible rencontrée est mémorisée. L'algorithme échoue si aucune des i_{max} tentatives n'a permis d'obtenir une solution admissible. Cette heuristique a été proposée par Turki [Tur97]. Nous l'avons appelée PPCCC_GLOUTON. Sa description détaillée figure ci-dessous.

Algorithme: PPCCC_GLOUTON

ENTREE: Un graphe $G = (V, E)$ et les données concernant les K câbles à tracer dans G.

SORTIE: Une solution admissible s^* du PPCCC dans G ou une indication signifiant que l'algorithme a échoué.

Etape 0. *(Initialisation)*
Poser $i := 0$, $s := \emptyset$, $F_1^* := \infty$, $s^* = \emptyset$.

Etape 1. Si $i = i_{max}$ aller à l'étape 4.
Sinon générer un ordre aléatoire des câbles: $O_1, ..., O_K$. Poser k:=1.

Etape 2. Déterminer le plus court chemin P_{O_k} reliant s_{O_k} à t_{O_k} dans le graphe $G(s, (s_{O_k}, t_{O_k}))$.
Si un tel chemin existe:

Poser $s := s \cup \{P_{O_k}\}$.
Si k=K et $F_1(s) < F_1^*$ alors poser $F_1^* := F_1(s)$ et $s^* := s$.

Sinon poser $i := i + 1$, $s := \emptyset$ et retourner à l'étape 1.

Etape 3. Si $k = K$ alors poser $i := i + 1$, $s := \emptyset$ et retourner à l'étape 1.
Sinon poser $k := k + 1$ et aller à l'étape 2.

Etape 4. STOP. Si $s^* = \emptyset$, l'algorithme a échoué, sinon s^* est la solution retournée par l'algorithme.

Le seul paramètre de PPCCC_GLOUTON est le nombre d'ordres aléatoires i_{max} générés. L'influence de ce paramètre sera discutée plus loin. L'exemple ci-dessous illustre le fonctionnement de l'algorithme que nous venons de décrire.

Exemple:

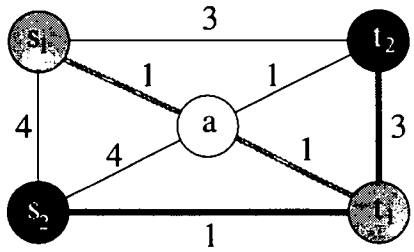


FIG. 4.10 -

Solution de PPCCC_GLOUTON en traitant le câble 1 puis le câble 2.

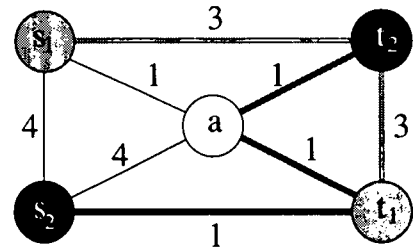


FIG. 4.11 -

Solution de PPCCC_GLOUTON en traitant le câble 2 puis le câble 1.

Dans l'exemple ci-dessus, nous avons appliqué PPCCC_GLOUTON au problème associé au réseau de la figure 4.6. La figure 4.10 représente la solution admissible obtenue en considérant le câble 1 avant le câble 2. Le chemin du câble 1 est identique au plus court chemin dans G reliant s_1 à t_1: $P_1 = (s_1, a, t_1)$. Soit s_{part} la solution composée uniquement de P_1. Le câble 2 passe par le chemin $P_2 = (s_2, t_1, t_2)$ qui est le plus court chemin reliant s_2 à t_2 dans le graphe $G(s_{part}, (s_2, t_2))$ (voir figure 4.9). La valeur de cette solution est 6. La figure 4.11 correspond à la solution obtenue en traitant le câble 2 avant le câble 1. Le coût de cette solution est alors de 9.

Il existe des exemples où il est impossible de trouver, avec PPCCC_GLOUTON, une solution admissible quand bien même une telle solution existe. Cela arrive lorsque les chemins des premiers câbles traités, quel que soit l'ordre des câbles utilisés, rendent impossible le tracé d'au moins un des câbles restant.

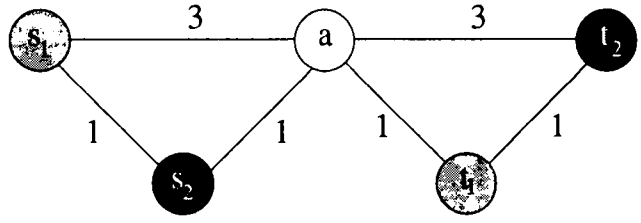


FIG. 4.12 -

Exemple où PPCCC_GLOUTON échoue systématiquement.

La figure 4.12 représente un exemple où PPCCC_GLOUTON ne peut jamais trouver de solution admissible alors qu'il en existe une. Comme dans les exemples précédents, la section des câbles ainsi que la capacité des arêtes valent 1. Deux câbles doivent être tracés.

Si le câble 1 est traité en premier, il utilise le plus court chemin possible c'est-à-dire le chemin $P_1 = (s_1, s_2, a, t_1)$. Le câble 2 ne peut alors plus relier s_2 à t_2 sans violer des contraintes de capacité. En effet, toutes les arêtes incidentes à s_2 ont une capacité résiduelle nulle. De même lorsque le câble 2 est tracé avant le câble 1, il emprunte le chemin $P_2 = (s_2, a, t_1, t_2)$. Toutes les arêtes incidentes au sommet t_1 sont saturées et ne peuvent plus accueillir le câble 1. Il est donc impossible de relier s_1 à t_1 sans violation des contraintes de capacité. Ainsi quel que soit l'ordre des câbles il est impossible de trouver une solution admissible pour ce problème en utilisant PPCCC_GLOUTON. Une solution admissible existe cependant qui consiste à faire passer le câble 1 par le chemin (s_1, a, t_1) et le câble 2 par le chemin (s_2, a, t_2).

Nous avons décrit une approche gloutonne basée sur la recherche de plus courts chemins admissibles (i.e. chemins ne violant pas les contraintes de capacité) dans des graphes auxiliaires. D'autres approches sont possibles, utilisant d'autres techniques pour construire les chemins empruntés par les câbles. Cela nous amène à définir la notion d'approche gloutonne pour le PPCCC.

Etant donné un ordre des câbles: $O_1, O_2, ..., O_K$, nous appellerons approche gloutonne pour le PPCCC toute méthode qui consiste à faire passer successivement, et selon l'ordre que nous venons de définir, les câbles par un chemin admissible. Les techniques utilisées pour construire les chemins admissibles traversés par les câbles importent peu. Nous supposerons toutefois qu'elles sont suffisamment élaborées pour trouver un chemin admissible reliant s_{O_k} à t_{O_k} $(1 \leq k \leq K)$ lorsqu'il en existe un. Cela signifie qu'une approche gloutonne pour le PPCCC échoue lorsqu'il n'existe pas, pour un câble O_k et compte tenu des câbles $O_1, ..., O_{k-1}$ déjà tracés, de chemin admissible entre s_{O_k} et t_{O_k}.

Soit $X \subset V$ et soit $\overline{X} = V \setminus X$ son complémentaire. Nous noterons par $(X, \overline{X})$ l'ensemble des arêtes de E ayant une extrémité dans X et l'autre dans $\overline{X}$. Par abus de langage, nous noterons $(s_k, t_k) \in (X, \overline{X})$ pour indiquer qu'un des deux sommets, s_k ou t_k appartient à X et l'autre à $\overline{X}$.
Etant donné un ensemble $X \subset V$ $(X \neq \emptyset)$ et une solution partielle $s_{part} = \{P_{O_1}, ..., P_{O_{k-1}}\}$ $(0 \leq k \leq K)$ nous allons également définir le terme suivant:

$$U_{max}(X) = \max_{(v_i, v_j) \in (X, \overline{X})} u_{ij}(s_{part})$$

$U_{max}(X)$ est la capacité résiduelle maximum d'une arête appartenant à la coupe $(X, \overline{X})$.

Théorème 1 *Une approche gloutonne pour le PPCCC échoue systématiquement si et seulement si, pour tout ordre des câbles: $O_1, ..., O_K$, il existe un ensemble $X \subset V$ et une solution partielle $s_{part} = \{P_{O_1}, ..., P_{O_{k-1}}\}$ $(0 \leq k \leq K)$ composée des $k-1$ premiers chemins admissibles obtenus à l'aide de l'approche gloutonne, tels que:*

$$(s_{O_k}, t_{O_k}) \in (X, \overline{X}) \text{ et } U_{max}(X) < \sigma_k$$

preuve:

($\Rightarrow$)

Soit $O_1, ..., O_K$ un ordre des câbles. Par hypothèse, l'approche gloutonne échoue systématiquement. Cela signifie qu'il existe un câble O_k ($1 \leq k \leq K$) pour lequel il n'existe pas de chemin admissible reliant s_{O_k} à t_{O_k}, lorsque les câbles $O_1, ..., O_{k-1}$ sont déjà tracés. Nous allons montrer que dans un tel cas, il existe une coupe $(X, \overline{X})$ telle que $(s_k, t_k) \in (X, \overline{X})$ et telle que $U_{max}(X) < \sigma_k$.

Supposons par l'absurde que nous ayons $U_{max}(X) \geq \sigma_k$ pour toute coupe $(X, \overline{X})$ telle que $(s_k, t_k) \in (X, \overline{X})$, et introduisons la notation suivante:

$$A(X) = \{v_j \in \overline{X} \mid \exists\ v_i \in X \text{ tel que } (v_i, v_j) \in E \text{ et } u_{ij}(s_{part}) \geq \sigma_k\}$$

$A(X)$ est l'ensemble des sommets de $\overline{X}$ connectés à un sommet de X par une arête dont la capacité résiduelle est supérieure à σ_k.
Posons $Y := s_k$. Posons ensuite $Y := Y \cup A(Y)$ et répétons cette dernière opération jusqu'à ce que $t_k \in Y$. Par construction et par définition de l'ensemble $A(Y)$, tous les sommets de Y sont connectés à s_k par un chemin dont les arêtes ont une capacité résiduelle supérieure ou égale à σ_k. Il existe donc un chemin admissible reliant s_k et t_k. Cela contredit le fait que l'approche gloutonne ait échoué. Il est donc faux de supposer que $U_{max}(X) \geq \sigma_k$ pour toute coupe $(X, \overline{X})$ telle que $(s_k, t_k) \in (X, \overline{X})$.

Ainsi il existe une coupe $(X, \overline{X})$ telle que $(s_k, t_k) \in (X, \overline{X})$ et pour laquelle nous avons $U_{max}(X) < \sigma_k$.

($\Leftarrow$)

Par hypothèse nous avons que pour tout ordre $O_1, ..., O_K$, il existe un ensemble $X \subset V$ et une solution partielle $s_{part} = \{P_{O_1}, ..., P_{O_{k-1}}\}$ tels que $(s_{O_k}, t_{O_k}) \in (X, \overline{X})$ et tels que $U_{max}(X) < \sigma_k$.

Une des extrémités du câble O_k se trouve dans X et l'autre dans $\overline{X}$. Ainsi tout chemin reliant s_{O_k} à t_{O_k} qui sera construit au cours de l'approche gloutonne passera par une arête appartenant à la coupe $(X, \overline{X})$. La capacité résiduelle de toutes les arêtes reliant X à $\overline{X}$ est strictement inférieure à la section du câble O_k. Il n'est donc pas possible de construire un chemin admissible reliant s_{O_k} à t_{O_k}. L'approche gloutonne va ainsi échouer. Ce raisonnement est valable quel que soit l'ordre des câbles considéré.

♯

Nous pouvons reprendre l'exemple de la figure 4.12 pour illustrer le théorème 1. L'approche gloutonne que nous utilisons est l'algorithme PPCCC_GLOUTON.
Considérons tout d'abord l'ordre des câbles où $O_1 = 1$ et $O_2 = 2$. Lorsque le premier câble est tracé, nous obtenons une solution partielle composée du chemin $P_1 = (s_1, s_2, a, t_1)$. En posant $X = \{s_2\}$, nous avons alors $U_{max}(X) = 0$ puisque toutes les arêtes incidentes à s_2 ont une capacité résiduelle nulle. Le câble 2 n'est pas encore tracé, $\sigma_2 = 1$ et nous avons

$(s_2, t_2) \in (X, \overline{X})$.
Lorsque l'ordre des câbles est $O_1 = 2$ et $O_2 = 1$, la solution partielle obtenue après que le premier câble soit traité est composée du chemin $P_2 = (s_2, a, t_1, t_2)$. En posant $X = \{t_1\}$ nous avons alors $U_{max}(X) = 0$ et $(s_1, t_1) \in (X, \overline{X})$. Le câble 1 n'est pas tracé et $\sigma_1 = 1$. Ainsi quel que soit l'ordre des câbles considéré par l'algorithme PPCCC_GLOUTON, il existe toujours un ensemble X et une solution partielle $s_{part} = \{P_{O_1}\}$ satisfaisant la condition du théorème 1. Nous savons donc que PPCCC_GLOUTON ne trouvera jamais de solution admissible pour l'exemple de la figure 4.12.

4.4.3 Méthode tabou appliquée au PPCCC

Nous allons reprendre les divers ingrédients intervenant dans une méthode tabou que nous avons déjà décrits à la section 2.2.2 du chapitre 2. Nous décrirons la manière dont nous les avons adaptés dans le cas du PPCCC.

4.4.3.1 Espace des solutions

Une solution est un ensemble de chemins $P_1, ..., P_K$, chaque chemin $P_k (1 \leq k \leq K)$ reliant le sommet s_k au sommet t_k. La visite de solutions ne respectant pas les contraintes de capacité est autorisée. L'espace des solutions S dans lequel nous allons travailler est donc formé de tous les ensembles de K chemins reliant s_k à t_k $(1 \leq k \leq K)$.

4.4.3.2 Fonction objectif

Même si le véritable objectif de notre problème est de minimiser la fonction F_1, le fait d'autoriser la visite de solutions pouvant violer les contraintes de capacité nous oblige à prendre en compte ces violations. La fonction que nous allons chercher à minimiser à l'aide de la méthode tabou est donc la fonction:

$$F_2(s) = F_1(s) + \alpha\Delta(s)$$

La variable $\alpha \in \mathbb{R}_+^*$ est un coefficient de pénalité dont la valeur sera discutée plus loin.

4.4.3.3 Solution initiale

La solution initiale que nous allons utiliser consiste simplement à faire passer chaque câble k par le plus court chemin dans G entre s_k et t_k $(1 \leq k \leq K)$. Notre solution initiale, s_0 n'est généralement pas admissible. En effet, la capacité des arêtes n'est pas prise en compte lors de sa construction. Comme elle est composée de plus courts chemins dans G, s_0 est optimale lorsqu'elle est admissible.

4.4.3.4 Le voisinage

Soient $s \in S$ une solution du PPCCC, P_k le chemin pris par le câble k dans s et (v_i, v_j) une arête appartenant à P_k. Soient également s' la solution partielle obtenue en supprimant P_k de s, et α le coefficient pénalisant les violations de capacité. Le graphe $H(s',(s_k, t_k),(v_i, v_j))$ est obtenu à partir du graphe initial G de la manière suivante:

- L'arête (v_i, v_j) est enlevée de G.

- On définit un nouveau coût, c'_{ab}, sur toutes les autres arêtes (v_a, v_b) de G de telle sorte que:

$$c'_{ab} = \begin{cases} c_{ab} + \alpha & \text{si } \sigma_k > u_{ab}(s') \\ c_{ab} & \text{sinon} \end{cases}$$

Le graphe $H(s',(s_k,t_k),(v_i,v_j))$ est donc le graphe obtenu en supprimant l'arête (v_i,v_j) de G et en pénalisant de α le coût des arêtes dont la capacité résiduelle, étant donné s', n'est pas suffisante pour pouvoir accueillir le câble k. La solution voisine de s associée au câble k et à l'arête (v_i,v_j) est obtenue en remplaçant le chemin P_k pris par le câble k dans s par un plus court chemin P reliant s_k à t_k dans $H(s',(s_k,t_k),(v_i,v_j))$. S'il n'existe pas de chemin reliant s_k à t_k dans $H(s',(s_k,t_k),(v_i,v_j))$, la solution voisine de s associée au câble k et à l'arête (v_i,v_j) est identique à s. Le chemin P, lorsqu'il existe, est différent de P_k car l'arête (v_i,v_j) n'existe pas dans $H(s',(s_k,t_k),(v_i,v_j))$. Il a tendance à éviter les arêtes ayant une capacité résiduelle inférieure à la section du câble k puisque le coût de ces arêtes est pénalisé dans $H(s',(s_k,t_k),(v_i,v_j))$.

Nous noterons par $N(s)$ l'ensemble de toutes les solutions voisines de s. La taille de $N(s)$ dépend du nombre de câbles K à traiter et du nombre d'arêtes appartenant à chacun des chemins formant s. Nous allons examiner un sous-ensemble N^* de $N(s)$ à chaque itération. N^* sera composé d'un certain nombre nb_{vois} de solutions voisines de s choisies de façon aléatoire. Le choix de ce paramètre sera discuté dans la partie consacrée aux résultats numériques.

Exemple:

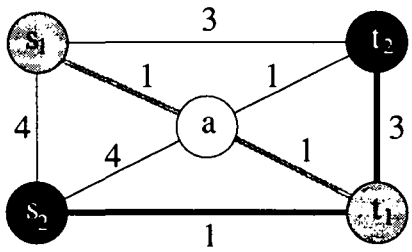


FIG. 4.13 -

Solution courante s.

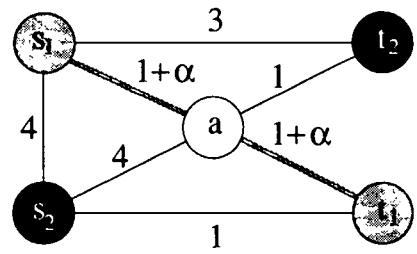


FIG. 4.14 -

Graphe $H(s',(s_2,t_2),(t_1,t_2))$.

Nous traitons à nouveau le problème associé au réseau de la figure 4.6. Rappelons que la section des câbles ainsi que la capacité des arêtes valent 1. La figure 4.13 représente une solution admissible s du PPCCC. En supprimant le chemin $P_2 = (s_2,t_1,t_2)$ du câble 2, nous obtenons une solution partielle s' composée uniquement du chemin $P_1 = (s_1,a,t_1)$. Considérons l'arête (t_1,t_2) de P_2. Le graphe $H(s',(s_2,t_2),(t_1,t_2))$ est représenté dans la figure 4.14.

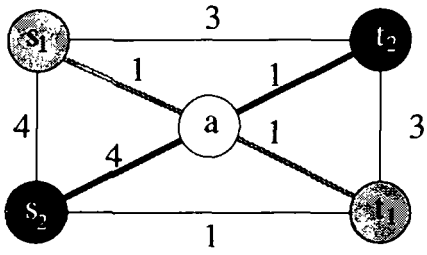


FIG. 4.15 –

Solution voisine de s lorsque $\alpha > 2$.

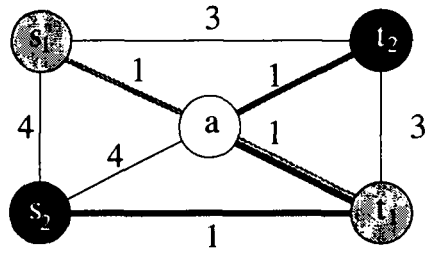


FIG. 4.16 –

Solution voisine de s lorsque $\alpha < 2$.

Le nouveau chemin emprunté par le câble 2 est le plus court chemin reliant s_2 à t_2 dans $H(s',(s_2,t_2),(t_1,t_2))$. Suivant la valeur de α, deux solutions différentes peuvent être obtenues. Si α est strictement supérieur à 2, la solution voisine de s est admissible et le chemin du câble 2 est le suivant: (s_2,a,t_2) (voir figure 4.15). Lorsque α est strictement inférieur à 2, le plus court chemin dans $H(s',(s_2,t_2),(t_1,t_2))$ est (s_2,t_1,a,t_2). La solution voisine de s n'est alors plus admissible (voir figure 4.16). Finalement si α vaut 2, la solution voisine de s pourra être soit la solution de la figure 4.15, soit celle représentée en 4.16, cela dépend de l'algorithme utilisé pour calculer les plus courts chemins. Aucune préférence particulière n'est accordée à une solution admissible par rapport à une solution qui ne l'est pas, si les deux solutions ont la même valeur.

4.4.3.5 La liste tabou

La liste tabou $\mathcal{T}$ que nous allons utiliser consiste à interdire tout changement de l'itinéraire du câble venant d'être modifié pendant un nombre θ d'itérations. Ce nombre est choisi aléatoirement à chaque itération dans l'intervalle $[\theta_{min},\theta_{max}]$. Nous avons fixé $\theta_{min} = \frac{K}{6}$ et $\theta_{max} = \frac{K}{3}$ où K est le nombre de câbles à tracer. D'autres valeurs ont été testées. Les meilleurs résultats ont été obtenus avec les valeurs que nous venons de mentionner.

4.4.3.6 Le coefficient de pénalité

Le coefficient de pénalité α est ajusté automatiquement lors du déroulement de la méthode tabou toutes les β itérations (voir [Gen94]). Si au cours des β dernières itérations seules des solutions admissibles ont été visitées, la valeur de α est réduite de moitié. Elle est doublée si les trois conditions suivantes sont satisfaites:

1) les β dernières solutions visitées étaient toutes non admissibles

2) le dépassement total de capacité n'a pas diminué à l'itération précédente

3) $\alpha \leq \frac{\alpha_{lim}}{2}$

La deuxième condition a pour but d'éviter de doubler α alors que sa valeur est encore assez grande pour provoquer une réduction des dépassements de capacité. Dans la troisième condition, la valeur limite α_{lim}, sert uniquement à éviter des problèmes de mémoire dus à une trop grande valeur de α.
Lorsque que la valeur de α n'est ni doublée, ni divisée par deux, elle reste inchangée. La valeur initiale de α a été fixée égale à la longueur moyenne des arêtes du graphe. Cette valeur est assez importante, elle a pour but de limiter le nombre de solutions non admissibles visitées en début d'exécution. On espère ainsi rapidement visiter une solution admissible.

La valeur de α_{lim} est égale à la valeur initiale de α multipliée par 128. Cela signifie que 7 doublements successifs de α sont autorisés au maximum. Au delà de cette valeur, soit le paramètre α est réduit de moitié, soit il reste inchangé. La fréquence d'ajustement de α, β, a été choisie égale à 20. D'autres valeurs de β ont été testées, elles n'ont pas permis d'obtenir de meilleurs résultats. Le coefficient α est donc ajusté toutes les 20 itérations.

4.4.3.7 Le critère d'aspiration

Le statut tabou d'un câble k est provisoirement levé (pour une itération) si une des deux conditions suivantes est remplie:

- La solution s' voisine de s obtenue en modifiant le câble k est admissible et $F_1(s') < F_1^*$.
- La solution s' voisine de s obtenue en modifiant le câble k est telle que $F_2(s') < F_2^*$.

4.4.3.8 Critères d'arrêt

L'exécution de la méthode tabou est interrompue si la solution initiale s_0 est admissible. En effet, si s_0 est admissible il s'agit alors d'une solution optimale. Un deuxième critère d'arrêt stoppe l'exécution de la méthode tabou dès qu'une solution admissible de coût égal à la valeur d'une des bornes inférieures Z_1 ou Z_2 est obtenue. L'exécution de la méthode tabou est également stoppée si au cours des ρ dernières itérations aucune des valeurs F_2^* et F_1^* n'a été modifiée. Le nombre d'itérations sans modification de F_1^* ou de F_2^* sera noté μ. La valeur de ρ est fixée à 500.

4.4.3.9 Procédures insérées dans la méthode tabou

Plusieurs procédures particulières sont utilisées dans la méthode tabou que nous avons développée. Deux d'entre elles sont des procédures de post-optimisation.

Procédure RETRACER_CABLES

Notre solution initiale est généralement non admissible, c'est la méthode tabou qui, en minimisant la fonction F_2, va tenter de trouver une solution admissible. Cela ne se fait pas toujours sans difficulté. Lorsque les violations des contraintes de capacité sont nombreuses et si le problème est de taille importante, un nombre élevé d'itérations peut s'écouler avant que la méthode tabou ne parvienne à trouver une solution admissible. Il peut également arriver qu'elle ne trouve pas de solution admissible alors qu'il en existe une. Pour essayer d'éviter un tel comportement, nous avons développé une procédure qui a pour but de réduire le dépassement total de capacité d'une solution non admissible. L'idée sur laquelle se base cette procédure consiste à repérer un groupe de câbles en rapport avec les contraintes de capacité violées et de retracer ces câbles. Ce processus est répété jusqu'à ce qu'une solution admissible soit obtenue ou qu'un nombre limite de répétitions soit effectué sans succès.

Cette procédure sera appelée RETRACER_CABLES, elle est décrite à la page suivante.

Algorithme: RETRACER_CABLES

ENTREE: Une solution s non admissible du PPCCC dans le graphe $G = (V, E)$.

SORTIE: Une solution $\bar{s}$ (pas forcément admissible) obtenue en retraçant certains câbles de s.

Etape 0. *(Initialisation)*
Poser $i := 0$.

Etape 1. Déterminer l'ensemble des câbles traversant dans s une arête dont la contrainte de capacité est violée. Soit $P_{k_1}, P_{k_2}, ..., P_{k_J}$ l'ensemble des chemins empruntés par ces câbles.

Etape 2. Déterminer l'ensemble des câbles ne violant aucune contrainte de capacité dans s mais dont le chemin a au moins un sommet commun avec un des chemins $P_{k_1}, P_{k_2}, ..., P_{k_J}$. Soit $P_{k_{J+1}}, ..., P_{k_{J+Q}}$ les chemins traversés par ces câbles.

Etape 3. Poser $\bar{s} := s \setminus \{P_{k_1}, ..., P_{k_{J+Q}}\}$ et j:=1.
Générer aléatoirement un ordre $O_1, ..., O_{J+Q}$ des câbles trouvés aux étapes 1 et 2.

Etape 4. Déterminer le plus court chemin P entre s_{O_j} et t_{O_j} dans le graphe $G(\bar{s}, (s_{O_j}, t_{O_j}))$.
Si $P = \emptyset$ alors poser $P := P_{k_{O_j}}$.
Poser $\bar{s} := \bar{s} \cup \{P\}$ et $j := j + 1$.
Si $j \leq J + Q$ recommencer l'étape 4, sinon poser $i := i + 1$.

Etape 5. Si $\bar{s}$ est admissible ou si $i = 1000$ STOP.
Sinon poser $s := \bar{s}$ et aller à l'étape 1.

La procédure RETRACER_CABLES est utilisée en dernier recours dans la méthode tabou lorsque celle-ci peine à trouver une solution admissible. Elle est appliquée lorsque le coefficient de pénalité α ne peut plus être doublé sans dépasser sa valeur limite α_{lim} et qu'aucune solution admissible n'a encore été visitée.

Procédure POST-OPT1

Cette première procédure de post-optimisation essaye d'améliorer une solution admissible s en tentant de remplacer successivement chaque chemin de s par un chemin plus court sans que cela ne provoque de violation des contraintes de capacité.

Algorithme: POST-OPT1

ENTREE: Une solution s admissible du PPCCC dans le graphe $G = (V, E)$.

SORTIE: Une solution admissible $\bar{s}$ de coût au pire égal à celui de s.

Etape 1. Poser $k := 1$, $\bar{s} := s$ et $F_{best} := F_1(s)$.

Etape 2. Soit P_k le chemin reliant s_k à t_k dans s. Poser $s' := s \setminus P_k$.
Déterminer le plus court chemin P reliant s_k à t_k dans le graphe $G(s', (s_k, t_k))$.
Si P est plus court que P_k alors poser $s' := s' \cup \{P\}$ puis $s := s'$.

Etape 3. Si $k = K$ alors

Si $F_1(s) < F_{best}$ retourner à l'étape 1.
Sinon STOP, $\bar{s}$ est la solution fournie par l'algorithme.

Sinon poser $k := k + 1$ et aller à l'étape 2.

Un procédé semblable a été proposé par Minoux [Min75] pour la recherche d'une solution admissible du PPCCC. L'objectif poursuivi par Minoux est de trouver une solution $s \in S$ qui minimise $\Delta(s)$. Pour cela, les chemins des câbles de s sont tour à tour remplacés par un plus court chemin calculé dans un graphe G''. Ce graphe est identique au graphe initial G si ce n'est que le coût des arêtes est différent. Soit s_{part} la solution partielle obtenue à partir de s en supprimant le chemin emprunté par le câble k. Le coût c''_{ij} d'une arête (v_i, v_j) de G'' lorsque le chemin du câble k est retracé est le suivant:

$$c''_{ij} = \begin{cases} 0 & \text{si} \quad u_{ij}(s_{part}) \geq \sigma_k \\ \sigma_k - u_{ij}(s_{part}) & \text{si} \quad 0 \leq u_{ij}(s_{part}) \leq \sigma_k \\ \sigma_k & \text{si} \quad u_{ij}(s_{part}) < 0 \end{cases}$$

Le coût c''_{ij} correspond à l'augmentation de $\Delta(s_{part})$ provoquée par le passage du câble k sur l'arête (v_i, v_j) de G''.

Procédure POST-OPT2

Cette procédure supprime deux chemins d'une solution admissible s et tente de les remplacer par deux autres chemins de longueur totale plus petite. Comme dans POST-OPT1, on fait attention à ne pas engendrer de violation des contraintes de capacité. Lorsque les chemins P_r et P_q sont ôtés de s, deux manières sont envisagées pour construire les nouveaux chemins. La première consiste à tracer le chemin reliant s_r à t_r avant celui reliant s_q à t_q. La deuxième revient à relier d'abord s_q à t_q puis s_r à t_r.

La procédure POST-OPT2 est décrite ci-dessous. Sa complexité est de l'ordre de $O(K^2|V|^2)$. En effet, pour chaque couple de câbles (r, q), quatre problèmes de plus court chemin sont résolus dans des graphes auxiliaires comportant $|V|$ sommets. L'algorithme utilisé pour la recherche des plus courts chemins est celui de Dijkstra dont la complexité est de l'ordre de $O(|V|^2)$.

Algorithme: POST-OPT2

ENTREE: Une solution s admissible du PPCCC dans le graphe $G = (V, E)$.

SORTIE: Une solution admissible $\bar{s}$ de coût au pire égal à celui de s.

Etape 0. *(Initialisation)*
Poser $r := 1$, $q := 2$, $F_{best} := F_1(s)$ et $\bar{s} := s$.

Etape 1. *(Le câble r est tracé avant le câble q.)*
Soient P_r et P_q les chemins reliant s_k à t_k et s_q à t_q dans $\bar{s}$.
Construire la solution partielle s' en supprimant P_r et P_q de $\bar{s}$.
Déterminer le plus court chemin P reliant s_r à t_r dans $G(s', (s_r, t_r))$.
Poser $s' := s' \cup \{P\}$.
Déterminer le plus court chemin P' reliant s_q à t_q dans $G(s', (s_q, t_q))$.
Si $P' = \emptyset$ (i.e. il n'existe pas de chemin reliant s_q à t_q dans $G(s', (s_q, t_q))$ aller à l'étape 2, sinon poser $s' := s' \cup \{P'\}$.
Si $F_1(s') < F_{best}$ poser $\bar{s} := s'$ et $F_{best} := F_1(s')$.

Etape 2. *(Le câble q est tracé avant le câble r.)*
Supprimer P et P' de s'.
Déterminer le plus court chemin P' reliant s_q à t_q dans $G(s', (s_q, t_q))$.
Poser $s' := s' \cup \{P'\}$.
Déterminer le plus court chemin P reliant s_r à t_r dans $G(s', (s_r, t_r))$.
Si $P = \emptyset$ (i.e. il n'existe pas de chemin reliant s_r à t_r dans $G(s', (s_r, t_r))$ aller à l'étape 3, sinon poser $s' := s' \cup \{P\}$.
Si $F_1(s') < F_{best}$ poser $\bar{s} := s'$ et $F_{best} := F_1(s')$.

Etape 3. Si $q = K$ alors

Si $r < K - 1$ poser $r := r + 1$, $q := r + 1$ et aller à l'étape 1.
Sinon STOP, $\bar{s}$ est la solution fournie par l'algorithme.

Sinon poser $q := q + 1$ et aller à l'étape 1.

4.4.3.10 Algorithme PPCCC_TABOU

Nous appellerons PPCCC_TABOU notre adaptation de la méthode tabou au PPCCC. La description de cet algorithme est présentée à la page suivante. Avant cela il nous reste à introduire quelques notations. Nous noterons $\underline{F}$ la meilleure des deux bornes Z_1 et Z_2 obtenues par relaxation lagrangienne. Les solutions s_{best} et s^*_{adm} sont respectivement la meilleure solution (admissible ou non) rencontrée au cours de l'exécution de la méthode tabou et la meilleure solution admissible visitée lors de cette exécution. La solution s^*_{adm} est la solution fournie par PPCCC_TABOU. Rappelons pour terminer que $F_1^* = F_1(s^*_{adm})$ et que $F_2^* = F_2(s_{best})$.

Algorithme: PPCCC_TABOU

ENTREE: Un graphe $G = (V, E)$ et les données concernant les K câbles à tracer dans G.

SORTIE: Une solution admissible s^*_{adm} ou une indication que l'algorithme a échoué.

Etape 0. *(Initialisation)*
Construire la solution initiale s_0 selon la technique décrite en 4.4.3.3.
Si $\Delta(s_0) = 0$ poser $s^*_{adm} := s_0$ et $F_1^* := F_1(s_0)$ puis STOP: s^*_{adm} est une solution optimale du PPCCC dans G.
Sinon poser $i := 0$, $s := s_0$, $F_1^* := \infty$, $s^*_{adm} := \emptyset$, $F_2^* := F_2(s)$, $s_{best} := s$, $\mathcal{T} := \emptyset$ $\rho := 500$ et $\mu := 0$.

Etape 1. *(Critères d'arrêt)*
Si $\mu = \rho$ aller à l'étape 5.
Si $F_1^* = \underline{F}$, STOP, s^*_{adm} est une solution optimale du PPCCC dans G.

Etape 2. *(Itération courante)*
Poser $i := i + 1$. Générer l'ensemble $N^* \subset N(s)$ composé de nb_{vois} solutions voisines de s. Ces solutions ne sont pas tabou ou alors sont tabou et satisfont le critère d'aspiration. Déterminer la meilleure solution s_{vois} de N^*. Soit k le câble dont le tracé a été modifié pour passer de s à s_{vois}. Mettre k dans $\mathcal{T}$ pour les θ prochaines itérations.
Si s_{vois} est admissible et s ne l'est pas ou si $F_1(s_{vois}) < F_1^*$ aller à l'étape 3.
Aller à l'étape 4.

Etape 3. *(Post-optimisation)*
Appliquer à s_{vois} la procédure POST-OPT1. Soit $\bar{s}$ la solution obtenue, poser $s_{vois} := \bar{s}$.
Appliquer à s_{vois} la procédure POST-OPT2. Soit $\bar{s}$ la solution obtenue, poser $s_{vois} := \bar{s}$.

Etape 4. *(Mises à jour)*
Poser $s := s_{vois}$ et $\mu := \mu + 1$.
Si s_{vois} est admissible et $F_1(s_{vois}) < F_1^*$, poser $s^*_{adm} := s_{vois}$, $F_1^* := F_1(s_{vois})$ et $\mu := 0$.
Si $F_2(s_{vois}) < F_2^*$, poser $s_{best} := s_{vois}$, $F_2^* := F_2(s_{vois})$ et $\mu := 0$.
Si i est un multiple de β alors

> Mettre à jour α selon la description faite en 4.4.3.6
> Si $\alpha > \frac{\alpha_{lim}}{2}$ et $F_1^* = \infty$ appliquer à s la procédure RETRACER_CABLES. Soit $\bar{s}$ la solution obtenue, poser $s := \bar{s}$.

Retourner à l'étape 1.

Algorithme: PPCCC_TABOU (SUITE)

Etape 5. *(Echec de l'algorithme)*
Si $s^*_{adm} = \emptyset$ l'algorithme n'a pas trouvé de solution admissible, STOP.

Etape 6. *(Dernière post-optimisation)*
Appliquer POST-OPT2 à s^*_{adm}. Soit $\bar{s}$ la solution obtenue.
Si $F_1(\bar{s}) < F_1^*$ alors poser $s^*_{adm} := \bar{s}$ et répéter l'étape 6.

Etape 7. La solution admissible s^*_{adm} est la solution fournie par PPCCC_TABOU.

Les deux procédures de post-optimisation sont appliquées à s_{vois} soit lorsque celle-ci est de coût inférieur à F_1^*, soit lorsqu'elle est admissible et que la solution s à partir de laquelle elle a été obtenue ne l'est pas. La procédure POST-OPT2 est également utilisée à la fin de la méthode tabou. Elle est alors appliquée à s^*_{adm}.

Nous avons testé deux variantes de la méthode tabou. La première correspond à la description faite ci-dessus. La deuxième est une variante plus rapide que nous avons appelée PPCCC_TR (TR pour tabou rapide). Cette variante est obtenue à partir de PPCCC_TABOU en supprimant les différents appels à la procédure POST-OPT2. Ainsi dans PPCCC_TR, seule la procédure POST-OPT1 est appliquée à l'étape 3 et l'étape 6 est supprimée.

4.5 Résultats numériques

4.5.1 Problèmes tests

Nous avons généré aléatoirement 18 types de problèmes. Chaque type est composé de 10 instances. Nous avons utilisé une procédure développée par [Tur97] pour construire nos problèmes tests. Cette procédure fonctionne de la manière suivante:

> Etant donné le nombre de sommets n ($n \geq 4$) du réseau G, ces sommets sont générés aléatoirement dans le carré $[0,10000]^2$.
>
> Une première série d'arêtes est choisie en construisant aléatoirement un arbre de cardinalité maximale dans G. D'autres arêtes sont ajoutées (elles sont choisies aléatoirement) de manière à ce que le nombre d'arêtes de G soit égal au maximum des deux valeurs $\lfloor \frac{3n}{2} \rfloor$ et $\frac{n(n-1)}{200}$. La longueur d'une arête (v_i, v_j) est égale à la partie entière inférieure de la distance euclidienne entre les sommets v_i et v_j. Les capacités des arêtes sont générées aléatoirement dans l'intervalle $[1, U]$ selon une loi uniforme où U est un entier positif qui borne supérieurement la valeur des capacités des arêtes.
>
> Etant donné un nombre K de câbles, les paires de sommets (s_k, t_k) de C, $(1 \leq k \leq K)$, sont choisies aléatoirement de telle sorte que s_k soit différent de

t_k. La section σ_k du câble k est choisie dans un intervalle $[1, \Sigma]$ selon une loi uniforme où Σ est un entier positif.

Nous avons travaillé avec des réseaux comportant 100, 200, 500 et 1000 sommets. Le fait que le nombre d'arêtes des réseaux soit égal au maximum des deux valeurs $\lfloor \frac{3n}{2} \rfloor$ et $\frac{n(n-1)}{200}$, permet d'obtenir des graphes de densité faible, mais jamais inférieure à 1%. Cette faible densité est caractéristique des réseaux associés aux problèmes de câblage rencontrés lors de la construction de centrales nucléaires.
Nous nous sommes arrangés de manière à ce que tous nos problèmes tests aient une solution admissible et que de plus, l'algorithme PPCCC_GLOUTON appliqué en fixant le paramètre i_{max} égal à 100 trouve toujours une solution admissible. Pour un problème donné, nous noterons U_{min} le plus petit entier tel qu'en choisissant les capacités des arêtes dans $[1, U_{min}]$ l'algorithme PPCCC_GLOUTON avec i_{max} fixé à 100 trouve une solution admissible.
En posant $U = U_{min}$ nous avons pu générer des problèmes où les capacités ont des valeurs faibles. D'autres problèmes, à priori plus faciles ont été construits en choisissant une valeur de U strictement supérieure à U_{min}. Deux valeurs différentes de Σ ont été utilisées pour générer nos instances, $\Sigma = 1$ et $\Sigma = 10$.

Les caractéristiques des 18 types de problèmes tests que nous avons traités sont présentées dans le tableau 4.1.

Types de problèmes	$n = \|V\|$	K	Σ	U
1	100	25	1	5
2	100	50	1	10
3	100	75	1	15
4	100	100	1	25
5	200	50	1	5
6	200	100	1	15
7	200	150	1	25
8	200	200	1	35
9	500	500	1	50
10	1000	1000	1	50
11	100	100	10	200
12	200	200	10	300
13	500	500	10	350
14	1000	1000	10	70
15	100	100	10	U_{min}
16	200	200	10	U_{min}
17	500	500	10	U_{min}
18	1000	1000	10	U_{min}

TAB. 4.1 -

Problèmes tests.

Lorsqu'il s'agira d'effectuer des tests sur les paramètres des algorithmes que nous avons présentés aux sections précédentes, les problèmes de type 10, 14 et 18 ne seront pas traités. Ces instances nécessitent en effet des temps de calcul très importants et ne seront étudiées que dans la partie consacrée aux comparaisons entre les algorithmes.

4.5.2 Tests numériques

4.5.2.1 PPCCC_GLOUTON

Nous avons vu en 4.4.2 que le nombre i_{max} d'ordres des câbles générés est le seul paramètre intervenant dans l'algorithme PPCCC_GLOUTON. Nous avons étudié l'influence de ce paramètre sur la qualité des solutions obtenues ainsi que sur le temps d'exécution de l'algorithme. Pour cela, les valeurs suivantes de i_{max} ont été testées: 50, 100, 200, 300, 400, 500, 1000, 3000, 5000. Les résultats obtenus figurent dans le tableau 4.2. E_{moyen} représente l'écart moyen (en %) par rapport à la borne inférieure $\underline{F}$ et E_{max} l'écart maximum. Ces écarts sont calculés sur les 10 instances de 15 types de problèmes (les types 10, 14 et 18 ne sont pas traités). La valeur N_{opt} correspond au nombre de fois qu'une solution de valeur égale à $\underline{F}$ est obtenue.

	Valeurs de i_{max}								
	50	100	200	300	400	500	1000	3000	5000
E_{moyen}	1.48	1.33	1.23	1.17	1.15	1.12	1.03	0.96	0.88
E_{max}	4.35	4.05	4.05	4.05	4.05	4.05	4.05	3.17	3.15
Temps moyen [s]	12.62	25.71	51.06	77.26	101.32	126.40	252.89	761.62	1283.4
N_{opt}	5	5	5	5	5	5	5	5	5

TAB. 4.2 -

Influence de i_{max} sur le comportement de PPCCC_GLOUTON

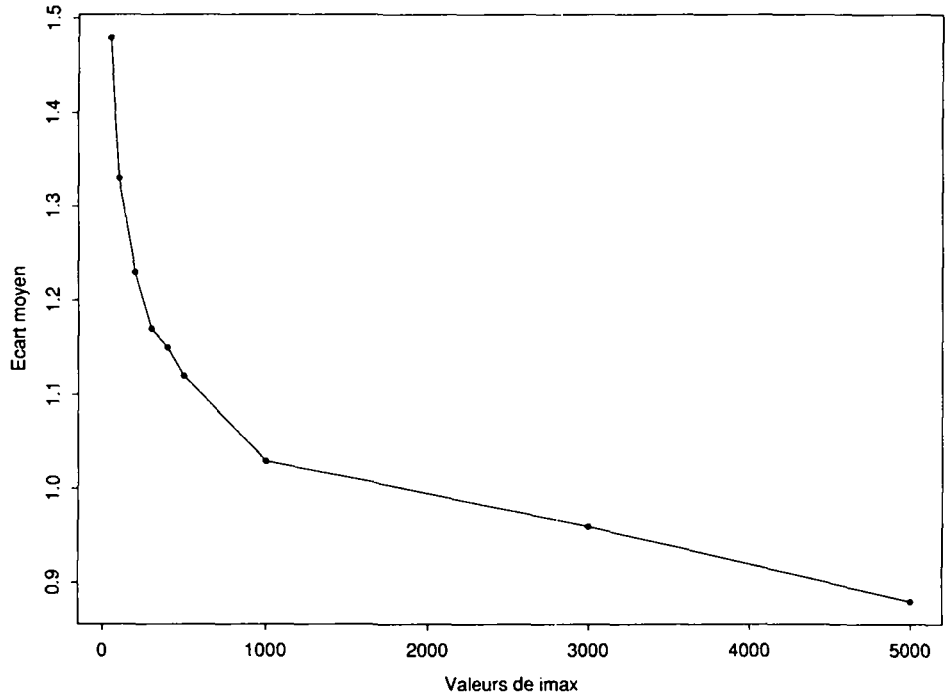


FIG. 4.17 -

Evolution des écarts moyens en fonction de la valeur de i_{max}.

En observant les temps moyens d'exécution des différentes variantes de PPCCC_GLOUTON figurant dans le tableau 4.2, nous pouvons remarquer que ceux-ci sont proportionnels à la valeur de i_{max}. Les écarts moyens par rapport à $\underline{F}$ sont faibles (moins de 1.5 %). Les bornes que nous avons obtenues par relaxation lagrangienne sont de bonne qualité. Elles égalent parfois la valeur de la solution optimale. Nous pouvons finalement remarquer qu'une forte augmentation de la valeur de i_{max} ne provoque qu'une légère amélioration de la qualité moyenne des solutions. En outre, plus la valeur de i_{max} est importante, plus ces améliorations sont faibles comme le montre le graphique de la figure 4.17.

Nous avons fixé la valeur de i_{max} à 1000. Ce choix nous semble être un assez bon compromis entre la qualité moyenne des solutions fournies par PPCCC_GLOUTON et le temps d'exécution de cet algorithme.

4.5.2.2 Variantes de la méthode tabou

Les tests que nous avons effectués avec les deux variantes PPCCC_TABOU et PPCCC_TR de la méthode tabou concernent le choix du paramètre nb_{vois} qui définit le nombre de solutions voisines de la solution courante examinées à chaque itération. Nous avons exprimé ce paramètre en fonction du nombre de câbles à traiter K. Les valeurs suivantes ont été testées: $\frac{K}{20}$, $\frac{K}{5}$, $\frac{K}{2}$ et K. Le tableau 4.3 contient les résultats que nous avons obtenus. Comme lors des tests effectués avec PPCCC_GLOUTON, les problèmes de type 10, 14 et 18, n'ont pas été traités.

	PPCCC_TABOU				*PPCCC_TR*			
nb_{vois}	E_{moy}	E_{max}	N_{opt}	Temps moyens	E_{moy}	E_{max}	N_{opt}	Temps moyens
$\frac{K}{20}$	0.49	2.60	5	299.07	0.89	4.03	5	25.17
$\frac{K}{5}$	0.49	2.48	5	359.36	0.67	3.173	5	87.79
$\frac{K}{2}$	0.50	2.76	5	489.1	0.64	2.89	5	258.33
K	0.52	2.676	5	614.02	0.63	2.89	5	654.71

TAB. 4.3 -

Influence de nb_{vois} sur le comportement de PPCCC_TABOU et PPCCC_TR

La valeur de nb_{vois} influence directement le temps moyen d'exécution de PPCCC_TR. Celui-ci semble être proportionnel au nombre de solutions voisines examinées à chaque itération. L'impact de nb_{vois} sur la qualité des solutions obtenues par PPCCC_TR est moins marqué. L'écart moyen par rapport à $\underline{F}$ varie peu pour les valeurs de nb_{vois} égales à $\frac{K}{5}$, $\frac{K}{2}$ et K. Une différence plus importante apparaît lorsque nb_{vois} est fixé à $\frac{K}{20}$.
L'influence de nb_{vois} sur le temps moyen d'exécution de PPCCC_TABOU est plus faible que celle exercée sur les temps de calcul de PPCCC_TR. Un tel comportement peut s'expliquer par l'utilisation de POST-OPT2. Cette procédure a une complexité de l'ordre de $O(K^2|V|^2)$. Cette complexité domine celle qui est nécessaire à chaque itération pour examiner nb_{vois} solutions et qui est de l'ordre de $O(nb_{vois}|V|^2) = O(K|V|^2)$. La qualité des solutions obtenues par PPCCC_TABOU n'est pas affectée par le choix de nb_{vois}. Cela laisse supposer que la procédure POST-OPT2 joue un rôle prépondérant dans le processus de

minimisation, la méthode tabou ne servant plus dès lors qu'à assurer la diversité des solutions auxquelles la procédure de post-optimisation est appliquée.
Lorsque nb_{vois} augmente, la différence entre les temps moyens de calcul des deux méthodes diminue. L'algorithme PPCCC_TABOU, qui est 10 fois plus lent que PPCCC_TR lorsque nb_{vois} vaut $\frac{K}{20}$, est plus rapide que ce dernier lorsque nb_{vois} vaut K. Plusieurs raisons expliquent un tel comportement. La première est que l'influence de POST-OPT2 sur le temps de calcul de PPCCC_TABOU diminue lorsque nb_{vois} augmente. En effet, pour un problème donné, le nombre d'appels à POST-OPT2 varie très peu lorsque nb_{vois} change. Le temps passé à effectuer cette procédure est donc sensiblement le même quelle que soit la valeur de nb_{vois}. La deuxième raison est, qu'en moyenne, PPCCC_TR effectue un plus grand nombre d'itérations que PPCCC_TABOU (pour $nb_{vois} = K$, 864 contre 731).

Finalement, nous avons fixé la valeur du paramètre nb_{vois} à $\frac{K}{5}$ pour les deux méthodes. Ce choix nous paraît être un bon compromis entre la qualité des solutions obtenues et les temps moyens d'exécution des algorithmes. De plus, le fait de choisir la même valeur pour PPCCC_TABOU et PPCCC_TR facilite la comparaison des deux méthodes.

4.5.2.3 Comparaisons

Nous avons comparé l'approche gloutonne aux deux variantes de la méthode tabou. Les 18 types de problèmes ont été pris en compte pour cette comparaison. Pour chaque type de problèmes, l'écart moyen E_{moy} et l'écart maximum E_{max} par rapport à la borne $\underline{F}$ sont donnés. Il s'agit là de moyennes et de valeurs maximum calculées sur les 10 instances de chaque type de problèmes. Ces écarts sont mesurés en %. Le nombre de solutions N_{opt} (parmi les 10 instances de chaque type) dont la valeur égale celle de $\underline{F}$ est également présenté ainsi que les temps moyens d'exécution (en secondes) de chacune des méthodes.

	PPCCC_GLOUTON				*PPCCC_TABOU*				*PPCCC_TR*			
Pb	E_{moy}	E_{max}	N_{opt}	Temps	E_{moy}	E_{max}	N_{opt}	Temps	E_{moy}	E_{max}	N_{opt}	Temps
1	0.007	0.061	4	3.65	0.007	0.061	4	2.1	0.007	0.061	4	1.12
2	0.273	1.470	1	10.77	0.001	0.003	1	6.29	0.200	1.149	1	3.28
3	0.486	1.429	0	17.62	0.014	0.140	0	14.02	0.083	0.314	0	4.82
4	0.222	0.800	0	23.55	0.001	0.002	0	14.91	0.084	0.298	0	6.98
5	0.100	0.502	0	23.33	0.001	0.002	0	16.03	0.012	0.093	0	9.06
6	0.256	0.850	0	47.60	0.003	0.031	0	33.68	0.027	0.109	0	12.86
7	0.399	0.719	0	72.68	0.002	0.018	0	72.06	0.056	0.139	0	21.74
8	0.668	2.081	0	95.79	0.058	0.406	0	106.58	0.137	0.467	0	29.13
9	0.507	0.935	0	670.86	0.003	0.013	0	1053.04	0.093	0.223	0	190.61
10	0.413	0.694	0	3365.09	0.007	0.016	0	6576.67	0.081	0.171	0	1066.08
11	1.000	2.108	0	23.04	0.887	1.661	0	13.98	1.131	1.975	0	6.06
12	0.625	1.226	0	96.64	0.426	1.021	0	72.70	0.550	1.698	0	30.28
13	0.831	1.259	0	671.20	0.427	0.708	0	769.76	0.527	0.819	0	252.24
14	4.677	5.366	0	3224.22	2.254	3.111	0	29640.55	2.940	3.918	0	2185.82
15	1.883	4.048	0	22.60	1.455	2.48	0	19.71	1.771	3.173	0	9.25
16	1.326	2.422	0	92.18	0.780	1.371	0	104.94	0.915	1.7	0	32.52
17	1.703	3.342	0	657.06	0.828	1.146	0	1293.82	1.064	1.790	0	267.98
18	13.164	21.544	0	2600.54	6.587	10.950	0	41423.80	8.123	12.723	0	3451.39
Moyennes	1.586			651.02	0.763			4513.04	0.989			421.18
Total ou maximum		21.544	5			10.950	5			12.723	5	

TAB. 4.4 -

Comparaison entre PPCCC_GLOUTON, PPCCC_TABOU et PPCCC_TR

Les problèmes dans lesquels la section des câbles vaut 1 semblent être plus faciles à traiter

que les autres. Quelle que soit la méthode utilisée, c'est pour ces problèmes que les écarts moyens par rapport à $\underline{F}$ sont les plus faibles. Ces derniers augmentent lorsque la section des câbles est choisie aléatoirement dans l'intervalle $[1, 10]$. Ils sont les plus importants lorsque la valeur de U est égale à U_{min}.
Du point de vue de la qualité des solutions, PPCCC_TABOU est l'algorithme le plus performant. Pour les 18 types de problèmes, il obtient chaque fois l'écart moyen le plus faible. La deuxième variante du tabou se comporte bien également. Les écarts moyens qu'elle obtient sont généralement inférieurs ou égaux à ceux produits par PPCCC_GLOUTON. Seule exception, les problèmes du type 11 où l'algorithme glouton est légèrement meilleur.
Au niveau des temps moyens de calcul, PPCCC_TABOU est la méthode la plus lente. Même si, pour des problèmes de petite taille (100 et 200 sommets) elle est généralement plus rapide que PPCCC_GLOUTON, ses temps moyens d'exécution deviennent tellement importants pour les grandes instances (500 et 1000 sommets) qu'elle est en moyenne 10 fois plus lente que PPCCC_TR et 7 fois moins rapide que PPCCC_GLOUTON. La cause d'un tel comportement réside dans la complexité de la procédure POST-OPT2 utilisée dans PPCCC_TABOU. Les deux autres méthodes ont des temps moyens de calcul qui sont à peu près du même ordre, la deuxième variante de la méthode tabou étant cependant plus rapide: 420 secondes en moyenne contre 650 pour PPCCC_GLOUTON.
Pour les problèmes les plus difficiles (problèmes de type 15, 16, 17 et 18), les deux variantes de la méthode tabou ont des temps de calcul qui sont plus importants que lors du traitement de problèmes de même taille où les contraintes de capacité sont moins restrictives (problèmes de type 11, 12, 13, 14 par exemple). Nous avons observé qu'en moyenne un nombre plus élevé d'itérations est effectué pour les problèmes les plus durs. Cela vient en partie du fait que pour cette catégorie de problèmes, les deux variantes de la méthode tabou ont plus de difficultés à trouver une solution qui soit admissible. Les cas où il faut faire appel à la procédure RETRACER_CABLE sont alors plus fréquents. Avec PPCCC_GLOUTON, les temps moyens d'exécution sont pratiquement identiques pour une taille fixée de problème. Ils peuvent même être plus faibles lorsque les problèmes sont plus difficiles. Cela peut s'observer en comparant le temps moyen de calcul des problèmes du type 18 à celui des problèmes du type 14. Ce comportement s'explique par le fait que le nombre de solutions admissibles visitées est moins important lorsque les problèmes sont difficiles. Dans ce cas, la majorité des ordres des câbles générés aléatoirement ne conduisent pas à des solutions admissibles. Pour chacun de ces ordres, un certain nombre de câbles ne peuvent ainsi pas être tracés, ce qui contribue à augmenter la vitesse de l'algorithme.

4.6 Liens entre le PPCCC et le BWP

Le PPCCC est lié de très près au "Bandwidth Packing Problem" (BWP). Le BWP introduit par Cox et al. [Cox91] est un problème apparaissant dans le domaine des télécommunications. Les données de ce problème sont pratiquement identiques à celles du PPCCC. Le réseau de communication est modélisé à l'aide d'un graphe $G = (V, E)$, où à chaque arête (v_i, v_j), sont associés un coût unitaire c_{ij} et une capacité u_{ij}. Des appels doivent être acheminés entre différents points du réseau. Chaque appel doit être transmis d'un point s_k (la source de l'appel) à un point t_k (sa destination). Les informations contenues dans un appel ne peuvent pas être divisées et doivent suivre le même cheminement dans le réseau. Nous noterons par $C = \{(s_1, t_1), ..., (s_K, t_K)\}$ l'ensemble des liaisons demandées. A chaque appel k, $(k = 1, ..., K)$, est associée une valeur σ_k qui correspond à la

capacité minimum que doit avoir une ligne de transmission (i.e. une arête de G) pour que l'appel puisse l'utiliser. Il s'agit là du volume de l'appel exprimé en unités de capacité. Finalement, un revenu r_k est associé à chaque appel k, $(k = 1, ..., K)$. Ce revenu, ainsi que le fait de travailler avec des coûts unitaires constituent les seules différences entre les données du PPCCC et celles du BWP.

Lorsqu'un chemin P de G reliant s_k à t_k est attribué à un appel k, cela induit un profit noté Π_k et défini de la façon suivante:

$$\Pi_k = r_k - \sigma_k \sum_{(v_i,v_j)\in P} c_{ij}$$

En général, le nombre d'appels est trop important pour que tous puissent être transmis. Le BWP consiste donc à choisir un sous-ensemble C^- des appels contenus dans C et à attribuer à chaque appel de C^- un chemin dans G respectant les contraintes de capacité et reliant l'origine à la destination de l'appel. L'objectif est bien entendu de maximiser le profit total des appels acheminés (i.e. des appels de C^-). En introduisant pour chaque appel k de C une variable booléenne z_k qui vaut 1 si l'appel k est dans C^- et zéro sinon, nous pouvons formuler le BWP en termes de programmation linéaire 0_1.

$$\textbf{MAX} \quad \sum_{k=1}^{K} r_k z_k - \sum_{k=1}^{K} (\sigma_k \sum_{\substack{(v_i,v_j)\in E \\ i<j}} c_{ij} x_{ij}^k)$$

$$\textbf{s.c.} \quad \sum_{k=1}^{K} \sigma_k x_{ij}^k \leq u_{ij} \quad \forall\, (v_i, v_j) \in E,\ i < j \qquad \textbf{(1)}$$

$$\sum_{\substack{(v_i,v_j)\in E \\ j<i}} x_{ji}^k + \sum_{\substack{(v_i,v_j)\in E \\ i<j}} x_{ij}^k = 2y_i^k \quad \forall\, k \in \{1, ..., K\}, \forall\, v_i \neq s_k,\ t_k \qquad \textbf{(2)}$$

$$\sum_{\substack{(v_i,v_j)\in E \\ j<i}} x_{ji}^k + \sum_{\substack{(v_i,v_j)\in E \\ i<j}} x_{ij}^k = z_k \quad \forall\, k \in \{1, ..., K\}, \forall\, v_i = s_k \text{ ou } t_k \qquad \textbf{(3')}$$

$$x_{ij}^k = 0 \text{ ou } 1 \qquad \begin{array}{l} \forall\, k \in \{1, ..., K\} \\ \forall\, (v_i, v_j) \in E,\ i < j \end{array} \qquad \textbf{(4)}$$

$$y_i^k = 0 \text{ ou } 1 \qquad \forall\, k \in \{1, ..., K\}, \forall\, v_i \in V \qquad \textbf{(5)}$$

$$z_k = 0 \text{ ou } 1 \qquad \forall\, k \in \{1, ..., K\} \qquad \textbf{(6)}$$

Les variables x_{ij}^k et y_i^k sont identiques à celles utilisées en 4.2.2 pour décrire le PPCCC en termes de programmation linéaire 0_1. Par rapport à cette description, la fonction objectif a changé, la contrainte **(3)** a été remplacée par la contrainte **(3')** et la contrainte **(6)** a été ajoutée. Les contraintes **(1)**, **(2)**, **(4)** et **(5)** sont identiques.

Plusieurs méthodes de résolution ont été utilisées pour aborder le BWP. Des heuristiques basées sur une méthode tabou ont été développées par Laguna et al. [Lag93] et par Anderson et al. [And93]. Cox et al. ont mis au point un algorithme génétique [Cox91], Parker et al. [Par95] utilisent une méthode de génération de colonne et Park et al. [Par96] ont développé des techniques de résolution basées sur la programmation en nombres entiers.

Etant donné un ensemble $C^- \subseteq C$ de K^- $(0 < K^- \leq K)$ câbles, nous appellerons sous-problème du BWP le problème, noté BWPS, consistant à trouver le meilleur cheminement possible des K^- câbles de C^- dans G. Sans perte de généralité, nous pouvons supposer que $C^- = \{(s_1, t_1), ..., (s_{K^-}, t_{K^-})\}$ et qu'ainsi $z_k = 1$ pour $k = 1, ..., K^-$ et $z_k = 0$ pour $k = K^- + 1, ..., K$. Le BWPS peut alors se formuler de la manière suivante:

$$\textbf{MAX} \quad \sum_{k=1}^{K^-} r_k - \sum_{k=1}^{K^-} \sigma_k \sum_{\substack{(v_i, v_j) \in E \\ i<j}} c_{ij} x_{ij}^k$$

$$\textbf{s.c.} \quad \sum_{k=1}^{K^-} \sigma_k x_{ij}^k \leq u_{ij} \quad \forall\, (v_i, v_j) \in E,\ i < j \qquad (1^-)$$

$$\sum_{\substack{(v_i, v_j) \in E \\ j<i}} x_{ji}^k + \sum_{\substack{(v_i, v_j) \in E \\ i<j}} x_{ij}^k = 2y_i^k \quad \forall\, k \in \{1, ..., K^-\}, \forall\, v_i \neq s_k,\ t_k \qquad (2^-)$$

$$\sum_{\substack{(v_i, v_j) \in E \\ j<i}} x_{ji}^k + \sum_{\substack{(v_i, v_j) \in E \\ i<j}} x_{ij}^k = 1 \quad \forall\, k \in \{1, ..., K^-\}, \forall\, v_i = s_k \text{ ou } t_k \qquad (3^-)$$

$$x_{ij}^k = 0 \text{ ou } 1 \qquad \begin{array}{l} \forall\, k \in \{1, ..., K^-\} \\ \forall\, (v_i, v_j) \in E,\ i < j \end{array} \qquad (4^-)$$

$$y_i^k = 0 \text{ ou } 1 \qquad \forall\, k \in \{1, ..., K^-\}, \forall\, v_i \in V \qquad (5^-)$$

Le premier terme de la fonction objectif est constant et peut donc être ignoré lorsque la maximisation s'effectue. Nous pouvons alors reformuler le BWPS comme un problème de

minimisation:

$$\textbf{MIN} \quad \sum_{k=1}^{K^-} \sum_{\substack{(v_i,v_j)\in E \\ i<j}} \sigma_k c_{ij} x_{ij}^k$$

$$\textbf{s.c.} \quad (1^-), (2^-), (3^-), (4^-), (5^-)$$

Ce problème diffère du PPCCC uniquement par la fonction objectif. Dans le cas du PPCCC, une arête (v_i, v_j) appartenant à un chemin reliant s_k à t_k possède un coût de c_{ij} alors que ce coût est de $\sigma_k c_{ij}$ dans le cas du BWPS. Tous les développements que nous avons présentés aux sections 4.3.1, 4.3.2 et 4.4 peuvent être adaptés au BWPS.

En relaxant les contraintes de capacités (contraintes de type (1^-)), le sous-problème de Lagrange peut alors être décomposé en K^- problèmes de plus court chemin indépendants les uns des autres. Le problème de plus court chemin associé à l'appel k est alors de la forme:

$$\textbf{MIN} \quad \sum_{\substack{(v_i,v_j)\in E \\ i<j}} \sigma_k(c_{ij}+\lambda_{ij})x_{ij}^k$$

$$\textbf{s.c.} \quad \sum_{\substack{(v_i,v_j)\in E \\ j<i}} x_{ji}^k + \sum_{\substack{(v_i,v_j)\in E \\ i<j}} x_{ij}^k = 2y_i^k \quad \forall\, v_i \neq s_k,\ t_k$$

$$\sum_{\substack{(v_i,v_j)\in E \\ j<i}} x_{ji}^k + \sum_{\substack{(v_i,v_j)\in E \\ i<j}} x_{ij}^k = 1 \quad \forall\, v_i = s_k \text{ ou } t_k$$

$$x_{ij}^k = 0 \text{ ou } 1 \quad \forall\, (v_i,v_j)\in E,\ i<j$$

$$y_i^k = 0 \text{ ou } 1 \quad \forall\, v_i \in V$$

Ce problème est quasiment identique à celui obtenu en relaxant les contraintes de capacité du PPCCC. La seule différence intervient dans le coût de l'arête (v_i, v_j). Ce coût est de $\sigma_k(c_{ij}+\lambda_{ij})$ dans le cas du BWPS alors qu'il est de $c_{ij}+\sigma_k\lambda_{ij}$ avec le PPCCC.

Si les contraintes de chemin sont relaxées (contraintes de type (2^-) et (3^-)), le sous-problème de Lagrange peut se décomposer en $|E|$ problèmes indépendants de sac de montagne à variables 0-1. Seuls le coût des objets et leur nombre changent lorsque la relaxation des contraintes de chemin est faite à partir du BWPS au lieu du PPCCC. En partant du BWPS, on dispose de K^- objets. Le coût de l'objet k $(1 \leq k \leq K^-)$ dans le problème du sac de montagne associé à l'arête (v_i, v_j) est alors de $\sigma_k c_{ij} - \mu_{ik} - \mu_{jk}$. Dans le cas du PPCCC, K objets peuvent être utilisés et le coût de l'objet k $(1 \leq k \leq K)$ dans le problème du sac de montagne associé à l'arête (v_i, v_j) est de $c_{ij} - \mu_{ik} - \mu_{jk}$.

Les algorithmes PPCCC_GLOUTON, PPCCC_TABOU et PPCCC_TR peuvent être appliqués à la résolution du BWPS. Il suffit pour cela de tenir compte du fait que le coût associé à une arête dépend du type d'appel (ou de câble) traité.

Une façon d'aborder la résolution du BWP consiste à sélectionner un ensemble $C^- \subseteq C$ d'appels et à résoudre le BWPS en utilisant par exemple une des méthodes que nous avons décrites dans ce chapitre. Ce procédé peut se répéter un certain nombre de fois, en choisissant chaque fois un nouvel ensemble C^-.

4.7 Conclusion

Le PPCCC est un problème qui nous a été initialement proposé par EDF (Electricité De France) pour résoudre un problème d'optimisation d'itinéraires de câbles. Ce problème est étroitement lié à d'autres problèmes issus de la pratique et apparaissant par exemple dans le domaine des télécommunications.
Nous avons développé une méthode tabou pour traiter le PPCCC. Deux variantes ont été testées. La première variante, PPCCC_TABOU est très performante du point de vue de la qualité des solutions obtenues mais nécessite des temps de calcul importants lorsque les instances traitées sont de grande taille. La deuxième variante, PPCCC_TR, est beaucoup plus rapide lorsqu'elle est appliquée à de gros problèmes. Les solutions qu'elle obtient sont en revanche un peu moins bonnes que celles produites à l'aide de PPCCC_TABOU. Les algorithmes PPCCC_TABOU et PPCCC_TR utilisent deux procédures de post-optimisation que nous avons élaborées et qui peuvent également être utilisées pour améliorer les solutions fournies à l'aide d'autres techniques de résolution.

Nous avons comparé les deux variantes de notre méthode tabou à une approche gloutonne, PPCCC_GLOUTON, développée précédemment. Nos heuristiques se comparent favorablement à cette dernière méthode, principalement au niveau de la qualité des solutions. Nous avons pu également exhiber un exemple simple pour lequel l'approche gloutonne ne peut jamais trouver une solution qui soit admissible. Une telle solution existe cependant pour cet exemple, et nos algorithmes parviennent à l'atteindre.

Les deux heuristiques ainsi que les procédures de post-optimisation que nous avons présentées dans ce chapitre sont également décrites dans [Cos98]. Elles sont utilisées par EDF où elles servent de base de comparaison afin de tester un nouvel algorithme actuellement en cours de développement.

Conclusion générale

Transporter des matières premières, des personnes, de l'information, assurer des services chez des clients, effectuer des travaux d'entretien sur des réseaux routiers ou sur des lignes électriques, etc. toutes ces activités sont le lot quotidien de nombreuses entreprises. Celles-ci ne sauraient être compétitives sans une bonne gestion des ressources allouées à ce type de problèmes. La recherche opérationnelle dispose de techniques qui permettent de traiter certains aspects de ces problèmes. Certaines d'entre elles ont un caractère très général et peuvent être appliquées à une grande variété de problèmes. Un des objectifs de ce travail a consisté à adapter ce type de techniques à deux problèmes liés au transport de biens: le problème de tournées sur les arcs avec capacité (PTAC) et le problème de plus courts chemins avec capacité (PPCCC).

Dans le cas du PTAC, deux heuristiques basées sur des méthodes de recherche locale ont été développées. La première est une adaptation de la méthode tabou. La deuxième est une méthode de descente à voisinage variable. Pour mettre en oeuvre cette dernière, nous avons défini une nouvelle classe de voisinages pour le PTAC. Les cas non orienté et orienté du PTAC ont été abordés à l'aide de ces deux techniques. Le PTAC orienté étant peu étudié, nous avons adapté une borne inférieure initialement proposée pour le cas non orienté, afin d'évaluer la qualité des solutions produites par nos algorithmes. Ceux-ci se sont montrés très efficaces. Dans le cas non orienté, nos heuristiques comptent parmi les méthodes les plus performantes disponibles actuellement. Dans le cas orienté, la valeur des solutions obtenues se situe en moyenne à moins de 4% de celle de la borne que nous avons développée. Des écarts importants (plus de 40%) ont cependant été enregistrés sur certaines instances. Des progrès peuvent donc encore être réalisés, soit en développant de nouvelles bornes inférieures prenant en compte la répartition des demandes dans le réseau (ce que ne fait pas la borne $\vec{LB}1$), soit en essayant de mettre en oeuvre de nouvelles heuristiques plus performantes encore.

Un aspect plus pratique du PTAC a été abordé en traitant un problème réel. Nous avons décrit de manière détaillée le processus de modélisation d'un problème de confection d'horaires de bateaux. Celui-ci a été transformé en PTAC avec fenêtres de temps. Certaines exigences liées à ce problème ont été difficiles à quantifier et à traduire en termes de contraintes. C'est pourquoi nous nous sommes attachés à la fois au développement d'algorithmes permettant la construction automatique d'un horaire et à l'élaboration d'outils offrant à l'utilisateur la possibilité d'agir sur les horaires obtenus. Les premières solutions que nous avons construites, bien qu'inutilisables en pratique, ne sont pas très éloignées des horaires en vigueur actuellement. Grâce à elles, nous avons pu constater que notre modèle était capable de prendre en compte de manière satisfaisante les principales contraintes liées à la confection des horaires de la CGN. Des développements sont actuellement en cours afin de rendre nos solutions réellement applicables.

Le dernier problème que nous avons traité dans ce travail est le problème de plus courts chemins avec capacité. Nous avons adapté à ce problème une méthode tabou. Des procédures de post-optimisation ont également été développées. Deux variantes de la méthode tabou ont été mises en oeuvre. Elles se différencient par l'usage plus ou moins intensif d'une des procédures de post-optimisation. Chacune d'elles fournit des solutions qui comparées à celles obtenues à l'aide d'une approche gloutonne sont en moyenne meilleures. L'écart est d'autant plus grand que la taille des problèmes traités est importante. Ce comportement est intéressant dans la mesure où les problèmes auxquels doit faire face EDF sont justement des problèmes de taille importante.

Le champ d'application des problèmes liés au transport de biens est énorme. Une véritable demande existe en matière de techniques mathématiques ou autres capables d'améliorer la gestion de tels problèmes. Cette demande est une motivation supplémentaire pour les chercheurs qui les pousse à se montrer encore plus inventifs et à élaborer des outils toujours plus efficaces. Nul doute qu'à l'avenir l'importance croissante de ces problèmes ira de pair avec le développement de méthodes de résolution de plus en plus sophistiquées et de plus en plus performantes.

Index des algortihmes

Chapitre 1. Problèmes de tournées sur les arcs: revue des principales heuristiques

Problème du postier chinois

Problème du postier rural

Outils algorithmiques

Problème de tournées sur les arcs avec capacité

Chapitre 2. Nouveaux développements pour la résolution du PTAC

Le PTAC non orienté

Le PTAC orienté

Chapitre 3. Une application des PTAC: la confection d'horaires de bateaux

Chapitre 4. Optimisation d'itinéraire de câbles

Bibliographie

[Aar51] T. van Aardenne-Ehrenfest and N. G. de Bruijn. *Circuits and Trees in Oriented Linear Graphs.* Simon Stevin 28, 203-217, 1951.

[Aar97] E. Aarts and J. K. Lenstra (Ed.). *Local Search in Combinatorial Optimization.* Wiley, West Sussex, 1997.

[Agg95] A. K. Aggarwal, M. Oblak and R. R. Vemuganti. *A Heuristic Solution Procedure for Multicommodity Integer Flows.* Computers and Operations Research 22 (10), 1075-1087, 1995.

[Ahu93] R. Ahuja, T. Magnanti and J.Orlin. *Networks, Flows, Theory, Algorithms, Applications.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[And93] C. A. Anderson, K. Fraughnaugh, M. Parker and J. Ryan. *Path Assignement for Call Routing: An Application of Tabu Search.* Annals of Operations Research 41, 301-312, 1993.

[Ass87] A. A. Assad, W. L. Pearn and B. L. Golden. *The Capacitated Postman Problem: Lower Bounds and Solvable Cases.* American Journal of Mathematical and Management Science 7 (1,2), 63-88, 1987.

[Ass95] A. A. Assad and B. L. Golden. *Arc Routing Methods and Applications.* Chapitre 5 de "Network Routing". Handbooks in Operations Research and Management Science, M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser (Eds). North-Holland, Amsterdam, 375-483, 1995.

[Bal88] M. O. Ball and M. J. Magazine. *Sequencing of Insertions in Printed Circuit Board Assembly.* Operations Research 36 (2), 192-201, 1988.

[Bar99] F. Bardet. *Conception des Horaires de la CGN: un Problème d'Arc Routing avec Fenêtres de Temps.* Travail pratique de diplôme, DMA-ROSE, Ecole Polythechnique Fédérale de Lausanne, 1999.

[Bea83] J. E. Beasley. *Route First-Cluster Second Methods for Vehicle Routing.* Omega 11, 403-408, 1983.

[Bel97a] J. M. Belenguer, E. Benavent and F. Cognata. *A Metaheuristic for the Capacitated Arc Routing Problem.* Manuscrit non publié, 1997.

[Bel97b] J. M. Belenguer and E. Benavent. *A Cutting Plane Algorithm for the Capacitated Arc Routing Problem.* Manuscrit non publié, 1997.

[Bel74] E. Beltrami and L. Bodin. *Networks and Vehicles Routing for Municipal Waste Collection.* Networks 4 (1), 65-94, 1974.

[Ben90] E. Benavent, V. Campos, A. Corberán, and E. Mota. *The Capacitated Arc Routing Problem: A Heuristic Algorithm.* Qüestiió 14 (1-3), 107-122, 1990.

[Ben92] E. Benavent, V. Campos, A. Corberán, and E. Mota. *The Capacitated Arc Routing Problem: Lower Bounds.* Networks 22, 669-690, 1992.

[Ber83] C. Berge. *Graphes*, Gauthier-Villars, Paris, 1983.

[Cha84] L. Chapleau, J. A. Ferland, G. Lapalme and J.-M. Rousseau. *A Parallel Insert Method for the Capacitated Arc Routing Problem.* Operations Research Letters 3 (2), 95-99, 1984.

[Chr73] N. Christofides. *The Optimum Travesal of a Graph.* Omega 1 (6), 719-732, 1973.

[Chr76] N. Christofides. *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem.* Report N° 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, 1976.

[Chr81] N. Christofides, V. Campos, A. Corberán and E. Mota. *An Algorithm for the Rural Postman Problem.* Imperial College Report IC.O.R.81.5, 1981.

[Chr86] N. Christofides, V. Campos, A. Corberán and E. Mota. *Algortihms for the Rural Postman Problem on a Directed Graph.* Mathematical Programming Study 26, 155-166, 1986.

[Cla64] G. Clarke and J. Wright. *Scheduling of Vehicules from a Central Depot to a Number of Delivery Points.* Operations Research 12, 568-581, 1964.

[Cor94] A. Corberán and J. M. Sanchis. *A Polyhedral Approach to the Rural Postman Problem.* European Journal of Operational Research, 79, 95-242, 1994.

[Cor97] A. Corberán, R, Marti and A. Romero. *A Tabu Search Algorithm for the Mixed Rural Postman Problem.* Research Report, Dept. of Satistics and OR, University of Valencia, Spain, 1997.

[Cos98] M.-C. Costa, A. Hertz and M. Mittaz. *Bounds and Heuristics for the Shortest Capacitated Paths Problem.* Rapport technique, ORWP 98/05, Ecole Polytechnique Fédérale de Lausanne, 1998.

[Cox91] L. A. Cox, L. Davis and Y. Qui. *Dynamic Anticipatory Routing in Circuit-Switched Telecommunications Networks.* in L. Davis (Ed.), Handbook of Genetic Algorithms, VanNostrand Reinhold, New York.

[Cro58] G. A. Croes. *A Method for Solving Travelling-Salesman Problems.* Operations Research 6, 791-812, 1958.

[DeA81] J. S. DeArmon. *A Comparison of Heuristics for the Capacitated Chinese Postman Problem.* Master's Thesis, University of Maryland at College Park, 1981.

[Dro87] M. Dror, H. Stern and P. Trudeau. *Postman Tour on a Graph with Precedence Relations on Arcs.* Networks 17 (3), 283-294, 1987.

[Dro97] M. Dror, A. Langevin. *A Generalized Traveling Salesman Problem Approach to the Directed Clustered Rural Postman Problem.* Transportation Science 31 (2), 1997.

[Edm67] J. Edmonds. *Optimum Branching.* Journal of Research National Bureau of Standards, section B, 71, 233-240,1967.

[Edm73] J. Edmonds and E. L. Johnson. *Matching, Euler Tours and the Chinese Postman.* Mathematical Programming 5, 88-124, 1973.

[Egl94] R. W. Eglese. *Routing Winter Gritting Vehicles.* Discrete Applied Mathematics 48 (3), 231-244, 1994.

[Egl95] R. W. Eglese and A. N. Letchford. *The Rural Postman Problem with Deadline Classes.* Working Paper, Dept. Management Science, Lancaster University, 1995.

[Egl96] R. W. Eglese and L. Y. O. Li. *A Tabu Search Based Heuristic for Arc Routing with a Capacity Constrint and Time Deadline.* Meta-Heuristic: Theory and Applications, 633-649, 1996.

[Eis95] H. A. Eiselt, M. Gendreau and G. Laporte. *Arc Routing Problems, Part I: The Chinese Postman Problem.* Operations Research 43, 231-242, 1995.

[Eul1736] L. Euler. *Solutio Problematis ad Geometrian Situs Pertinentis.* Commentarii academiae scientarum Petropolitanae 8, 128-140, 1736.

[Fis81a] M. L. Fisher and R. Jaikumar. *A Generalized Assignment Heuristic for Vehicule Routing.* Networks 11, 109-124, 1981.

[Fis81b] M. L. Fisher. *The Lagrangian Relaxation Method for Solving Integer Programming Problems.* Management Science 27, 1-18, 1981.

[Fle90] H. Fleischner. *Eulerian Graphs and Related Topics, Part 1, Volume 1.* Annals of Discrete Mathematics 45, North-Holland, Amsterdam, 1990.

[For62] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, New Jersey, 1962.

[Fre78] G. N. Frederickson, M. S. Hecht and C. E. Kim. *Approximation Algorithms for some Routing Problems.* SIAM Journal on Computing 7 (2), 178-193. 1978.

[Fre79] G. N. Frederickson. *Approximation Algorithms for some Postman Problems.* Journal of the ACM 26 (3), 538-554, 1979.

[Gen92] M. Gendreau, A. Hertz and G. Laporte. *New Insertion and Post-optimization Procedures for the Travelling Salesman Problem.* Operations Research 40, 1086-1094, 1992.

[Gen94] M. Gendreau, A. Hertz and G. Laporte. *A Tabu Search Heuristic for the Vehicle Routing Problem.* Management Science 40, 1276-1290, 1994.

[Geo74] A. M. Geoffrion. *Lagrangean Relaxation For Integer Programming.* Mathematical Programming Study 2, 82-114, 1972.

[Ghi2000] G. Ghiani and G. Laporte. *A Branch-and-Cut Algorithm for the Undirected Rural Postman Problem.* à paraître dans Mathematical Programming, 2000.

[Glo89] F. Glover. *Tabu Search, Part I.* ORSA Journal on Computing 1, 190-206, 1989.

[Glo90] F. Glover. *Tabu Search, Part II.* ORSA Journal on Computing 2, 4-32, 1990.

[Glo93] F. Glover and M. Laguna. *Tabu Search.* in Modern Heuristic Techniques for Combinatorial Problems. C. Reeves (Ed.). Blackwell, Oxford, 70-150.

[Gol81] B. L. Golden and R. T. Wong. *Capacitated Arc Routing Problems.* Networks 11 (3), 305-315, 1981.

[Gol83] B. L. Golden, J. S. DeArmon and E. K. Baker. *Computational Experiments with Algorithms for a Class of Routing Problems.* Computers and Operations Research 10 (1), 47-59, 1983.

[Gre94] P. Greistorfer. *Algorithms and Implementations for the Mixed Capacitated Chinese Postman Problem.* Working Paper 33, Department of Business, University of Graz, 1994.

[Gua62] M. Guan. *Graphic Programming Using Odd and Even Points.* Chinese Mathematics 1, 273-277, 1962.

[Gua84] M. Guan. *On the Windy Postman Problem.* Discrete Applied Mathematics. 9, 41-46, 1984.

[Han86] P. Hansen. *The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming.* présenté au "Congress on Numerical Methods in Combinatorial Optimization", Capri, Italie, 1986.

[Han97] P. Hansen and N. Mladenović. *An Introduction to Variable Neighborhood Search.* Les Cahiers du GERAD, G-97-51, 1997.

[Hel74] M. Held, P. Wolfe and H. P. Crowder. *Validation of Subgradient Optimisation.* Mathematical Programming 6, 62-88, 1974.

[Her99] A. Hertz, G. Laporte and P. Nanchen. *Improvement Procedures for the Undirected Rural Postman Problem* INFORMS Journal on Computing 11, 53-62, 1999.

[Her2000] A. Hertz, G. Laporte and M. Mittaz. *A Tabu Search Heuristic for The Capacitated Arc Routing Problem.* à paraître dans Operations Research, 2000.

[Hir92] R. Hirabayashi, Y. Sarutawatari and N. Nishida. *Tour Construction Algorithm for the Capacitated Arc Routing Problem.* Asia-Pacific Journal of Operational Research 9, 155-175, 1992.

[Jan93] K. Jansen. *Bounds for the General Capacitated Routing Problem.* Networks 23, 165-173, 1993.

[Ken78] J. Kennington. *A Survey of Linear Cost Multicommodity Networks Flows.* Operations Research 26, 209-236, 1978.

[Lag93] M. Laguna and F. Glover. *Bandwidth Packing: A Tabu Search Approach.* Management Science 39 (4), 492-500, 1993.

[Len76] J. K. Lenstra and A. H. G. Rinnooy Kan. *On General Routing Problems.* Networks 6 (3), 273-280, 1976.

[Let96] A. N. Letchford. *Polyhedral Results for Some Constrained Arc-Routing Problem.* Ph.D. Thesis, Department of Management Science, Lancaster University, 1996.

[Li92] L. Y. O. Li. *Vehicle Routing for Winter Gritting.* Ph.D. Dissertation, Department of OR & OM, Lancaster University, 1992.

[Lin88] Y. Lin and Y.Zhao. *A New Algorithm for the Directed Chinese Postman Problem.* Computers and Opererations Research 15 (6), 577-584. 1988.

[Luk87] A. Lukka and L. Salminen. *Comparison of Three Heuristics for an Arc Routing Problem.* Research Report 6, Departement of Information Technology, Lappenranta University of Technology, Finland, 1987.

[Man84] U. Manber and S. Israni. *Pierce Point Minmization and Optimal Torch Path Determination in Flame Cutting.* Journal of Manufacturing Systems 3 (1), 81-89, 1984.

[Mar90] S. Martello and P. Toth. *Lower Bounds and Reduction Procedures for the Bin Packing Problem.* Discrete Applied Mathematics 28, 59-70, 1990.

[McB97] R. McBride and J. Mamer. *Solving Multicommodity Flow Problems with a Primal Embedded Network Simplex Algorithm.* INFORMS Journal on Computing 9, 154-162, 1997.

[Mid93] M. Middendorf and F. Pfeiffer. *On the Complexity of the Disjoint Paths Problem.* Combinatorica 13 (1), 97-107, 1993.

[Min75] M. Minoux. *Résolutions des Problèmes de Multiflots en Nombres Entiers dans les Grands Réseaux.* RAIRO Operations Research 3, 21-40, 1975.

[Min78] E. Minieka. *Optimisation Algorithms for Networks and Graphs.* Marcel Dekker, Inc., New York, 1978.

[Min79] E. Minieka. *The Chinese Porstman Problem for Mixed Networks.* Management Science 25 (7), 643-648, 1979.

[Mla97] N. Mladenović and P. Hansen. *Variable Neighborhood Search.* Computers and Operations Research 24, 1097-1100, 1997.

[Orl74] C. S. Orloff. *A Fundamental Problem in Vehicle Routing.* Networks 4, 35-64, 1974.

[Pap76] C. H. Papadimitriou. *On the Complexity of Edge Traversing.* Journal of the ACM 23 (3), 544-554, 1976.

[Par95] M. Parker and J. Ryan. *A Column Generation Algorithm for Bandwidth Packing.* Telecommunications Systems 2, 185-196, 1995.

[Par96] K. Park, S. Kang and S. Park. *An Integer Programming Approach to the Bandwidth Packing Problem.* Management Science 42, 1277-1291, 1996.

[Pea88] W. L. Pearn. *New Lower Bounds for the Capacitated Arc Routing Problem.* Networks 18, 181-191,1988.

[Pea89] W. L. Pearn. *Approximate Solutions for the Capacitated Arc Routing Problem.* Computers and Operations Research 16 (6), 589-600, 1989.

[Pea91] W. L. Pearn. *Augment-Insert Algorithms for the Capacitated Arc Routing Problem.* Computers and Operations Research 18 (2), 189-198, 1991.

[Pea94] W. L. Pearn and M. L. Li. *Algorithms for the Windy Postman Problem.* Computers and Operations Research 21 (6), 641-651, 1994.

[Ree93] C. R. Reeves (Ed.). *Modern Heuristic Techniques for Combinatorial Optimization.* Blackwell, Oxford, 1993.

[Roc94] Y. Rochat and F. Semet. *A Tabu Search Approach for Delivering Pet Food and Flour in Switzerland.* Journal of the Operational Research Society 45, 1233-1246, 1994.

[Roc95] Y. Rochat and E. Taillard. *Probabilistic Diversification and Intensification in Local Search for Vehicle Routing.* Journal of Heuristics 1, 147-167, 1995.

[Ros75] G. T. Ross and R. M. Soland. *A Branch and Bound Algorithm for the Generalized Assignment Problem.* Mathematical Programming 8, 91-103, 1975.

[Ros77] D. J.Rosenkrantz, R. E. Stearns and P. M. Lewis III. *An Analysis of Several Heuristics for the Travelling Salesman Problem.* SIAM Journal on Computing 6, 563-581, 1977.

[Roy89] S. Roy and J. M. Rousseau. *The Capacitated Canadian Postman Problem.* INFOR 27 (1), 58-73, 1989.

[Sol87] M. M. Solomon. *Algorithms for the Vehicule Routing and Scheduling Problems with Time Window Constraints.* Operations Research, 35 (2), 254-265, 1987.

[Tur97] I. Turki. *Optimisation d'Itinéraires de Câbles Electriques.* Mémoire d'ingénieur, CNAM-IIE, Paris, 1997.

[Ulu85] G. Ulusoy. *The Fleet Size and Mix Problem for Capacitated Arc Routing.* European Journal of Operational Research 22 (3), 329-337, 1985.

[Vyg95] J. Vygen. *NP-Completeness of somme Edge-Disjoint Paths Problems.* Discrete Applied Mathematics 61, 83-90, 1995.

[Win88] Z. Win. *Contributions to Routing Problems.* PhD Dissertation, Universität Augsburg, 1988.

[Win89] Z. Win. *On the Windy Postman Problem on Eulerian Graphs.* Mathematical Programming 44, 97-112, 1989.

Curriculum vitae

Nom:	Michel MITTAZ
Date de naissance:	23 juin 1972
Lieu d'origine:	Chermignon (VS)
État civil:	Célibataire
Adresse:	Ch. du Scex, 1971 Grimisuat

Formation

1991	Certificat de maturité de type C, Lycée-Collège des Creusets, Sion
1996	Diplôme d'ingénieur mathématicien de l'Ecole Polythechnique Fédérale de Lausanne (EPFL)
1999	Certificat du cours postgrade en recherche opérationnelle et statistique de l'EPFL intitulé *Aide à la décision en management et technologie*

Activité professionnelle

1996-1999	Assistant en Recherche Opérationnelle à l'Ecole Polytechnique Fédérale de Lausanne (Profs. D. de Werra et A. Hertz, Département de Mathématiques)