ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Le problème du postier chinois mixte avec pénalités pour les virages

par
Nathalie Perrier

Sciences de la gestion

Mémoire présenté en vue de l'obtention
du grade de maître ès sciences
(M.Sc.)

Février 2001

À Michel

Sommaire

Dans ce mémoire, trois nouvelles heuristiques pour le problème du postier chinois mixte avec pénalités pour les virages sont proposées. Étant donné un ensemble de clients représentés par les liens d'un réseau, le problème consiste à desservir chaque client au moins une fois de façon à minimiser la longueur totale de la tournée accomplie par un véhicule et la somme des paramètres de pénalité associés aux virages effectués par le véhicule. Plus général que le problème d'origine sans étiquettes de pénalité attachées aux virages, le problème du postier chinois mixte avec pénalités pour les virages s'applique naturellement dans les villes aux prises avec la planification des tournées pour le service de voirie.

La première heuristique proposée est basée sur une transformation développée à l'origine pour le problème de transbordement ainsi que sur une méthode de recherche à voisinage variable. En plus de résoudre le problème du postier chinois mixte avec pénalités pour les virages, l'heuristique permet de traiter les cas non orienté, orienté et mixte du problème du postier rural avec pénalités pour les virages. Les autres heuristiques sont des adaptations au cas mixte de méthodes récemment développées pour deux problèmes différents du postier chinois avec pénalités pour les virages.

Est également proposée dans ce mémoire une adaptation au cas mixte d'une borne inférieure initialement utilisée pour le problème du postier rural dirigé avec pénalités pour les virages. Cette borne permet d'évaluer la qualité des solutions produites par les heuristiques sur des ensembles de nouvelles instances générées aléatoirement. Les résultats obtenus sur des données tirées de la littérature montrent qu'aucune des trois heuristiques ne s'avère aussi efficace qu'une méthode existante suggérée pour résoudre un problème plus général.

Remerciements

En premier lieu, mes directeurs de recherche, Messieurs Gilbert Laporte et Patrick Soriano. Merci d'abord pour la proposition de recherche qui a donné naissance à ce mémoire. Merci aussi pour les judicieux conseils, les suggestions, les expériences partagées, les remises en question et le soutien financier. Merci surtout pour la liberté accordée du début à la fin dans l'élaboration de ce travail.

Je remercie sincèrement M. David Soler pour les travaux de recherche sur le sujet et les données qui ont servi aux tests numériques. Je tiens aussi à remercier M. Michel Mittaz pour les codes de programmation.

J'éprouve une vive reconnaissance envers Mesdames Monique Dupont, Claire Dubois et Nicole Duchesne pour les nombreuses trouvailles de même que tous les membres de la bibliothèque de l'École des Hautes Études Commerciales et celle de mathématiques et d'informatique de l'Université de Montréal qui ont contribué de près ou de loin à la recherche de documents.

J'aimerais exprimer ma gratitude au personnel du Centre de recherche sur les transports pour son aide et son soutien ainsi qu'au personnel du laboratoire informatique de l'École des Hautes Études Commerciales et celui du centre Virtuose pour sa disponibilité.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Le domaine des transports regorge d'applications à des problèmes qui, aujourd'hui, consistent non seulement à transporter des marchandises et des personnes mais, entre autres, à desservir des clients, à développer et à entretenir des infrastructures, à transmettre de l'information et à distribuer de l'énergie. Les problèmes liés au transport surgissent dans tous les secteurs de l'économie, dans le domaine privé et le domaine public, mais ce n'est qu'après une longue tradition et de nombreux succès à résoudre ces problèmes dans le secteur privé que les applications au secteur public sont récemment devenues l'objet de travaux de recherche plus répandus. Des économies notables peuvent être réalisées en améliorant la planification des activités de transport ce qui explique les efforts déployés par la recherche opérationnelle et les mathématiques appliquées dans ce domaine.

On distingue deux grandes familles de problèmes quant à la façon de modéliser les clients à desservir. La première regroupe les problèmes dans lesquels les clients à desservir sont associés aux sommets d'un réseau et que l'on appelle problèmes de tournées sur les sommets (PTS). Cette représentation est utilisée lorsque chaque point de service est isolé dans un réseau. Les PTS ont longtemps fait l'objet d'études et plusieurs méthodes efficaces ont été développées pour résoudre ces problèmes. Le problème du voyageur de commerce (PVC) constitue le pilier des PTS. Lorsqu'un ensemble important de clients à desservir est concentré sur une même rue, il est parfois plus naturel de représenter les ensembles de clients à l'aide d'arcs ou d'arêtes dans un réseau. Ces problèmes sont appelés problèmes de tournées sur les arcs (PTA). Dans certaines applications, les rues entières peuvent aussi nécessiter un service comme pour la livraison du courrier, la collecte des déchets, le déneigement et les tournées d'autobus scolaires. Comme pour les PTS, de nombreuses variantes de PTA

existent allant du problème du postier chinois (PPC) au problème de tournées sur les arcs avec contrainte de capacité (PTAC). Les PTA ont cependant reçu moins d'attention que leurs homologues sur les sommets.

Dans le problème du postier chinois mixte (PPCM), les clients, représentés par des arcs et des arêtes, sont desservis par un véhicule opérant à partir d'un point de départ dans le réseau. Chaque client doit être visité au moins une fois par le véhicule, et la tournée doit débuter et se terminer au même point. Le problème consiste à déterminer un circuit de longueur totale minimum traversant au moins une fois chaque client du réseau.

La tournée d'un PPCM peut renfermer des virages à gauche, des virages à droite et des virages en «U», les seuls coûts considérés dans le PPCM étant les longueurs associées aux liens. Or, dans de nombreuses applications, ces types de virage sont difficile, voire impossible à réaliser ou encore ils peuvent être interdits. Par exemple, dans le cas du déneigement des rues, les virages à droite sont préférables aux virages à gauche ou aux traverses afin de laisser les rues dégagées.

Les problèmes d'optimisation combinatoire se divisent en deux classes importantes. Il y a les problèmes qui peuvent être facilement résolus en des temps de calcul raisonnables dits polynomiaux par rapport à la taille de ces problèmes. Pour les résoudre, il n'y a pas d'erreur possible car on dispose généralement de méthodes exactes appelées algorithmes polynomiaux. Par contre, d'autres problèmes sont plus difficiles à résoudre. Qualifiés de NP-complets ou NP-difficiles, ces problèmes ne peuvent pas être résolus de façon exacte en des temps de calcul raisonnables même pour des tailles de problèmes modérées. Des heuristiques sont alors développées afin de trouver pour ces problèmes des solutions dont l'écart par rapport à une borne inférieure sur la valeur d'une solution optimale soit le plus petit possible.

Dans ce mémoire, trois heuristiques pour le problème du postier chinois mixte avec pénalités pour les virages (PPCM-PV) sont proposées. Le PPCM-PV est une généralisation du PPCM consistant à associer des pénalités aux virages. Par exemple, dans le cas du déneigement, si les déplacements à favoriser dans une tournée sont les virages à droite, les pénalités possibles associées aux virages pourraient être de 0 pour un virage à droite, 2 pour une traverse de rue, 4 pour un virage à gauche et 5 ou 10 pour un virage en «U». D'autres applications pourraient exiger l'interdiction des virages en «U» en leur associant une pénalité très élevée. Le problème consiste à déterminer une tournée de façon à ce que chaque lien soit visité au moins une fois et que la somme de la longueur totale de la tournée et des pénalités associées aux virages inclus dans la tournée soit minimisée.

Ce problème apparaît surtout dans les villes qui doivent planifier le service de voirie. L'entretien, le déneigement et le nettoyage des rues ainsi que l'enlèvement des ordures en sont des exemples. D'une part, les citadins sont de plus en plus exigeants en matière de service de voirie. D'autre part, les municipalités veulent amoindrir les coûts reliés à ce service. Un besoin marqué pour l'amélioration de la planification des services de voirie est donc présent.

Dans les deux premiers chapitres, différents PTA, avec et sans pénalité pour les virages, et les principales méthodes utilisées pour les résoudre sont passés en revue. Une formulation mathématique du PPCM-PV est ensuite présentée au troisième chapitre. Au quatrième chapitre, trois heuristiques sont proposées afin de résoudre le problème. Les algorithmes y sont également expliqués en détail. Finalement, des comparaisons basées sur un ensemble de nouvelles instances ainsi que les résultats sur des instances tirées de la littérature sont exposés au cinquième chapitre.

Chapitre 1

PTA sans pénalité pour les virages: revue des principales méthodes

La solution au problème des sept ponts de Königsberg marque les débuts de la théorie des graphes. Le problème consistait à déterminer un trajet débutant et se terminant au même point, et traversant exactement une fois chacun des sept ponts de la rivière Pregel à Königsberg, aujourd'hui appelée Kaliningrad, dans l'est de la Prusse (figure 1.1).

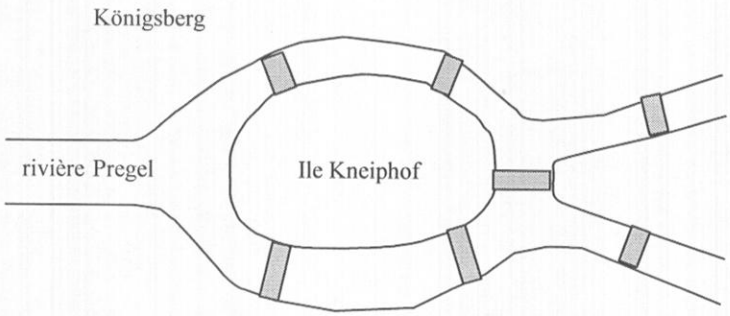


Figure 1.1: Le problème des sept ponts de Königsberg

En 1736, Leonhard Euler [Eul1736] montra que le problème des sept ponts de Königsberg est impossible à résoudre car les degrés de tous les sommets du graphe qui représentent les masses de terre sont impairs (figure 1.2). Le degré d'un sommet v dans un graphe est le nombre de liens du graphe ayant une extrémité en v [Ber73]. En fait, Euler prouva qu'un graphe non orienté et connexe, i.e. un graphe dans lequel il existe un parcours d'un sommet à tout autre sommet du graphe, est eulérien si et

seulement si tous ses sommets sont de degré pair. Cette condition est appelée condition d'unicursalité.

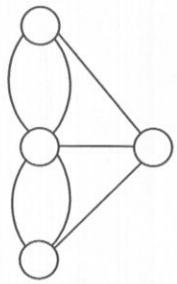


Figure 1.2: Graphe représentant le problème des sept ponts de Königsberg

1.1 Le problème du postier chinois

En 1962, Meigu Guan [Gua62] s'est intéressé à un problème plus général que celui des sept ponts de Königsberg. Le problème, appelé problème du postier chinois (PPC) en l'honneur de ce mathématicien chinois, consistait à déterminer une tournée de facteur de longueur totale minimum, qui commence et prend fin au même point, et qui couvre au moins une fois chaque segment à desservir.

Dans cette section, les principales méthodes de résolution du PPC dans les graphes non orientés, orientés et mixtes sont étudiées.

1.1.1 Le PPC dans un graphe non orienté

Soit le graphe $G = (V, E)$, un graphe connexe et non orienté. L'ensemble $V = \{v_1, v_2, \ldots, v_n\}$ représente les sommets et l'ensemble E est celui des arêtes du graphe. À chaque arête (v_i, v_j) de E est associé un coût non négatif c_{ij} représentant la longueur de l'arête (v_i, v_j). On suppose ici que le coût associé à chaque arête du graphe est identique quel que soit le sens dans lequel cette arête est parcourue.

Lorsque les coûts dépendent de l'orientation des arêtes, il est alors question du problème du postier chinois asymétrique [Min79].

Il existe deux cas selon que le graphe G est eulérien ou non. Si G est eulérien, le problème se ramène à la détermination d'un cycle eulérien dans G. Tel que défini dans [Ber73], un cycle est une chaîne (i.e. une séquence $(a_1, a_2, \ldots, a_t)$ d'arêtes de sorte que chaque arête de la séquence ait une extrémité en commun avec l'arête précédente et l'autre extrémité en commun avec l'arête suivante) dans laquelle la même arête ne figure pas deux fois dans la séquence et les deux sommets aux extrémités de la chaîne coïncident. Un cycle eulérien dans un graphe non orienté $G = (V, E)$ est un cycle qui utilise toute arête de E une fois et une fois seulement.

L'algorithme suivant permet de construire un cycle eulérien. Il s'agit de l'algorithme END-PAIRING proposé par Edmonds et Johnson [Edm73]. D'autres méthodes pour construire un cycle eulérien sont présentées dans [Kau67] et [Fle91].

Algorithme END-PAIRING

Étape 1. Déterminer un premier cycle C_1 dans G. Si C_1 couvre chaque arête de E, STOP.

Étape 2. Choisir un sommet v appartenant à C_1 et incident à une arête ne figurant pas dans C_1. En commençant au sommet v, construire un deuxième cycle C_2 tel que C_1 et C_2 n'aient pas d'arête commune.

Étape 3. Fusionner les cycles C_1 et C_2 pour former le cycle C_1. Si C_1 couvre chaque arête de E, STOP. Sinon, retourner à l'étape 2.

Le fonctionnement de l'algorithme END-PAIRING est illustré à l'aide du graphe non orienté de la page suivante (figure 1.3). Si le sommet a est choisi comme extrémités du premier cycle, il peut en résulter le cycle C_1 formé de la séquence d'arêtes suivante: (a, b), (b, e), (e, d) et (d, a). Posons maintenant $v = b$. Le deuxième

cycle C_2 construit correspond à la chaîne (b, c), (c, f), (f, e), (e, g) et (g, b). Les cycles C_1 et C_2 sont ensuite fusionnés comme suit. À partir du sommet initial a du cycle C_1, suivre les sommets du cycle jusqu'à ce que le sommet commun b soit atteint. Traverser ensuite le cycle C_2 au complet. Terminer la visite des sommets du cycle C_1 à partir du sommet b. Chacune des arêtes du graphe étant couverte, le cycle eulérien ainsi formé correspond à la suite d'arêtes (a, b), (b, c), (c, f), (f, e), (e, g), (g, b), (b, e), (e, d) et (d, a).

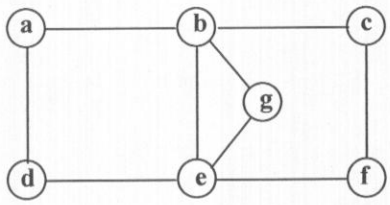


Figure 1.3: Graphe servant à illustrer l'algorithme END-PAIRING

Si G n'est pas eulérien, il existe au moins une arête dans G qui doit être traversée plus d'une fois afin de satisfaire la condition d'unicursalité. Dans le contexte des problèmes de tournées sur les arcs, toute traverse supplémentaire de lien est appelée passage à vide, le lien ayant déjà été desservi lors de la première traverse. Puisque la somme des coûts associés aux arêtes de G est constante, le problème du postier chinois se ramène à minimiser la longueur des passages à vide nécessaires pour rendre G eulérien afin de construire un cycle eulérien dans G.

Les passages à vide peuvent être déterminés en résolvant un problème de couplage parfait de coût minimum dans un graphe auxiliaire complet G_0 ayant pour sommets l'ensemble V_0 des sommets de degré impair dans G. Un couplage parfait de coût minimum dans G_0 est un ensemble d'arêtes de G_0 deux à deux non adjacentes et incidentes à tous les sommets de G_0 tel que la somme des arêtes de G_0 appartenant au couplage est minimum.

L'algorithme suivant permet de résoudre le problème du postier chinois dans un graphe non orienté. Il s'agit de l'algorithme présenté dans [Edm73] et [Orl74]. Christofides [Chr73] propose également d'utiliser un couplage parfait pour résoudre le PPC.

Algorithme PPC_NON_ORIENTÉ

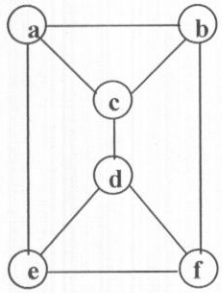
Étape 1. Déterminer l'ensemble V_0 des sommets de degré impair dans G. Construire le graphe complet G_0 dont l'ensemble des sommets est V_0. Le coût de chaque arête (v_i, v_j) de G_0 est égal à la longueur de la plus courte chaîne SC_{ij} entre les sommets v_i et v_j dans G.

Étape 2. Trouver un couplage parfait de coût minimum dans le graphe G_0.

Étape 3. Pour chaque arête (v_i, v_j) du couplage optimal, dédoubler dans G toutes les arêtes appartenant à la plus courte chaîne SC_{ij} reliant les sommets v_i et v_j dans G.

Étape 4. Déterminer un cycle eulérien en appliquant END-PAIRING sur G.

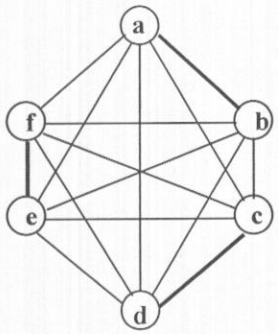
La figure 1.4 de la page suivante illustre le fonctionnement de l'algorithme à l'aide du graphe non orienté G de la figure 1.4 (a). Le coût associé à chaque arête de G est égal à 1. Tous les sommets de G sont de degré impair. La matrice des longueurs des plus courtes chaînes entre toutes les paires de sommets de degré impair apparaît à la figure 1.4 (b). Une solution optimale au problème de couplage parfait de coût minimum correspond aux arêtes dessinées en gras sur la figure 1.4 (c). Par conséquent, il faut dupliquer l'arête (a, b) correspondant à la plus courte chaîne entre les sommets a et b ainsi que les arêtes (c, d) et (e, f) correspondant respectivement aux plus courtes chaînes entre les sommets c et d et les sommets e et f. La figure 1.4 (d) illustre le graphe eulérien G^*. Un cycle eulérien dans G^* correspond à la suite d'arêtes (a, b), (b, c), (c, d), (d, f), (f, e), (e, d), (d, c), (c, a), (a, b), (b, f), (f, e) et (e, a).



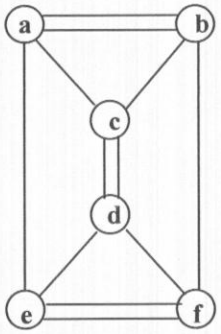
(a) Graphe initial G

	a	b	c	d	e	f
a	0	1	1	2	1	2
b	1	0	1	2	2	1
c	1	2	0	1	2	2
d	2	2	1	0	1	1
e	1	2	2	1	0	1
f	2	1	2	1	1	0

(b) Matrice des longueurs des plus courtes chaînes



(c) Graphe auxiliaire G_0



(d) Graphe eulérien G^*

Figure 1.4: Illustration de l'algorithme pour le PPC non orienté

1.1.2 Le PPC dans un graphe orienté

Le problème du postier chinois dans un graphe orienté apparaît dans les contextes où des rues à sens unique doivent être desservies. Soit $G = (V, A)$, un graphe orienté et fortement connexe où A désigne l'ensemble des arcs. Un graphe orienté est fortement connexe si pour toute paire (v_i, v_j) de sommets distincts, il existe un chemin (i.e. une chaîne $(a_1, a_2, \ldots, a_t)$ d'arcs telle que pour tout arc a_i, $1 \leq i < t$, le sommet terminal de a_i coïncide avec le sommet initial de a_{i+1}) reliant v_i à v_j et un chemin reliant v_j à v_i [Ber73]. Lorsque G est orienté, on appelle circuit eulérien le circuit traversant exactement une fois chaque arc de A. Un circuit est un cycle

$(a_1, a_2, \ldots, a_t)$ d'arcs tel que pour $1 \leq i < t$, l'extrémité terminale de l'arc a_i coïncide avec l'extrémité initiale de l'arc a_{i+1}.

Comme dans le cas non orienté, le graphe G est eulérien si le nombre de fois qu'on entre dans un sommet v_i de V est égal au nombre de fois qu'on en sort, noté ici $d^-(i) = d^+(i)$. Le graphe G est alors dit symétrique. Par conséquent, si pour un sommet v_i de V, le nombre d'arcs de G arrivant dans v_i est supérieur au nombre d'arcs sortant de v_i, soit $d^-(i) > d^+(i)$, il faut alors traverser à vide certains arcs quittant le sommet v_i. De même, si le nombre d'arcs de G quittant le sommet v_i est supérieur au nombre d'arcs arrivant dans v_i, soit $d^+(i) > d^-(i)$, certains arcs arrivant dans v_i devront alors être dupliqués. Le problème du postier chinois dans un graphe orienté revient donc, comme dans le cas non orienté, à minimiser le nombre de passages à vide nécessaires pour rendre G eulérien.

Deux cas peuvent être considérés selon que le graphe G est symétrique ou non.

Si G est symétrique, un circuit eulérien peut être déterminé en adaptant l'algorithme END-PAIRING pour le cas orienté simplement en tenant compte de la direction des arcs dans la construction des circuits. L'algorithme de Fleury pour le cas non orienté s'adapte aussi facilement (voir Christofides [Chr75], p.202). L'algorithme décrit dans [van51] et [Edm73] permet également de trouver un circuit eulérien dans un graphe symétrique.

Si G n'est pas symétrique, certains arcs doivent être copiés de façon à rendre le graphe eulérien. Les arcs à copier peuvent être déterminés en résolvant un problème de flot à coût minimum dans un graphe auxiliaire G_0. Soit $G' = (I \cup J \cup T, A')$ un graphe orienté où I est l'ensemble des sommets émetteurs engendrant des unités de flot destinées aux sommets récepteurs de J et tel qu'à chaque arc (v_i, v_j) de A' correspondent un coût unitaire de transport du flot transmis par cet arc ainsi que les bornes inférieure et supérieure du flot sur cet arc. Le problème du flot à coût

minimum (aussi connu sous la rubrique problème de transbordement) consiste à déterminer le nombre d'unités de flot passant par chacun des arcs de A' de façon à satisfaire les demandes des sommets récepteurs et tel que le coût total soit minimum. Le nombre d'unités de flot passant par chacun des arcs de A' doit respecter les bornes inférieure et supérieure sur le flot transmis par les arcs du réseau. De plus, pour tout sommet de transbordement $v_i \in T$, le nombre de fois qu'on entre dans v_i doit être égal au nombre de fois qu'on en sort, souvent appelées contraintes de conservation du flot.

L'algorithme qui suit est similaire à celui présenté dans [Edm73]. La valeur optimale du flot x_{ij} traversant chaque arc représente le nombre de copies additionnelles à ajouter au graphe G afin de le rendre symétrique.

Algorithme PPC_ORIENTÉ

Étape 1. Pour tout sommet $v_i \in V$, calculer la valeur $f_i = d^-(i) - d^+(i)$. Si tous les f_i sont nuls, aller à l'étape 5.

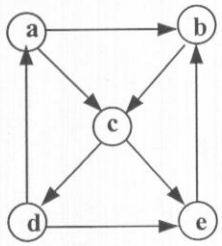
Étape 2. Déterminer l'ensemble des sommets émetteurs $I = \{v_i \in V : f_i > 0\}$ et l'ensemble des sommets récepteurs $J = \{v_j \in V : f_j < 0\}$. Construire le réseau $G_0 = (V, A_0)$ en ajoutant à G les arcs virtuels $(\cdot, v_i)$ associés aux sommets émetteurs v_i et les arcs virtuels $(v_j, \cdot)$ associés aux sommets récepteurs v_j. Attribuer à chaque arc virtuel $(\cdot, v_i)$ une disponibilité de valeur f_i et à chaque arc virtuel $(v_j, \cdot)$ une demande de valeur $|f_j|$. Attribuer à chaque arc de $A_0 \cap A$ une borne supérieure infinie.

Étape 3. Trouver un flot compatible de coût minimum dans G_0 satisfaisant les demandes des sommets récepteurs.

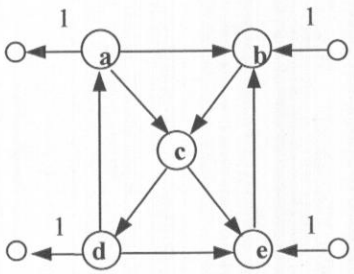
Étape 4. Soit x_{ij} la valeur du flot traversant un arc de A_0. Ajouter à G x_{ij} copies de chaque arc (v_i, v_j) appartenant à $A_0 \cap A$.

Étape 5. Déterminer un cycle eulérien en appliquant l'algorithme END-PAIRING modifié pour le cas orienté sur G.

Le fonctionnement de l'algorithme est illustré à l'aide du graphe orienté G de la figure 1.5 (a). Le coût associé à chaque arc de G est égal à 1. D'abord, les valeurs f_i sont $f_a = -1, f_b = 1, f_c = 0, f_d = -1$ et $f_e = 1$. Par conséquent, les sommets émetteurs sont les sommets b et e et les sommets récepteurs sont les sommets a et d. Le réseau G_0 correspondant au modèle graphique de cet exemple apparaît à la figure 1.5 (b). Le réseau précise les disponibilités et les demandes associées aux arcs virtuels.



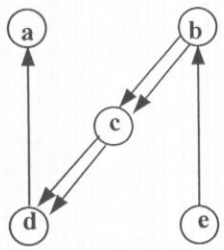
(a) Graphe initial G



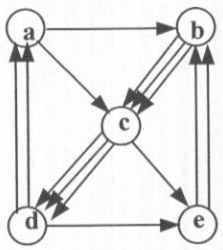
(b) Graphe auxiliaire G_0

Figure 1.5: Illustration de l'algorithme pour le PPC orienté

Un flot compatible de coût minimum est représenté sur la figure 1.5 (c). La figure 1.5 (d) correspond au graphe G^* construit à partir du flot optimal de la figure 1.5 (c). Ce graphe est complètement orienté et symétrique, donc eulérien.



(c) Un flot optimal dans G_0



(d) Graphe G^*

Figure 1.5 (suite): Illustration de l'algorithme pour le PPC orienté

Les arcs à copier peuvent aussi être déterminés en résolvant un problème de transport dans un graphe auxiliaire biparti complet $G_0 = (V_0, A_0)$ tel que proposé dans [Bel74] et dans [Orl74]. Le problème de transport est un cas particulier du problème de transbordement dans lequel aucune capacité n'est attribuée aux arcs du réseau et tel que $T = \varnothing$. Un graphe est biparti si l'ensemble de ses sommets peut être partitionné en deux groupes de sorte que deux sommets du même groupe ne soient jamais adjacents. Un groupe est formé des sommets émetteurs de l'ensemble $I = \{v_i \in V : f_i > 0\}$, l'autre groupe, de l'ensemble des sommets récepteurs $J = \{v_j \in V : f_j < 0\}$. Le coût d'un arc (v_i, v_j) dans G_0 est égal à la longueur du plus court chemin PCC_{ij} reliant v_i à v_j dans G. À titre d'exemple, la figure 1.6 illustre le réseau de transport associé au graphe G de la figure 1.5 (a). Les nombres en gras situés près des arcs virtuels correspondent aux disponibilités et aux demandes. Les autres nombres se rapportent aux longueurs des plus courts chemins. Les arcs dessinés en gras sur la figure1.6 correspondent à une solution optimale au problème de transport.

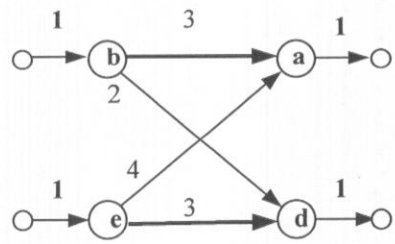


Figure 1.6: Réseau de transport associé au graphe de la figure 1.5 (a)

Soit x_{ij} le nombre d'unités transportées dans G_0 entre les sommets v_i et v_j dans la solution optimale au problème de transport. Pour chaque arc $(v_i, v_j) \in A_0$, il faut ajouter à G x_{ij} copies des arcs appartenant au plus court chemin PCC_{ij} qui relie v_i à v_j dans G. Le graphe eulérien construit correspond au graphe G^* de la figure 1.5 (c).

Lin et Zhao [Lin88] proposent un algorithme basé sur la formulation d'un programme linéaire en nombres entiers pour le PPC orienté à partir de laquelle ils déduisent le programme dual. En appliquant le théorème des écarts complémentaires

aux modèles des problèmes primal et dual, les auteurs identifient les conditions nécessaires et suffisantes pour qu'une solution au PPC orienté soit optimale. En premier lieu, l'algorithme consiste à générer une solution initiale satisfaisant un sous-ensemble de ces conditions. La solution initiale de la relaxation n'est pas nécessairement admissible. Cette solution est ensuite modifiée à chaque itération de l'algorithme jusqu'à ce que toutes les conditions nécessaires et suffisantes soient satisfaites.

1.1.3 Le PPC dans un graphe mixte

Soit $G = (V, E \cup A)$, un graphe mixte et fortement connexe. Le graphe G possède un circuit eulérien si et seulement s'il est pair et équilibré [For62]. Un graphe mixte est dit équilibré si pour tout sous-ensemble $S \subseteq V$ la différence D entre le nombre d'arcs orientés de S vers V et le nombre d'arcs orientés de V vers S est inférieure ou égale au nombre d'arêtes reliant S à $V \setminus S$.

La figure 1.7 illustre un exemple où la deuxième condition n'est pas satisfaite. Pour le sous-ensemble $S = \{a, c, e\}$, $D = 2$ et il n'y a aucune arête reliant S à $V \setminus S$ pour rétablir l'équilibre.

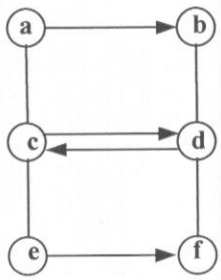


Figure 1.7: Graphe pair mais non équilibré

Si G est pair, il y a deux cas à considérer selon que G est équilibré ou non.

Si G est équilibré, le PPC mixte équivaut à trouver un circuit eulérien dans G. Pour ce faire, les arêtes de G sont orientées afin de rendre le graphe symétrique.

L'algorithme proposé par Papadimitriou [Pap76] pour un problème relié au PPC, le problème du graphe eulérien minimum (PGEM), permet de déterminer si G est équilibré et d'orienter ses arêtes. Étant donné un graphe mixte $G = (V, E \cup A)$, le PGEM consiste à trouver un graphe mixte et eulérien $G^* = (V, E \cup A \cup A^*)$ contenant le graphe G et tel que l'ensemble des nouveaux arcs A^* soit de cardinalité minimum. L'algorithme se résume à résoudre un problème du flot maximal. Dans le problème du flot maximal, aucun coût n'est associé aux unités de flot circulant dans le réseau et celui-ci n'admet qu'un seul sommet émetteur, la source S, et un seul sommet récepteur, le puits P. L'objectif est d'acheminer la plus grande quantité possible de flot de S à P tout en respectant les bornes inférieure et supérieure sur le flot transmis par les arcs du réseau.

Le PPC mixte revient ensuite à trouver un circuit eulérien dans le graphe orienté à l'aide de l'algorithme END-PAIRING modifié pour le cas orienté ou encore en adaptant l'algorithme décrit dans [van51] pour le cas mixte [Edm73]. L'algorithme suivant correspond à celui présenté dans [Pap76].

Algorithme PPC_MIXTE_PAIR_ÉQUILIBRÉ

Étape 1. Soit n_i le nombre d'arcs de A^* quittant le sommet $v_i \in V$ (si $n_i > 0$) ou arrivant dans v_i (si $n_i < 0$). Soit s_k le sommet initial de l'arête orientée $e_k = (v_i, v_j)$. Pour tout $v_i \in V$, poser $n_i = 0$ et pour toute arête $e_k = (v_i, v_j) \in E$, poser $s_k = 0$.
Dans le graphe partiel $G_1 = (V, A)$ engendré par A, calculer $f_i = d^-(i) - d^+(i)$ pour tout sommet $v_i \in V$. Dans le graphe partiel $G_2 = (V, E)$ engendré par E, calculer le degré d_i de chaque sommet $v_i \in V$.

Algorithme PPC_MIXTE_PAIR_ÉQUILIBRÉ

Étape 1. (suite)

Poser $G_3 = G_2$. Pour chaque sommet $v_i \in V$ tel que $d_i < f_i$, retirer toutes les arêtes incidentes à v_i dans G_3, poser $f_i = f_i - d_i$, et, pour tout sommet $v_j \in V$ adjacent à v_i dans G_2, poser $f_j = f_j + 1$ et $s_j = v_i$. Pour chaque sommet $v_i \in V$ tel que $d_i \leq -f_i$, retirer toutes les arêtes incidentes à v_i dans G_3, poser $f_i = f_i + d_i$, et, pour tout sommet $v_j \in V$ adjacent à v_i dans G_2, poser $f_j = f_j - 1$ et $s_j = v_j$.

Soient $C_1 = (V_1, E_1)$, …, $C_c = (V_c, E_c)$ les c composantes connexes de G_3.

Poser $r = 1$.

Étape 2. Soit la composante connexe $C_r = (V_r, E_r)$.

Définir un ensemble W_r composé d'un sommet pour chaque arête de E_r.

Soit S la source et P le puits. Soit m_j, le nombre de copies de l'arc (v_i, P), pour tout $v_i \in E_r$.

Si $|V_r| = 1$, poser $n_i = f_i$ pour l'unique sommet v_i de V_r. Si $|V_r| > 1$, poser $m_i = (f_i + d_i) / 2$ pour tout sommet v_i de V_r.

Construire le graphe orienté $G_4 = (V'', A'')$ tel que $V'' = V_r \cup W_r \cup \{S, P\}$ et tel que $A'' = A_1 \cup A_2 \cup A_3$ où

$A_1 = \{(e_i, v_i), (e_i, v_j) : e_i = (v_i, v_j) \in E_r\}$,

$A_2 = \{(S, e_i) : e_i = (v_i, v_j) \in E_r\}$

et $A_3 = \{m_j$ copies de l'arc $(v_i, P) : v_i \in E_r\}$.

Attribuer à chaque arc de A'' une borne inférieure de flot nulle et une borne supérieure de flot de 1.

Résoudre le problème du flot maximal défini dans G_4.

Soit x_{kl} la valeur du flot traversant un arc (e_k, v_l) de A_1 de la composante connexe C_r. Si $r = 1$, poser $G_5 = G_2$.

Pour chaque arête $e_k = (v_i, v_j)$ de E_r, si $x_{kl} = 1$ et si $l = v_i$, poser $s_k = v_i$. Si $x_{kl} = 1$ et si $l = v_j$, poser $s_k = v_j$. Orienter l'arête $e_k = (v_i, v_j)$ de G_5 de v_i vers v_j si $s_k = v_i$ et de v_j vers v_i si $s_k = v_j$.

Pour une arête $e_k = (v_i, v_j)$ de E_r, si $s_k = 0$, orienter e_k de façon arbitraire, de v_i vers v_j par exemple, et poser $n_i = n_i - 2$.

Algorithme PPC_MIXTE_PAIR_ÉQUILIBRÉ (suite)

Étape 2. (suite)

Pour un sommet v_i de V_r, s'il y a p arêtes orientées vers v_i tel que $p < m_j$, poser $n_i = n_i + (m_j - p)$.

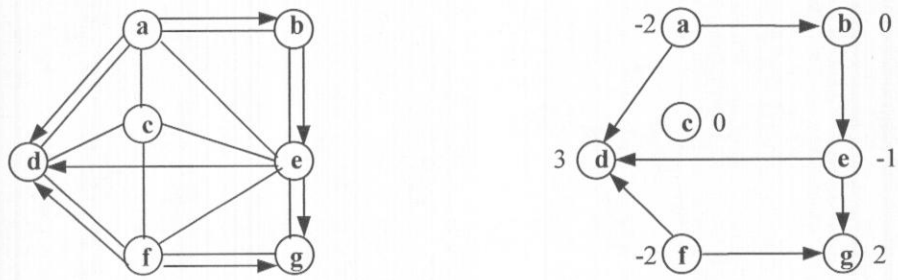
Si $r < c$, poser $r = r + 1$ et retourner à l'étape 2.

Étape 3. Si $n_i = 0$ pour tout $v_i \in V$, alors G est équilibré et G_5 est le graphe symétrique.

Si pour un v_i donné $n_i \neq 0$, STOP car G n'est pas équilibré.

Étape 4. Construire un circuit eulérien dans G_5 à l'aide de l'algorithme END-PAIRING modifié pour le cas orienté.

Le fonctionnement de l'algorithme est illustré à l'aide du graphe mixte pair de la figure 1.8 (a). Les nombres situés près des sommets du graphe G_1 de la figure 1.8 (b) correspondent aux disponibilités (s'ils sont positifs) ou aux demandes (s'ils sont négatifs).

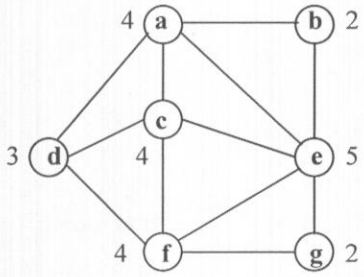


(a) Graphe initial G (b) Graphe partiel G_1

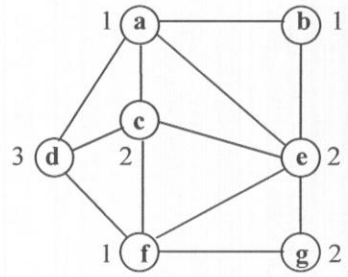
Figure 1.8: Illustration de l'algorithme pour le PPC pair et équilibré

Les nombres situés près des sommets du graphe G_2 de la figure 1.8 (c) de la page suivante correspondent aux degrés des sommets. Le graphe G_3 de la figure 1.8 (d) correspond au graphe G_2 car il n'y a qu'une seule composante connexe. Une composante connexe de G est un sous-graphe connexe contenu dans G. Les valeurs

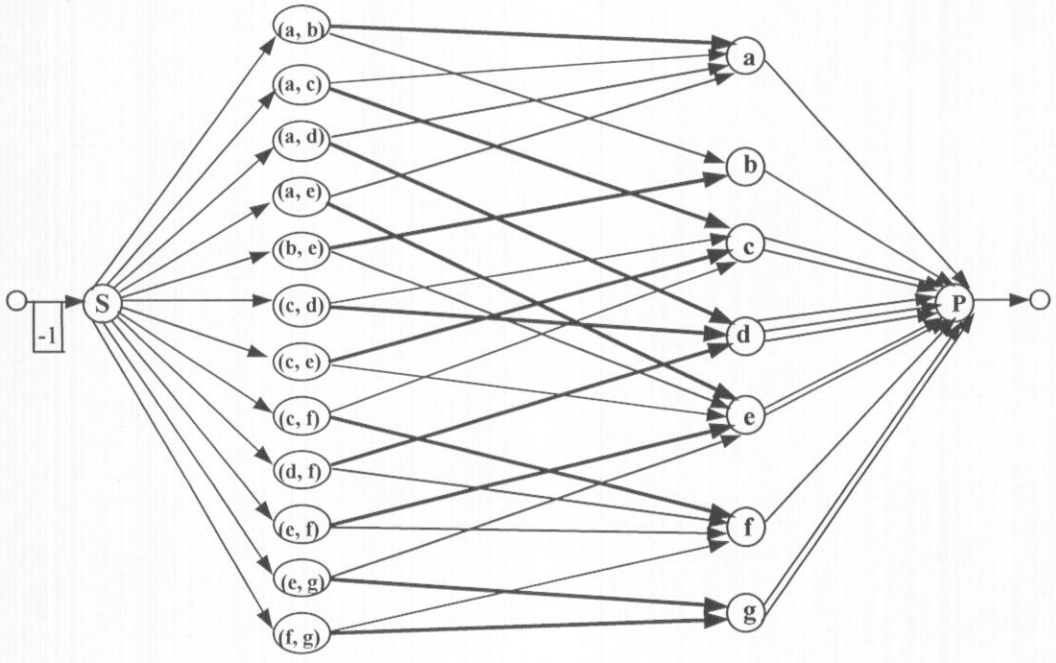
m_i sont indiquées près des sommets du graphe G_3. Le graphe G_4 dans lequel est défini un problème du flot maximal apparaît à la figure 1.8 (e). Le seul arc de coût non nul est l'arc $(\cdot, S)$ de coût égal à -1. Minimiser le coût total revient donc à maximiser la quantité de flot émise par le sommet S tout en tenant compte des capacités des arcs. Une solution optimale au problème du flot maximal dans G_4 correspond aux arcs dessinés en gras.



(c) Graphe partiel G_2



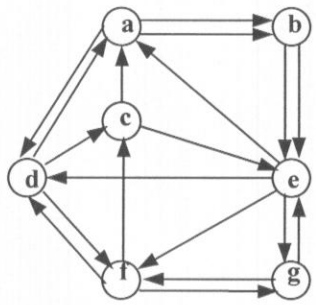
(d) Graphe G_3



(e) Graphe G_4

Figure 1.8 (suite): Illustration de l'algorithme pour le PPC pair et équilibré

L'algorithme END-PAIRING peut finalement être appliqué au graphe G_5 construit à partir de la solution du flot optimal (figure 1.8 (f)).



(f) Graphe G_5

Figure 1.8 (suite): Illustration de l'algorithme pour le PPC pair et équilibré

Une autre méthode pour orienter les arêtes de G de façon à obtenir un graphe orienté et symétrique G_5 consiste à résoudre d'abord un problème de flot à coût minimum dans un graphe auxiliaire G' et ensuite à orienter les arêtes de G' [For62].

Edmonds et Johnson [Edm73] ont proposé une méthode pour le cas où le graphe G est pair mais pas équilibré. Cette méthode est également présentée dans [Min78] et dans [Min79]. Elle consiste à rendre G symétrique tout en maintenant la parité de ces sommets. Les arêtes à orienter de même que les arcs et les arêtes orientées à copier sont déterminés en résolvant un problème de flot à coût minimum dans un graphe auxiliaire. La solution optimale du PPC mixte est alors obtenue en appliquant l'algorithme END-PAIRING modifié pour le cas orienté sur le graphe pair et symétrique construit à partir de la solution optimale au problème de flot à coût minimum.

La méthode, telle que présentée dans [Edm73], dans [Min78], dans [Min79] et dans [Bru81], ne maintient cependant pas nécessairement la condition de parité des sommets comme l'a soulevé Frederickson [Fre79]. Ce dernier propose d'appliquer une

procédure appelée EVENPARITY afin d'assurer que la condition soit satisfaite [Fre79]. Mittaz [Mit99] propose de définir le graphe auxiliaire dans lequel un flot compatible de coût minimum est recherché de manière légèrement différente que celle présentée dans [Min78] et dans [Min79] afin de maintenir la parité des sommets.

L'algorithme qui suit correspond à celui présenté dans [Mit99].

Algorithme PPC_MIXTE_PAIR

Étape 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 $f_i = d^-(i) - d^+(i)$.

Si tous les f_i sont nuls, aller à l'étape 3.

Étape 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 $f_i / 2$ sur chaque sommet v_i tel que $f_i > 0$ et une demande de $-f_i / 2$ sur tout sommet v_i tel que $f_i < 0$.

Trouver un flot de coût minimum dans G_2 satisfaisant les demandes.

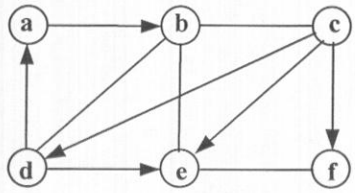
Soit x_{ij} la valeur du flot traversant un arc de $A \cup A_1 \cup A_2 \cup A_3$.

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.

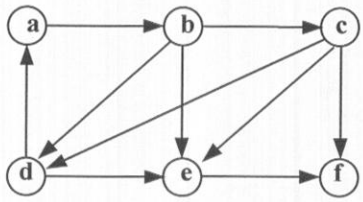
Augmenter G_3 en ajoutant $2 \cdot x_{ij}$ copies de chaque arc (v_i, v_j) appartenant à $A \cup A_1 \cup A_2$. G_3 est maintenant un graphe orienté pair et symétrique.

Étape 3. Construire un circuit eulérien dans G_3 en utilisant l'algorithme END-PAIRING adapté au cas orienté.

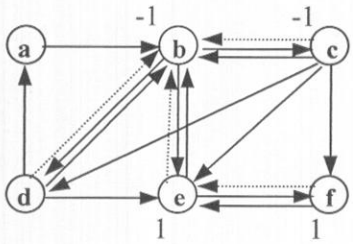
La figure 1.9 illustre le fonctionnement de cet algorithme. Le graphe pair et non équilibré est représenté à la figure 1.9 (a). À chaque arc et à chaque arête est associé un coût égal à 1. En orientant chacune des arêtes de E, on obtient le graphe G_1 de la figure 1.9 (b). La figure 1.9 (c) représente 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. Les nombres situés près des sommets correspondent aux demandes et aux disponibilités. Un flot compatible de coût minimum dans G_2 est illustré à la figure 1.9 (d).



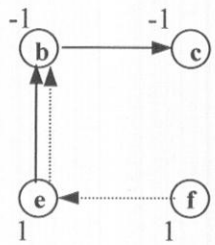
(a) Graphe initial G



(b) Graphe G_1



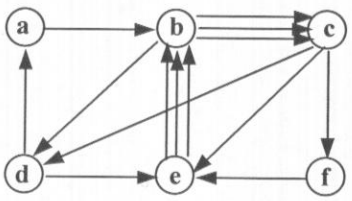
(c) Graphe auxiliaire G_2



(d) Un flot optimal dans G_2

Figure 1.9: Illustration de l'algorithme pour le PPC mixte pair

La figure 1.9 (e) correspond au graphe G_3 construit à partir de la solution optimale au problème de flot à coût minimum. Ce graphe est complètement orienté et symétrique. Il est donc eulérien et l'algorithme END-PAIRING peut lui être appliqué.



(e) Graphe G_3

Figure 1.9 (suite): Illustration de l'algorithme pour le PPC mixte pair

Pour le cas où le graphe mixte G n'est pas pair, Papadimitriou a montré que le PPC mixte est NP-complet [Pap76]. Pour traiter ce cas, Frederickson [Fre79] a développé deux heuristiques, appelées MIXED1 et MIXED2.

L'algorithme MIXED1 consiste à transformer le graphe G en un graphe pair G_1, puis à résoudre un problème de flot à coût minimum dans un graphe auxiliaire G'_2. Étant donné que le graphe G_3 construit à partir de la solution optimale au problème de flot à coût à minimum n'est plus forcément pair, une troisième étape, appelée EVENPARITY, permet finalement de rétablir la parité des sommets du graphe G_3. Un circuit eulérien dans G est finalement construit en appliquant l'algorithme END-PAIRING pour le cas orienté à G_3.

L'algorithme MIXED2 revient à transformer le graphe G en un graphe symétrique G_1, puis à transformer G_1 en un graphe pair et symétrique G_2 qui n'est pas nécessairement orienté. Les arêtes de G_2 sont donc orientées, à l'aide de l'algorithme de Papadimitriou par exemple, de façon à obtenir un graphe orienté et symétrique G_3.

En appliquant l'algorithme END-PAIRING à G_3, on obtient finalement un circuit eulérien dans G.

Frederickson [Fre79] a montré que les algorithmes MIXED1 et MIXED2 donnent des solutions dont la valeur est toujours inférieure ou égale à deux fois celle d'une solution optimale. Cependant, si les deux algorithmes sont appliqués en tandem à un problème et que la meilleure des deux solutions est retenue, le ratio entre le coût de la meilleure solution fournie par les deux algorithmes et le coût d'une solution optimale est alors réduit à 5/3 [Fre79]. Un résumé des résultats de Frederickson est présenté dans [Bru81].

Pearn et Liu [Pea95b] de même que Pearn et Chou [Pea99] proposent d'améliorer les algorithmes MIXED1 et MIXED2 développés par Frederickson. Les nouveaux algorithmes donnent de meilleurs résultats que MIXED1 et MIXED2.

Récemment, Raghavachari et Veerasamy [Rag99] ont modifié l'algorithme MIXED1 et ont montré que le ratio entre le coût de la solution obtenue en appliquant successivement les algorithmes MIXED1 modifié et MIXED2 et le coût d'une solution optimale au PPCM est réduit à 3/2. Les auteurs étudient les propriétés de solutions admissibles pour le PPCM et développent une nouvelle borne inférieure sur le coût d'une solution optimale au PPCM. L'algorithme MIXED1 modifié consiste, premièrement, à résoudre un problème de flot à coût minimum dans le graphe mixte G afin d'identifier certaines arêtes à orienter. Les coûts de ces arêtes orientées de même que ceux des arcs de G sont ensuite fixés à 0 forçant ainsi l'algorithme à copier un arc ou une arête orientée au lieu de copier une arête non orientée lors de la transformation de G en un graphe pair.

L'algorithme suivant correspond à celui présenté dans [Mit99] pour le PPC mixte. Il s'agit d'une variante de l'algorithme MIXED1 dans laquelle le graphe auxiliaire G_2 est défini de manière à maintenir la parité des sommets.

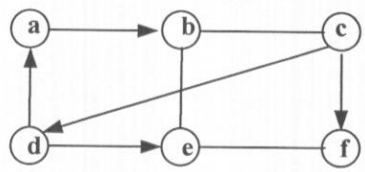
Algorithme PPC_MIXTE1

Étape 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, les arcs de G sont considérés comme des arêtes.
Déterminer un couplage parfait de coût minimum dans G_0.
Poser $G' = G$.

Étape 2. 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.

Étape 2. Appliquer l'algorithme PPC_MIXTE_PAIR à G'.

Le fonctionnement de l'algorithme PPC_MIXTE1 est illustré à l'aide de la figure 1.10. Dans le graphe initial G de la figure 1.10 (a), tous les coûts associés aux arcs et aux arêtes valent 1.

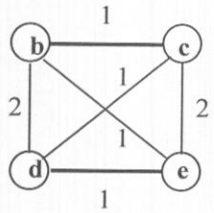


(a) Graphe initial G

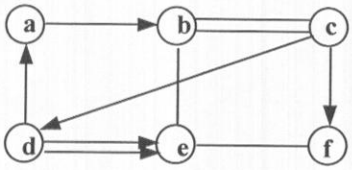
Figure 1.10: Illustration de l'algorithme PPC_MIXTE1

Le graphe G_0 (figure 1.10 (b)) correspond au graphe complet dans lequel un couplage parfait est recherché. Les nombres situés sur les arêtes correspondent aux longueurs des plus courtes chaînes. Les arêtes du couplage optimal sont dessinées en

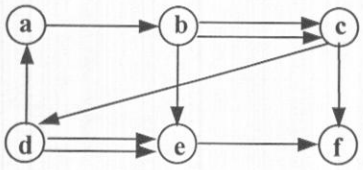
gras dans G_0. Le graphe G' représenté sur la figure 1.10 (c) est obtenu en dupliquant l'arc (d, e) et l'arête (b, c). En orientant les arêtes du graphe G', on obtient le graphe G_1 de la figure 1.10 (d). La figure 1.10 (e) représente le graphe G_2 dans lequel un flot compatible de coût minimum satisfaisant les demandes et les disponibilités est cherché. Les arcs appartenant à A_3 sont en pointillés. Les nombres situés près des sommets correspondent aux demandes et aux disponibilités.



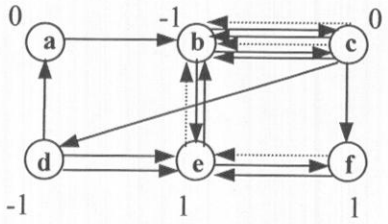
(b) Graphe G_0



(c) Graphe pair G'



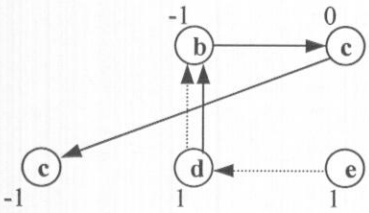
(d) Graphe G_1



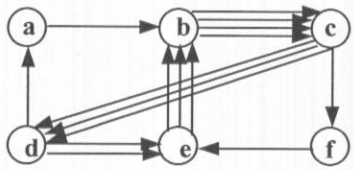
(e) Graphe G_2

Figure 1.10 (suite): Illustration de l'algorithme PPC_MIXTE1

Un flot compatible de coût minimum dans G_2 est illustré à la figure 1.10 (f) de la page suivante. La figure 1.10 (g) correspond au graphe G_3 construit à partir de la solution optimale au problème de flot à coût minimum. Ce graphe est complètement orienté et symétrique. Il est donc eulérien et l'algorithme END-PAIRING peut lui être appliqué.



(f) Un flot optimal dans G_2



(g) Graphe G_3

Figure 1.10 (suite): Illustration de l'algorithme PPC_MIXTE1

Plusieurs méthodes exactes ont également été proposées pour résoudre le PPC mixte [Min79; Kap79; Ral93; Chr84; Wan95; Yan98; Nob96]. Ces méthodes sont basées sur des formulations du PPC mixte en programmes linéaires en nombres entiers et utilisent les techniques de séparation et d'évaluation progressive, de coupes, ou encore de la relaxation lagrangienne.

1.2 Le problème du postier rural

Le problème du postier rural (PPR) a été introduit par Orloff [Orl74]. Étant donné un graphe fortement connexe $G = (V, E \cup A)$ et un sous-ensemble d'arcs et d'arêtes obligatoires R tel que $R \subseteq E \cup A$, le PPR consiste à déterminer un circuit de longueur minimum passant au moins une fois par chaque arc et chaque arête de R. Lenstra et Rinnooy Kan ont montré que le PPR est NP-difficile [Len76] sauf si G est non orienté ou orienté et que le sous-graphe partiel de G induit par R est connexe.

Dans cette section, les principales méthodes de résolution pour le PPR lorsque G est non orienté, orienté et mixte sont présentées.

1.2.1 Le PPR dans un graphe non orienté

Soit $G = (V, E)$ un graphe connexe non orienté, et soit $R \subseteq E$ l'ensemble des arêtes à desservir. Soit $G_R = (V_R, R)$ le sous-graphe de G engendré par R où V_R représente l'ensemble des sommets de V incidents aux arêtes de R.

En général, le PPR est résolu dans un graphe simplifié G_S composé uniquement des sommets appartenant à V_R. L'algorithme suivant permet de construire un tel graphe.

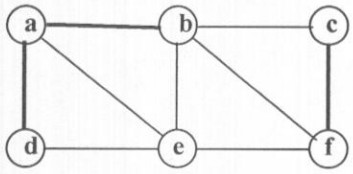
Algorithme PPR_NON_ORIENTÉ_SIMPLIFIÉ

Étape 1. Construire le graphe $G_S = (V_R, R \cup E_S)$ tel qu'à chaque paire de sommets $v_i, v_j \in V_R$ correspondent 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.

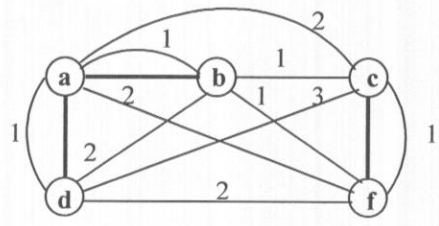
Étape 2. Enlever de E_S toutes les arêtes (v_i, v_j) pour lesquelles il existe un sommet $v_k \in V_R$ tel que: $c_{ij} = c_{ik} + c_{kj}$.

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

La figure 1.11 illustre la construction d'un graphe G_S. À chaque arête du graphe initial G de la figure 1.11 (a) correspond un coût égal à 1. Les arêtes en gras sont celles qui appartiennent à R. Les nombres situés près des arêtes représentent leur coût.

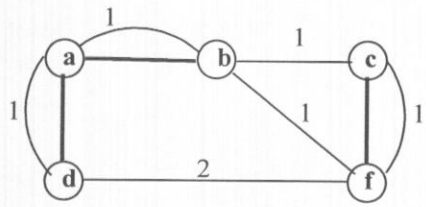


(a) Graphe initial G

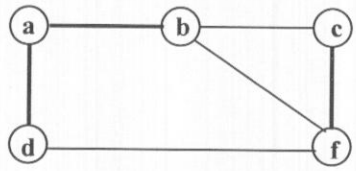


(b) Graphe G_S après l'étape 1

Figure 1.11: Exemple de simplification d'un graphe non orienté



(c) Graphe G_S après l'étape 2



(d) Graphe G_S après l'étape 3

Figure 1.11 (suite): Exemple de simplification d'un graphe non orienté

Il y a deux cas à considérer selon que G_R est connexe ou non. Lorsque G_R est connexe, deux cas peuvent également se présenter selon que G_R est pair ou non.

Si G_R est connexe et pair, un cycle eulérien C_S peut être déterminé dans G_S à l'aide de l'algorithme END-PAIRING (cf. section 1.1.1) car le PPR devient alors un PPC non orienté ($R = E$). À ce cycle C_S dans G_S correspond un cycle C pour le PPR dans G. Le cycle C est obtenu à partir de C_S en remplaçant dans C_S chaque arête non desservie par la plus courte chaîne qui lui est associée dans G. Par exemple, un cycle eulérien C_S dans le graphe G_S de la figure 1.11 (d) correspond à la séquence d'arêtes (a, b), (b, c), (c, f), (f, d) et (d, a). En remplaçant dans C_S les arêtes non desservies (b, c) et (f, d) respectivement par les plus courtes chaînes (b, c) et (f, e), (e, d) qui leur sont associées dans G, on obtient la suite d'arêtes (a, b), (b, c), (c, f), (f, e), (e, d) et (d, a) formant un cycle eulérien dans G.

Si G_R est connexe et non pair, le PPR non orienté revient alors à résoudre un problème de couplage parfait de coût minimum dans un graphe auxiliaire complet G_0 ayant pour sommets l'ensemble V_0 des sommets de degré impair dans G_R.

L'algorithme de la page suivante permet de résoudre le problème du postier rural dans un graphe non orienté.

Algorithme PPR_NON_ORIENTÉ_CONNEXE

Étape 1. Appliquer à G l'algorithme PPR_NON_ORIENTÉ_SIMPLIFIÉ de façon à obtenir un graphe simplifié G_S.

Étape 2. Poser $G' = G_R$.

Étape 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_S.
Trouver un couplage parfait de coût minimum dans G_0.
Pour chaque arête (v_i, v_j) du couplage optimal, ajouter à G' une copie de chaque arête appartenant à SC_{ij}.

Étape 4. Construire un cycle eulérien C' dans G' à l'aide de l'algorithme END-PAIRING.

Étape 5. Construire un cycle eulérien C dans G à partir de C' en remplaçant dans C' chaque arête non desservie par la plus courte chaîne qui lui est associée dans G.

Si G_R n'est pas connexe, le PPR dans un graphe non orienté est NP-difficile [Len76]. Frederickson [Fre79] a proposé de résoudre le PPR dans ce cas à l'aide d'une heuristique basée sur des principes similaires à ceux développés par Christofides [Chr76] pour le problème du voyageur de commerce.

Cette approche consiste à rendre le graphe G_R connexe puis pair. Le graphe G_R est rendu connexe en déterminant un arbre maximal de coût minimum dans un graphe complet G'' dont les sommets correspondent aux composantes connexes de G_R. Un arbre maximal de coût minimum dans un graphe $G'' = (V'', E'')$ est un graphe non orienté connexe sans cycle possédant le même ensemble de sommets que G'' et dont le coût total des arêtes formant l'arbre soit minimum. L'algorithme PPR_NON_ORIENTÉ_CONNEXE permet ensuite de transformer le graphe G_R en un graphe pair.

L'algorithme qui suit utilise de tels principes.

Algorithme PPR_NON_ORIENTÉ

Étape 1. Appliquer à G l'algorithme PPR_NON_ORIENTÉ_SIMPLIFIÉ de façon à transformer le graphe G en un graphe simplifié G_S.

Étape 2. Soient $C_1, C_2, \ldots, C_c$ les différentes composantes connexes de G_R.
Poser $G' = G_R$. Si $c = 1$, aller à l'étape 4.
Construire un graphe complet $G'' = (V'', E'')$ tel qu'à chaque composante connexe C_p de G' corresponde dans V'' un sommet v_p'', $1 \leq p \leq c$, et tel qu'à chaque arête (v_p'', v_q'') de E'' est associé un coût égal à longueur de la plus courte chaîne SC_{pq} reliant un sommet de C_p à un sommet de C_p dans G_S.

Étape 3. 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_S appartenant à SC_{pq}.

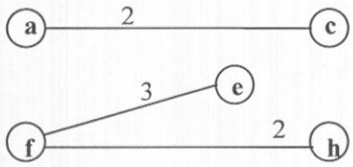
Étape 4. Appliquer les étapes 3, 4 et 5 de PPR_NON_ORIENTÉ_CONNEXE à G' de façon à obtenir un cycle eulérien dans G.

La figure 1.12 illustre le fonctionnement de l'algorithme PPR_NON_ORIENTÉ. Le graphe initial G dont les coûts associés aux arêtes sont tous de 1 apparaît à la figure 1.12 (a). Les arêtes obligatoires sont dessinées en gras. La figure 1.12 (b) représente le graphe simplifié G_S. Les nombres situés au-dessus des arêtes de G_S correspondent aux longueurs des plus courtes chaînes qui leur sont associées dans G.

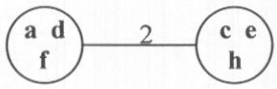
(a) Graphe initial G (b) Graphe simplifié G_S

Figure 1.12: Illustration de l'algorithme PPR_NON_ORIENTÉ

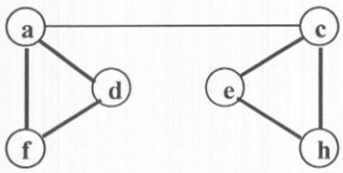
La figure 1.12 (c) représente les arêtes qui lient entre elles les différentes composantes connexes de G_R. Les coûts définis dans G_S apparaissent près des arêtes. Le graphe G'' est représenté à la figure 1.12 (d). Ce graphe est composé de deux sommets uniquement. Un arbre maximal de coût minimum dans G'' correspond à l'arête (a, c) de G_S. C'est cet arc qui est ajouté à G' (voir figure 1.12 (e)). La figure 1.12 (f) représente l'union des arêtes de G' et de celles du couplage.



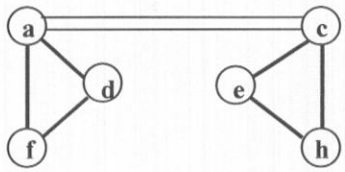
(c) Arêtes connectant les composantes connexes de G_R



(d) Graphe complet G''



(e) Graphe connexe G'



(f) Graphe connexe et pair G'

Figure 1.12 (suite): Illustration de l'algorithme PPR_NON_ORIENTÉ

Hertz et al. [Her99] ont proposé des heuristiques de post-optimisation pour le PPR non orienté. Deux de ces heuristiques (2-OPT et DROP_ADD) utilisent, entre autres, le principe de la recherche locale.

Étant donné le problème visant à minimiser la fonction-objectif $f(s)$ sous la contrainte que $s \in S$ où S est l'ensemble des solutions admissibles, une méthode de recherche locale est une méthode itérative qui consiste, à partir d'une solution initiale $s_0 \in S$, à se déplacer d'une solution courante s à une solution $s' \in N(s)$ voisine de s et

choisie selon un certain critère. À chaque solution $s \in S$ correspond un voisinage $N(s) \subseteq S$ composé des solutions admissibles voisines de s qui peuvent être obtenues en appliquant un certain type de modification à s. La recherche se termine si aucune solution du voisinage ne peut satisfaire le critère de sélection. On peut également arrêter la recherche lorsque le nombre maximum d'itérations permises est atteint, lorsque le nombre maximum d'itérations successives sans amélioration de la meilleure solution s_{best} trouvée jusqu'à présent est atteint ou encore dès qu'un temps limite de calcul est dépassé.

Description d'une méthode de recherche locale

Étape 1. (*Initialisation et définition du voisinage*)

Définir la structure de voisinage N utilisée lors la recherche.

Trouver une solution initiale $s_0 \in S$. Poser $s = s_0$.

Étape 2. (*Mise à jour*)

Poser $s_{best} = s$ et $f_{best} = f(s_{best})$.

Étape 3. (*Exploration du voisinage*)

Trouver une solution $s' \in N(s)$.

Si pour tout $s' \in N(s)$ le critère de sélection n'est pas satisfait ou encore si le critère d'arrêt est atteint, STOP.

Étape 4. (*Déplacement ou non*)

Poser $s = s'$. Si $f(s) < f_{best}$, retourner à l'étape 2. Sinon, retourner à l'étape 3.

Un critère de sélection souvent utilisé dans le développement d'une méthode de recherche locale consiste à choisir à chaque itération un voisin $s' \in N(s)$ tel que $f(s') < f(s)$. Ce processus, appelé méthode de descente, est répété jusqu'à ce qu'il ne soit plus possible d'améliorer la solution courante s. La solution s' voisine de s peut être n'importe quelle solution de $N(s)$ telle que $f(s') < f(s)$. En général, la solution s' voisine de s correspond à la meilleure solution trouvée dans le voisinage $N(s)$ de s.

Description d'une méthode de descente

Étape 1. (*Initialisation et définition du voisinage*)

Définir la structure de voisinage N utilisée lors la descente.

Trouver une solution initiale $s_0 \in S$. Poser $s = s_0$.

Étape 2. (*Mise à jour*)

Poser $s_{best} = s$ et $f_{best} = f(s_{best})$.

Étape 3. (*Exploration du voisinage*)

Déterminer $s' \in N(s)$ telle que $f(s') = \min f(s'')$ sous la contrainte que $s'' \in N(s)$.

Étape 4. (*Déplacement ou non*)

Si $f(s') < f_{best}$, poser $s = s'$ et retourner à l'étape 2. Sinon, STOP.

Soit T une tournée et soit $R_T \subseteq R$ l'ensemble des arêtes obligatoires qui sont desservies dans T. La première procédure dont il est question dans [Her99] consiste à raccourcir la longueur d'une tournée T en remplaçant, si possible, une chaîne P d'arêtes non requises de T par la plus courte chaîne reliant ses extrémités.

Algorithme SHORTEN

Étape 1. Choisir une orientation de T et un sommet v_i de T. Le sommet v_i est le premier sommet de la tournée T. Une arête de R_T doit être desservie lors de sa dernière apparition dans T.

Étape 2. Déterminer le sommet v_j de T origine de la première arête desservie dans T rencontrée en partant de v_i. Soit P la chaîne reliant v_i à v_j dans T.

Algorithme SHORTEN (suite)

Étape 3. Soit Q la chaîne reliant v_j à v_i dans T.
S'il existe une arête non desservie (v_k, v_j) appartenant à Q, alors déterminer le cycle $C = (v_j, \ldots, v_k, v_j)$, renverser son orientation dans T et aller à l'étape 2.
Si toutes les arêtes de Q ayant pour sommet terminal le sommet v_j sont des arêtes desservies ou si aucune arête (v_k, v_j) n'apparaît dans Q, aller à l'étape 4.

Étape 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}.

Étape 5. Répéter les étapes 1 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 être obtenue.

L'exemple suivant illustre le fonctionnement de l'algorithme SHORTEN. La figure 1.13 représente le graphe initial G. Les coûts associés aux arêtes de G sont de 1 et les arêtes de R sont dessinées en gras.

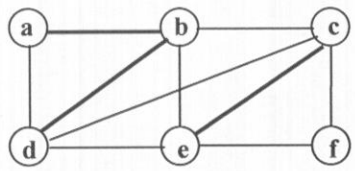
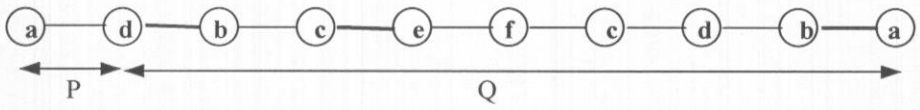


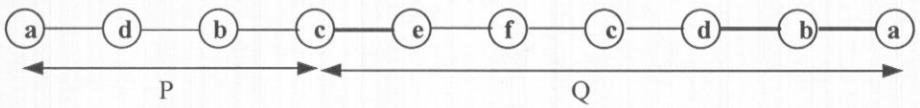
Figure 1.13: Graphe initial G

Une tournée $T = (a, d, b, c, e, f, c, d, b, a)$ desservant les arêtes de R est représentée à la figure 1.14 (a). Le sommet de départ v_i est le sommet a. Le service de l'arête (d, b) est déplacé en fin de tournée sur l'arête (d, b) qui devient une arête desservie (figure 1.14 (b)). Cela permet de rallonger la chaîne inutile partant de v_i. Ainsi, $P = (a, d, b, c)$ et $Q = (c, e, f, c, d, b, a)$. L'arête (f, c) de Q est une arête non desservie dont le sommet terminal est $c = v_j$. L'orientation du cycle $C = (c, e, f, c)$ est

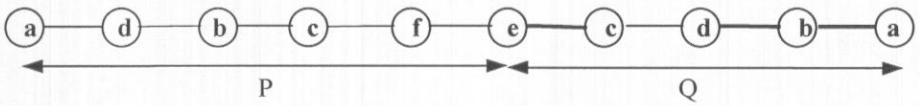
alors renversée (figure 1.14 (c)) ce qui permet d'étendre P jusqu'au sommet e. De cette façon, $P = (a, b, d, c, f, e)$ et $Q = (e, c, d, b, a)$. La chaîne P est ensuite remplacée par la plus courte chaîne (a, b, e) reliant les sommets a et e pour obtenir la tournée de la figure 1.14 (d).



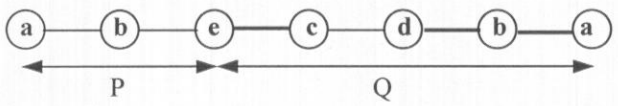
(a) Une tournée T dans G



(b) Déplacement du service de l'arc (d, b)



(c) Inversion du cycle $C = (c, e, f, c)$



(d) Remplacement de la chaîne $P = (a, d, b, c, f, e)$ par la chaîne (a, b, e)

Figure 1.14: Illustration de l'algorithme SHORTEN

Une deuxième procédure proposée par Hertz et al. [Her99] consiste à enlever de T et à rajouter dans T successivement toutes les arêtes appartenant à R_T tout en raccourcissant T après chacune de ces opérations. Cette méthode, composée des sous-routines DROP et ADD, est équivalente à l'algorithme UNSTRIGING_STRINGING utilisé pour le PVC dans [Gen92].

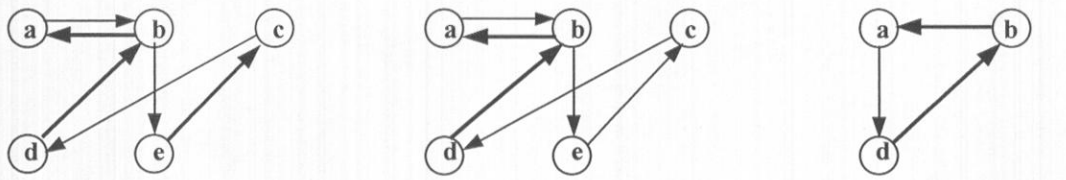
La procédure DROP permet de supprimer le service d'une arête (v_i, v_j) de T. La nouvelle tournée T' ainsi obtenue dessert alors les arêtes de $R_T \setminus (v_i, v_j)$.

Algorithme DROP

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

Étape 2. Appliquer SHORTEN à T pour obtenir une tournée T'.

La figure 1.15 illustre le fonctionnement de l'algorithme DROP. Considérons le graphe initial G de la figure 1.13. La figure 1.15 (a) correspond à une tournée T desservant toutes le arêtes de R. En supprimant de R l'arête (c, e), la chaîne $P = (a, b, e, c, d)$ devient une chaîne inutile (figure 1.15 (b)). La procédure SHORTEN remplace cette chaîne par l'arête (a, d) reliant a à d dans G pour former une nouvelle tournée T' desservant les arêtes de $R \setminus (c, e)$ (figure 1.15 (c)).



(a) Une tournée T dans G (b) L'arête (c, e) est retirée de R (c) Tournée finale T'

Figure 1.15: Illustration de l'algorithme DROP

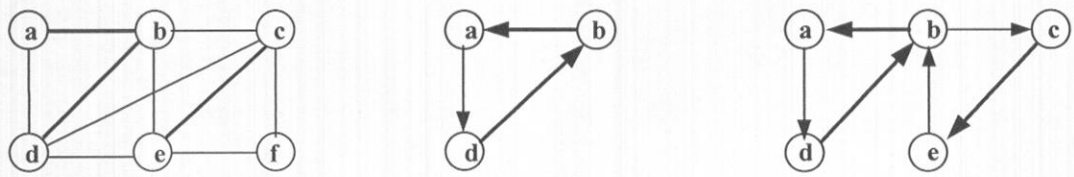
La procédure ADD permet d'insérer dans T une arête (v_i, v_j) de R n'appartenant pas à R_T afin qu'elle puisse être desservie.

Algorithme ADD

Étape 1. Soit (v_i, v_j) une arête de R n'appartenant pas à R_T et soit sc_{ij} la longueur de la plus courte chaîne SC_{ij} entre v_i et v_j dans G.
Si les sommets v_i et v_j ne sont pas 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).

Étape 2. Poser $R_T = R_T \cup \{(v_i, v_j)\}$ et appliquer SHORTEN à T pour obtenir une tournée T'.

L'exemple de la figure 1.16 illustre un cas possible de la procédure ADD où les sommets v_i et v_j n'appartiennent pas à T.



(a) Graphe initial G (b) Un cycle T couvrant $R_T = \{(a, b), (b, d)\}$ (c) (c, e) est ajoutée à R_T

Figure 1.16: Illustration de l'algorithme ADD

La méthode DROP_ADD décrite dans [Her99] peut maintenant être présentée.

Algorithme DROP_ADD

Étape 1. Poser $T_{best} = T$ et poser $f_{best} = f(T)$. Soit E' l'ensemble des arêtes qui ont été choisies à l'étape 2. Poser $E' = \varnothing$.

Étape 2. Choisir une arête (v_i, v_j) desservie dans T n'appartenant pas à E' et construire un cycle T' desservant toutes les arêtes de $R_T \setminus \{(v_i, v_j)\}$ en utilisant DROP. Poser $E' = E' \cup \{(v_i, v_j)\}$.

Étape 3. Poser $T = T'$.
Construire un cycle T' desservant les arêtes de R_T en appliquant ADD à T.

Étape 4. Poser $T = T'$. Si $f(T) < f_{best}$, poser $T_{best} = T$, poser $f_{best} = f(T)$, poser $E' = \varnothing$, et retourner à l'étape 2.

Étape 5. Si toutes les arêtes de R_T ont été choisies, STOP. Sinon, retourner à l'étape 2.

Hertz et al. proposent une autre méthode de post-optimisation similaire à la méthode 2-opt développée pour le PVC dans [Cro58].

Algorithme 2-OPT

Étape 1. Poser $T_{best} = T$ et poser $f_{best} = f(T)$. Soit E_1 l'ensemble des paires d'arêtes qui ont été choisies à l'étape 2. Poser $E_1 = \varnothing$.

Étape 2. Soit E_2 l'ensemble des arêtes desservies dans T.
Choisir une orientation de T et sélectionner dans T une paire d'arêtes (e_s, e_t) où $e_s = (v_i, v_j)$ et $e_t = (v_k, v_h)$ telle que $(e_s, e_t) \notin E_1$. Remplacer les arêtes (v_i, v_j) et (v_k, v_h) respectivement par 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' la nouvelle tournée obtenue. Poser $T = T'$ et appliquer SHORTEN à T pour obtenir une tournée T'. Poser $E_1 = E_1 \cup \{(e_s, e_t) \mid e_s = (v_i, v_j) \text{ et } e_t = (v_k, v_h)\}$.

Algorithme 2-OPT (suite)

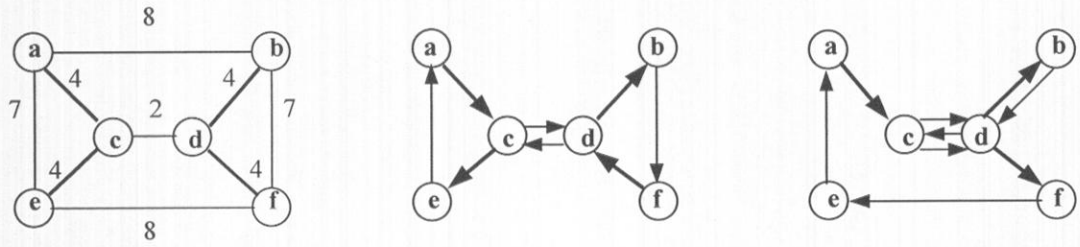
Étape 3. Si (v_i, v_j) et/ou (v_k, v_h) sont des arêtes de E_2 non couvertes par T', poser $T = T'$ et construire une tournée T' desservant les arêtes de E_2 en utilisant ADD.

Étape 4. Poser $T = T'$. Si $f(T) < f_{best}$, poser $T_{best} = T$, poser $f_{best} = f(T)$, poser $E_1 = \varnothing$, et retourner à l'étape 2. Sinon, poser $T = T_{best}$.

Étape 5. Si toutes les paires d'arêtes (v_i, v_j) et (v_k, v_h) de T ont été choisies, chacune avec les deux orientations possibles pour T, STOP. Sinon, retourner à l'étape 2.

L'heuristique 2-opt pour le PVC retire, à chaque itération, deux arêtes non adjacentes qui sont remplacées par deux arêtes différentes, modifiant ainsi seulement quatre attributs de la solution. L'application de la méthode 2-OPT à une tournée T peut cependant impliquer plusieurs étapes intermédiaires avant d'atteindre une tournée de coût inférieur à celui de T. Il est donc difficile de prévoir le comportement de l'algorithme.

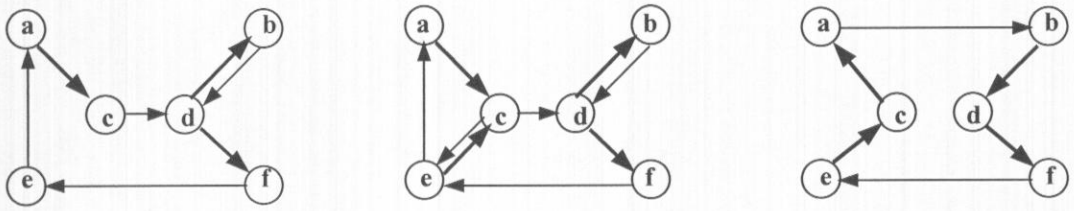
L'exemple de la figure 1.17, tiré de [Her99], illustre le fonctionnement de l'algorithme 2-OPT. Le graphe de départ et le cycle T qui a été orienté sont représentés dans les figures 1.17 (a) et 1.17 (b). Les arêtes (c, e) et (b, f) sont retirées de T, elles sont remplacées respectivement par les chaînes (e, f) et (c, d, b) (figure 1.17 (c)).



a) Graphe initial G b) Une tournée T dans G c) Remplacement des arêtes (c, e) et (b, f)

Figure 1.17: Illustration de l'algorithme 2-OPT

À l'étape 2 de la procédure SHORTEN, le cycle (*d*, *c*, *d*) de *T* est supprimé (figure 1.15 (d)). L'arête à desservir (*c*, *e*) ne se trouvant plus dans *T*, il faut la rajouter en utilisant ADD. La tournée *T* ainsi obtenue apparaît à la figure 1.15 (e). La procédure ADD faisant appel à la procédure SHORTEN à l'étape 3, les chaînes (*c*, *e*, *a*) et (*a*, *c*, *d*, *b*) de *T* sont respectivement remplacées par les chaînes (*c*, *a*) et (*a*, *b*) qui sont plus courtes. La tournée *T* une fois raccourcie apparaît à la figure 1.15 (f).



(d) Tournée *T* après l'étape 2 (e) (*c*, *e*) est rajoutée à *T* (f) Tournée finale *T*

Figure 1.17 (suite): Illustration de l'algorithme 2-OPT

D'autres heuristiques ont également été proposées pour le PPR dans un graphe non orienté, notamment dans [Chr81], [Pea95a] et [deC98].

Quelques méthodes exactes ont été développées pour le PPR dans un graphe non orienté entre autres par Christofides, Campos, Corberán et Mota [Chr81] ainsi que par Ghiani et Laporte [Ghi00] et Corberán et Sanchis [Cor94].

Garfinkel et Webb [Gar99] ont proposé une formulation du PPR dans un graphe non orienté basée sur l'étude de la structure des solutions optimales.

1.2.2 Le PPR dans un graphe orienté

Soit $G = (V, A)$ un graphe orienté, fortement connexe et soit $R \subseteq A$ l'ensemble des arcs à desservir. Comme pour le cas non orienté, le PPR est résolu dans un graphe simplifié G_S composé uniquement des sommets appartenant à V_R.

L'algorithme suivant permet de construire un graphe simplifié pour le PPR orienté.

Algorithme PPR_ORIENTÉ_SIMPLIFIÉ

Étape 1. Construire le graphe $G_S = (V_R, R \cup A_S)$ tel qu'à chaque paire de sommets $v_i, v_j \in V_R$ correspondent dans A_S deux arcs de direction opposée (v_i, v_j) et (v_j, v_i). 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.

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

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

Comme dans le cas non orienté, il y a deux cas à traiter selon que G_R est connexe ou non. De même, lorsque G_R est connexe, deux cas peuvent être considérés selon que G_R est symétrique ou non.

Si G_R est connexe et symétrique, un circuit eulérien C_S peut être déterminé dans G_S à l'aide de l'algorithme END-PAIRING adapté pour le cas orienté (cf. section 1.1.2). À ce circuit C_S dans G_S correspond un circuit C pour le PPR dans G. Le circuit C est obtenu à partir de C_S en remplaçant dans C_S chaque arc non desservi par le plus court chemin qui lui est associé dans G.

Si G_R est connexe et non symétrique, le PPR devient un PPC orienté. Dans ce cas, le PPR peut se résoudre à l'aide d'un algorithme polynomial basé sur l'une des deux approches proposées pour le PPC orienté (cf. section 1.1.2).

L'algorithme PPR_ORIENTÉ_CONNEXE utilise la résolution d'un problème de flot à coût minimum dans un graphe auxiliaire.

Algorithme PPR_ORIENTÉ_CONNEXE

Étape 1. Appliquer à G l'algorithme PPR_ORIENTÉ_SIMPLIFIÉ de façon à obtenir un graphe simplifié G_S.

Étape 2. Poser $G' = G_R$.

Étape 3. Calculer la valeur $f_i = d^-(i) - d^+(i)$ pour chaque sommet $v_i \in G'$.

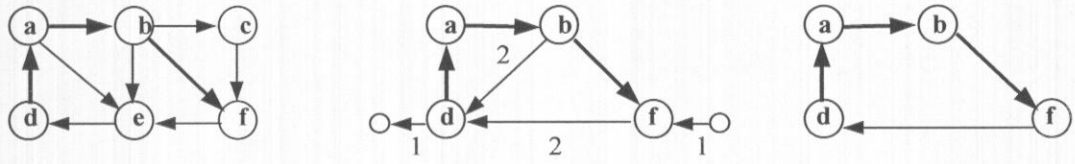
Étape 4. Soit A' l'ensemble des arcs de G'. Déterminer l'ensemble des sommets émetteurs $I = \{v_i \in V_R : f_i > 0\}$ et l'ensemble des sommets récepteurs $J = \{v_j \in V_R : f_j < 0\}$. Construire le réseau $G_0 = (V_R, A' \cup A_0)$ où A_0 est l'ensemble des arcs virtuels $(\cdot, v_i)$ associés aux sommets émetteurs v_i et des arcs virtuels $(v_j, \cdot)$ associés aux sommets récepteurs v_j. Attribuer à chaque arc virtuel $(\cdot, v_i)$ une disponibilité de valeur f_i et à chaque arc virtuel $(v_j, \cdot)$ une demande de valeur $|f_i|$. Attribuer à chaque arc de A' une capacité infinie.
Trouver un flot compatible de coût minimum dans G_0 satisfaisant les demandes.

Étape 5. Soit x_{ij} la valeur du flot traversant un arc de $A' \cup A_0$. Ajouter à G' x_{ij} copies de chaque arc (v_i, v_j) appartenant à A'.

Étape 6. Construire un circuit eulérien C' dans G' à l'aide de l'algorithme END-PAIRING adapté pour le cas orienté.

Étape 7. Construire un circuit eulérien C dans G à partir de C' en remplaçant dans C' chaque arête non desservie par le plus court chemin qui lui est associé dans G.

Le fonctionnement de l'algorithme PPR_ORIENTÉ_CONNEXE est illustré à l'aide de la figure 1.18.



(a) Graphe initial G (b) Graphe simplifié G_S (c) Graphe symétrique G'

Figure 1.18: Illustration de l'algorithme PPR_ORIENTÉ_CONNEXE

Le coût associé à chacun des arcs du graphe G de la figure 1.18 (a) est égal à 1. Les arcs à desservir sont dessinés en gras. Les nombres situés près des arcs virtuels du graphe G_S indiquent les disponibilités et les demandes. Les autres nombres correspondent aux coûts définis à l'étape 1. Le graphe symétrique obtenu à la fin de l'étape 5 est représenté à la figure 1.18 (c).

Si G_R n'est pas connexe, le PPR est NP-difficile [Len76]. Ball et Magazine [Bal88] proposent de résoudre le PPR orienté en transformant d'abord le graphe G_R en un graphe symétrique G' puis en rendant G' fortement connexe.

L'algorithme BALANCE_CONNECT est celui présenté dans [Bal88].

Algorithme BALANCE_CONNECT

Étape 1. Appliquer à G l'algorithme PPR_ORIENTÉ_SIMPLIFIÉ de façon à obtenir un graphe simplifié G_S.

Étape 2. Poser $G' = G_R$.

Étape 3. Appliquer les étapes 3, 4 et 5 de PPR_ORIENTÉ_CONNEXE de façon à transformer G' en un graphe symétrique.

Algorithme BALANCE_CONNECT (suite)

Étape 4. Si G' est connexe aller à l'étape 5.

Soient $C_1, C_2, \ldots, C_c$ les différentes composantes connexes de G' et soit PCC_{ij} la longueur d'un plus court chemin reliant v_i à v_j dans G_S. Soit pcc_{ij} la longueur du plus court chemin PCC_{ij} entre v_i et v_j dans G_S.

Construire un graphe complet $G'' = (V'', E'')$ tel qu'à chaque composante connexe C_p de G' corresponde dans V'' un sommet v_p'', $1 \leq p \leq c$, et tel qu'à chaque arête (v_p'', v_q'') de E'' est associé le coût c_{pq}'' tel que

$$c_{pq}'' = min\{pcc_{ij} + pcc_{ji} : v_i \in C_p \text{ et } v_j \in C_q\}.$$

Déterminer un arbre maximal de coût minimum dans G''.

Pour chaque arête (v_p'', v_q'') appartenant à l'arbre optimal, considérer les sommets v_i de C_p et v_j de C_q tels que $c_{pq}'' = pcc_{ij} + pcc_{ji}$ et ajouter à G' tous les arcs de G_S appartenant à PCC_{ij} ainsi que tous les arcs de G_S appartenant à PCC_{ji}.

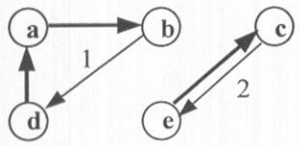
Étape 5. Construire un circuit eulérien C' dans G' à l'aide de l'algorithme END-PAIRING adapté pour le cas orienté.

Étape 6. Construire un circuit eulérien C dans G à partir de C' en remplaçant dans C' chaque arc non desservi par le plus court chemin qui lui est associé dans G.

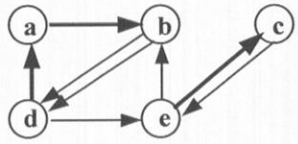
Le fonctionnement de l'algorithme BALANCE_CONNECT est illustré à l'aide de la figure 1.19.

(a) Graphe initial G (b) Graphe simplifié G_S

Figure 1.19: Illustration de l'algorithme BALANCE_CONNECT



(c) Graphe symétrique à la fin de l'étape 3



(d) Graphe symétrique et fortement connexe à la fin de l'étape 5

Figure 1.19 (suite): Illustration de l'algorithme BALANCE_CONNECT

Une seconde heuristique pour le PPR orienté consiste à inverser les deux phases de l'algorithme BALANCE_CONNECT. Les composantes connexes du graphe G_R sont d'abord connectées entre elles pour former un graphe fortement connexe qui est ensuite rendu symétrique. Cette approche a été étudiée par Ball et Magazine [Bal88], Christofides et al. [Chr86] et Su [Su92].

Mittaz [Mit99] a adapté au cas orienté les méthodes de post-optimisation de Hertz et al. [Her99] pour le PPR non orienté.

Deux adaptations de SHORTEN au cas orienté ont été étudiées par Mittaz. Dans les deux adaptations, la seule orientation à considérer est celle du circuit à raccourcir. La première adaptation, appelée RACCOR_SIMPLE, consiste simplement à supprimer l'étape 3 de la procédure SHORTEN décrite à la section 1.2.1. Dans la deuxième adaptation, nommée RACCOR, l'étape 3 est remplacée par une nouvelle étape ALLONGER qui consiste à augmenter la taille du chemin inutile P à raccourcir.

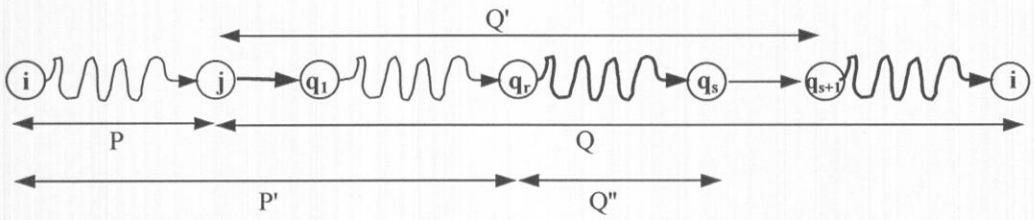
Algorithme ALLONGER

Étape 1. Le service d'un arc de R_T s'effectue lors de sa dernière apparition dans T.
Soit un chemin inutile $P = (i, \ldots, j)$. Le sommet i est le premier sommet de T.
Soit $Q = T \setminus P = (j, q_1, \ldots, q_t, i)$.
Par définition du chemin P, l'arc (j, q_1) est desservi dans T.
Chercher dans Q un arc non desservi (q_s, q_{s+1}) tel que $q_s = j$.
Si un tel arc n'existe pas dans Q alors poser $P' = P$ et $T' = T$, STOP.
Sinon poser $Q' = (j, q_1, \ldots, q_s, q_{s+1})$.

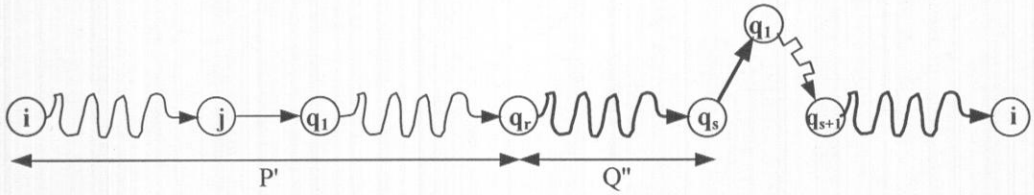
Étape 2. Soit $PCC(i, j)$ le plus court chemin reliant le sommet i au sommet j.
Si (j, q_1) est le seul arc desservi de Q' alors poser $q_r = q_s$. Sinon q_r est l'origine du deuxième arc desservi de Q'.
Poser $P' = P + (j, q_1, \ldots, q_r)$ et $Q' = (q_r, q_{r+1}, \ldots, q_s)$.
Poser $T' = P' + Q'' + (q_s, q_1) + PCC(q_1, q_{s+1}) + (q_{s+1}, q_{s+2}, \ldots, q_t, i)$.

Étape 3. Soit $pcc(i, j)$ la longueur du plus court chemin reliant le sommet i au sommet j.
Soient $L(Q')$ et $L(Q'')$ les longueurs de Q' et Q'' respectivement.
Poser $L_1 = pcc(i, j) + L(Q')$ et $L_2 = pcc(i, q_r) + L(Q'') + c(q_s, q_1) + pcc(q_1, q_{s+1})$.
Si $L_1 \leq L_2$, poser $P' = P$ et $T' = T$.

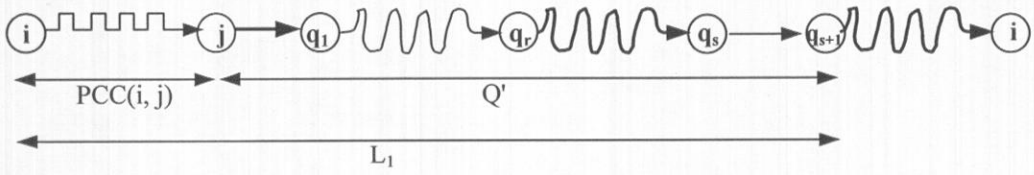
Le principe de la procédure ALLONGER consiste à essayer de déplacer le service de l'arc (j, q_1) de manière à pouvoir augmenter la taille du chemin inutile P. Par définition de P, l'arc (j, q_1) n'apparaît qu'une fois dans Q. Pour déplacer le service de (j, q_1), Mittaz suggère d'introduire une copie supplémentaire de cet arc dans Q. Pour ce faire, un arc non desservi $(q_s, q_{s+1}) = (j, q_{s+1})$ est recherché dans Q. Si un tel arc existe, une tournée T' est construite en remplaçant (q_s, q_{s+1}) par le chemin $(q_s, q_1) + PCC(q_1, q_{s+1})$ (figure 1.20 (b)). Dans T', un chemin inutile P' est obtenu en prolongeant le chemin P jusqu'à l'origine du prochain arc desservi de T'. Mittaz propose de déterminer s'il est plus avantageux de raccourcir P' (i.e. de remplacer P' par un plus court chemin reliant ses extrémités) dans T' ou de raccourcir P dans T (voir figures 1.20 (c) et 1.20 (d)). La meilleure de ces deux variantes est finalement retenue.



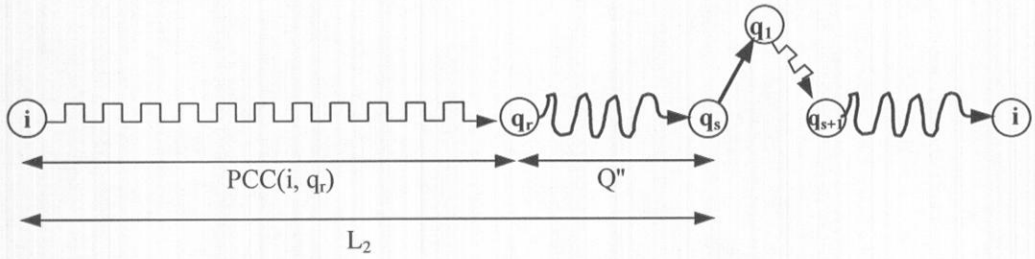
(a) Tournée T



(b) Tournée T' : le service de l'arc (j, q_1) est déplacé sur l'arc (q_s, q_1)

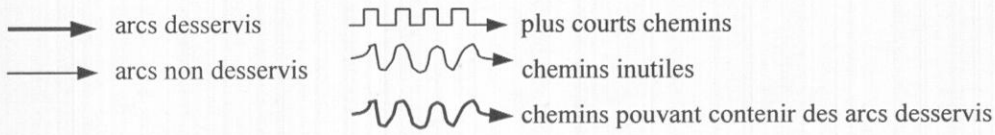


(c) Longueur du tronçon L_1



(d) Longueur du tronçon L_2

Figure 1.20: Illustration de la procédure ALLONGER



Algorithme RACCOR

Étape 1. Choisir un sommet v_i de T. Le sommet v_i est le premier sommet de la tournée T. Un arc de R_T doit être desservi lors de sa dernière apparition dans T.

Étape 2. En partant de v_i, déterminer le sommet v_j de T qui est le sommet initial du premier arc desservi dans T. Soit P le chemin reliant v_i à v_j dans T.

Étape 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 $P = P'$.

Étape 4. Si la longueur du plus court chemin PCC_{ij} reliant v_i à v_j dans G est inférieure à celle de P, remplacer P par PCC_{ij}.

Étape 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 être obtenue.

Les résultats figurant dans [Mit99] montrent que la procédure RACCOR décrite ci-dessus est plus efficace que RACCOR_SIMPLE. Les modifications apportées par Mittaz à la procédure ADD pour le cas orienté interviennent lorsqu'un arc à desservir (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 $(v_i, v_j) + PCC_{ji}$ est alors ajouté à T.

Algorithme ADD_ORIENTÉ

Étape 1. Si le sommet v_i n'est pas présent dans T, trouver le sommet $v_k \in T$ minimisant la valeur $pcc_{ki} + pcc_{jk}$ et ajouter à T le circuit $PCC_{ki} + \{(v_i, v_j)\} + PCC_{jk}$.
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 $(v_i, v_j) + PCC_{ji}$.

Étape 2. Poser $R_T = R_T \cup (v_i, v_j)$ et appliquer RACCOR à T pour obtenir une tournée au moins aussi courte que T.

Tel que mentionnée dans [Mit99], la procédure DROP ne nécessite aucune adaptation, si ce n'est qu'il faut utiliser la procédure RACCOR au lieu de SHORTEN. Pour adapter la procédure DROP_ADD, il suffit de remplacer ADD par ADD_ORIENTÉ.

Le PPR dans un graphe orienté a été étudié par Gün [Gün93] qui propose trois formulations différentes du problème afin de le résoudre à l'optimalité. Gün propose également des algorithmes polynomiaux pour quatre cas particuliers du PPRO.

1.2.3 Le PPR dans un graphe mixte

Soit $G = (V, E \cup A)$ un graphe mixte fortement connexe et un ensemble $R \subseteq E \cup A$ de liens à desservir. Corberán et al. [Cor00a] proposent de simplifier le graphe G de manière à ce que chaque sommet de V soit incident à un arc à desservir et que chaque arête de E se trouve dans R. L'algorithme suivant permet de simplifier un graphe mixte G et correspond à celui décrit dans [Cor00a].

Algorithme PPR_MIXTE_SIMPLIFIÉ

Étape 1. Construire le graphe $G_S = (V_R, R \cup A_S)$ tel qu'à chaque paire de sommets $v_i, v_j \in V_R$ correspondent dans A_S deux arcs de direction opposée (v_i, v_j) et (v_j, v_i). 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.

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

Étape 3. Enlever de A_S tout arc (v_i, v_j) parallèle et de coût égal à une arête de R.

Étape 4. Enlever de A_S tout arc (v_i, v_j) parallèle, de même direction et de coût égal à un arc de R.

Corberán et al. [Cor00a] ont développé une heuristique de recherche avec tabous pour le PPR mixte.

Telles que décrites à la section 1.2.1, il est certain que les méthodes de recherche locale et de descente s'arrêtent à la rencontre du premier optimum local. Or, les problèmes d'optimisation combinatoire possèdent pour la plupart un ensemble de solutions de taille importante comportant plusieurs optima locaux qui bloquent systématiquement les méthodes constructives et de recherche locale. D'autres types de méthodes, dont la recherche avec tabous, ont donc été développées de façon à dépasser les optima locaux rencontrés lors de l'exploration de l'ensemble des solutions. Ces méthodes, appelées méta-heuristiques, contrôlent et guident une heuristique interne de recherche locale pour lui permettre de franchir l'obstacle de l'optimalité locale. En particulier, la méthode de recherche avec tabous permet, dans certaines conditions, le passage d'une solution courante à une solution voisine dont le coût est supérieur à celui de la solution courante.

La méthode de recherche avec tabous développée par Corberán et al. utilise une procédure similaire à SHORTEN comme méthode interne de recherche locale. La solution initiale est 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.

L'algorithme qui suit correspond à la méthode constructive développée par Corberán et al. pour le PPR mixte et décrite dans [Mit99].

Algorithme PPR_MIXTE

Étape 1. Appliquer à G l'algorithme PPR_MIXTE_SIMPLIFIÉ de façon à obtenir un graphe simplifié $G_S = (V_R, R \cup A_S)$.

Étape 2. *(Orientation des arêtes)*

Poser $A_E = \varnothing$. 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:

Calculer $f_i = d^-(i) - d^+(i)$ et $f_j = d^-(j) - d^+(j)$.

Orienter (v_i, v_j) de v_i vers v_j si $f_i \geq f_j$ et v_j vers v_i sinon.

Insérer l'arête orientée dans A_E.

Étape 3. *(Construction d'une graphe connexe G')*

Soient $C_1, \ldots, C_c$ les composantes connexes de G_R.

Soient $\delta^-(i) = min\{c_{ji} : (v_j, v_i) \in E \cup A\}$ et $\delta^+(i) = min\{c_{ij} : (v_i, v_j) \in E \cup A_S\}$.

Attribuer un coût c_{ij}' à chaque arc (v_i, v_j) reliant deux composantes connexes de G_R tel que:

$$c_{ij}' = \begin{cases} c_{ij} + \delta^-(i) + \delta^+(i) & \text{si } f_i \leq 0 \text{ et } f_j \geq 0 \\ c_{ij} - \delta^+(i) + \delta^+(i) & \text{si } f_i > 0 \text{ et } f_j \geq 0 \\ c_{ij} + \delta^-(i) - \delta^-(i) & \text{si } f_i \leq 0 \text{ et } f_j < 0 \\ c_{ij} - \delta^+(i) - \delta^-(i) & \text{si } f_i > 0 \text{ et } f_j < 0 \end{cases}$$

Construire un graphe complet non orienté $G'' = (V'', E'')$ tel qu'à chaque composante connexe C_p de G_R corresponde dans V'' un sommet v_p'', $1 \leq p \leq c$, et tel qu'à chaque arête (v_p'', v_q'') de E'' est associé un coût c_{pq}'' tel que

$$c_{pq}'' = min\{ c_{ij}' : v_i \in C_p \text{ et } v_j \in C_q, \text{ ou } v_i \in C_q \text{ et } v_j \in C_p\}.$$

Déterminer un arbre maximal de coût minimum dans G'' et introduire les arcs correspondants dans G'. Si G' est symétrique, aller à l'étape 6.

Etape 4. *(Transformation de G' en un graphe symétrique)*

Construire un graphe $H = (V, A_E \cup A_E' \cup A_E'' \cup A_S)$ tel qu'à chaque arête (v_i, v_j) de E correspondent dans $A_E \cup A_E'$ deux arcs de direction opposée et de coût c_{ij} et tel qu'à chaque arc (v_j, v_i) de A_E soit associé un arc de coût nul (v_i, v_j) dans A_E''.

Algorithme PPR_MIXTE (suite)

Etape 4. *(Transformation de G' en un graphe symétrique)* (suite)

Attribuer à chaque arc de $A_E \cup A_E' \cup A_S$ une capacité infinie et à chaque arc de A_E'' une capacité de 2.

Calculer les valeurs $f_i = d^-(i) - d^+(i)$ pour chaque sommet v_i de G'.

Définir une disponibilité de valeur f_i sur chaque sommet v_i de H tel que $f_i > 0$ et une demande de $-f_i$ sur tout sommet v_i tel que $f_i < 0$.

Trouver un flot de coût minimum dans H satisfaisant les demandes. Soit x_{ij} la valeur du flot traversant un arc de $A_E \cup A_E' \cup A_E'' \cup A_S$. Pour chaque arc $(v_j, v_i) \in A_E''$ inverser l'orientation de l'arête (v_i, v_j) de 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 $(v_i, v_j) \in A_E \cup A_E' \cup A_S$.

Si G' est orienté aller à l'étape 6.

Étape 5. 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'.

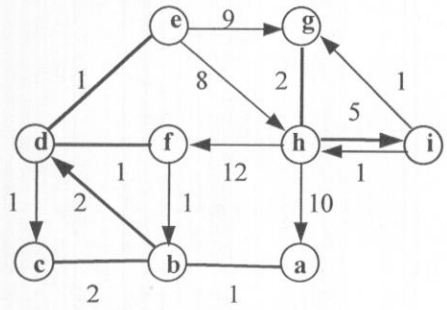
Construire un graphe complet non orienté $G_0 = (V_0, E_0)$.

Soit pcc_{ij} la longueur du 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 $c_{ij}^0 = min\{pcc_{ij}, pcc_{ji}\}$ 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'.

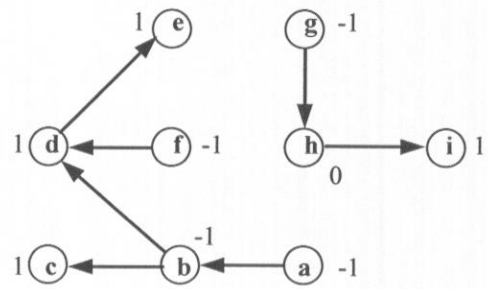
Étape 6. Déterminer un circuit eulérien dans G' en utilisant END_PAIRING adapté pour le cas orienté.

Corberán et al. [Cor00a] proposent un ordre particulier dans lequel les arêtes de E doivent être orientées à l'étape 2. Ils proposent également quelques variations sur la construction de l'arbre maximal de coût minimum à l'étape 3.

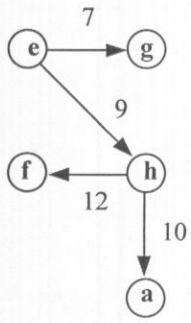
Le fonctionnement de l'algorithme PPR_MIXTE est illustré à l'aide de l'exemple suivant tiré de [Mit99]. La figure 1.21 (a) représente le graphe simplifié G_S. Le graphe G' obtenu après l'étape 2 apparaît à la figure 1.2 (b). Il est obtenu en orientant successivement les arêtes (a, b), (b, c), (d, e), (f, d) et (g, h). Les valeurs des f_i à la fin de l'étape 2 sont indiquées au-dessus des sommets de G'. La figure 1.21 (c) 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 3. G'' est représenté par la figure 1.2 (d). 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 1.21 (e)). Les valeurs f_e et f_g sont mises à jour dans G'.



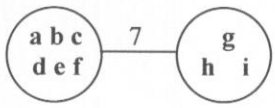
(a) Graphe simplifié G_S



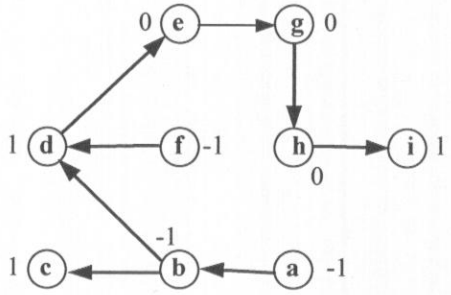
(b) Graphe G'



(c) Arcs connectant les composantes connexes de G_R



(d) Graphe G''

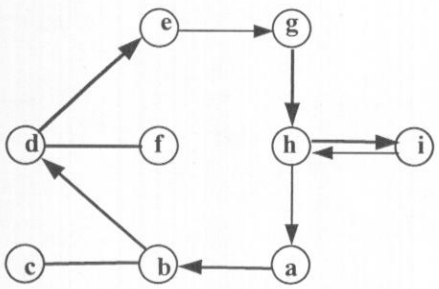


(e) Graphe G' à la fin de l'étape 2

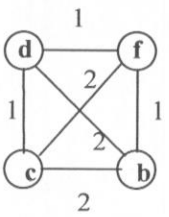
Figure 1.21: Exemple illustrant le fonctionnement de PPR_MIXTE

Le coût c_{ij}' d'un arc (v_i, v_j) tient compte du 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 l'exemple. En utilisant cet arc pour connecter les deux composantes connexes de G' (figure 1.21 (b)), les sommets e et g deviennent symétriques dans G' (figure 1.21 (e)). 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_S). De même, il faut entrer une fois de moins en g d'où un gain de $\delta_e^- = 1$ (= c_{ig} dans G_S). Le fait d'utiliser l'arc (e, g) pour connecter les deux composantes de G' permet d'éviter de rajouter dans G', à l'étape 4, deux arcs de coût supérieur ou égal à 1. Ainsi on a $c_{eg}' = c_{eg} - 1 - 1 = 9 - 1 - 1 = 7$.

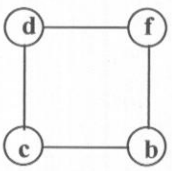
La figure 1.21 (f) représente le graphe G' obtenu à la fin de l'étape 4. Ce graphe contient encore deux arêtes à desservir qui ne sont pas orientées. La figure 1.21 (g) représente le graphe G_0 dans lequel un couplage parfait M de coût minimum est recherché. Les arêtes de ce couplage sont en pointillés. La figure 1.21 (h) représente l'union des arêtes de G' et de celles du couplage.



(f) Graphe symétrique G' à la fin de l'étape 3



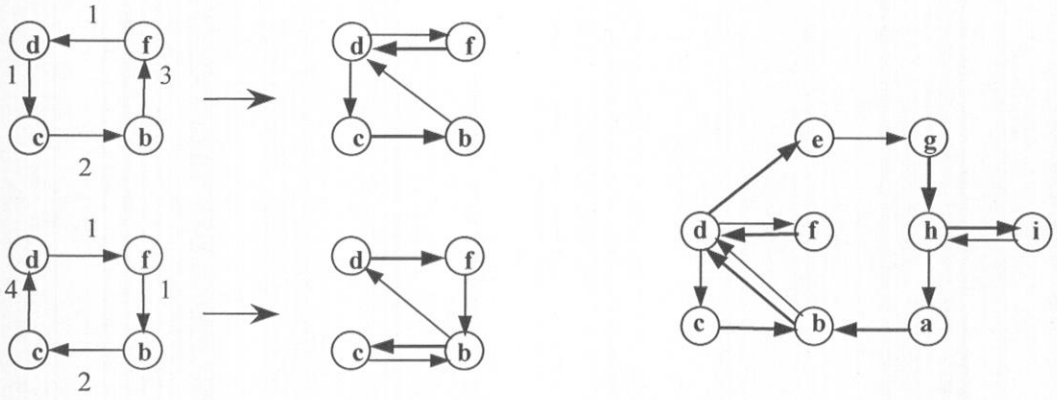
(g) Graphe G_0



(h) $E' \cup M$

Figure 1.21 (suite): Exemple illustrant le fonctionnement de PPR_MIXTE

La figure 1.21 (i) représente les deux orientations possibles du cycle induit par $E' \cup M$ dans le graphe G_0 ainsi que leurs orientations correspondantes dans G. Le meilleur circuit de ces deux orientations est ajouté à G' pour obtenir un graphe symétrique connexe et orienté (figure 1.21 (j)).



(i) Les deux orientations du cycle induit par $E' \cup M$

(j) Graphe final G'

Figure 1.21: Exemple illustrant le fonctionnement de PPR_MIXTE

Des algorithmes d'ordre polynomial pour le PPRM sur des structures en forme d'arbre ou de cercle ont été proposés par Anily, Gendreau et Laporte [Ani96; Ani97; Ani00].

Dans de nombreux contextes réels, des contraintes additionnelles de capacité doivent être satisfaites. Introduit par Golden et Wong [Gol81], le problème de tournées sur les arcs avec contrainte de capacité (PTAC) 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 $V = \{v_0, v_1, \ldots, v_n\}$ est l'ensemble des sommets, E l'ensemble des arêtes et A l'ensemble des arcs. Le sommet v_0 est appelé le dépôt où sont basés des véhicules de même capacité W. Comme dans le PPR, R représente le sous-ensemble de $E \cup A$ des liens à desservir (clients). À chaque

élément (v_i, v_j) de R est associé, en plus de son coût c_{ij}, une demande positive notée q_{ij}. Le PTAC consiste à trouver un ensemble de cycles ou de circuits partant du dépôt, revenant au dépôt et desservant des clients dont la demande totale n'excède pas W tel que la longueur totale soit minimum et que chaque élément de R soit traversé au moins une fois par un véhicule. Une excellente revue du PTAC est présentée par Mittaz [Mit99] qui propose également deux heuristiques basées sur des méthodes de recherche locale qu'il a développées pour résoudre le PTAC.

Pour des revues détaillées sur les PTA, le lecteur intéressé peut consulter Bodin et al. [Bod83], Fleischner [Fle90; Fle91], Eiselt, Gendreau et Laporte [Eis95I; Eis95II] ainsi que Assad et Golden [Ass95].

Chapitre 2

PTA avec pénalités pour les virages: revue des principales méthodes

Les problèmes de tournées sur les arcs avec pénalités pour les virages (PTA-PV), aussi appelés problèmes du postier à un véhicule avec pénalités pour les virages (PPUV-PV) [Mar98], sont des généralisations des problèmes classiques de tournées sur les arcs. Dans ce chapitre, nous allons présenter la définition de Martínez et Soler [Mar98] pour le PPUV-PV.

Le PPUV-PV est défini sur un graphe $G = (V, E \cup A)$ fortement connexe où V est l'ensemble des sommets, E l'ensemble des arêtes et A l'ensemble des arcs. Comme dans le cas du PPR, R représentera le sous-ensemble de $E \cup A$ des liens à desservir. À chaque élément $e_{ij} = (v_i, v_j)$ de $E \cup A$ est associé un coût non négatif c_{ij}. À chaque paire d'éléments $e_{ij} = (v_i, v_j)$, $e_{jk} = (v_j, v_k)$ de $E \cup A$ est associé un virage $[e_{ij}v_je_{jk}]$ résultant du passage de (v_i, v_j) à (v_j, v_k) au sommet v_j. Si e_{ij} et $e_{jk} \in E$, alors à cette paire est associé un autre virage $[e_{kj}v_je_{ji}]$ résultant du passage de (v_k, v_j) à (v_j, v_i) au sommet v_j. De plus, à chaque arête $e_{ij} = (v_i, v_j) \in E$ correspondent deux virages en «U»: $[e_{ji}v_ie_{ij}]$ au sommet v_i et $[e_{ij}v_je_{ji}]$ au sommet v_j. À chaque virage $[e_{ij}v_je_{jk}]$ dans G est associé une pénalité non négative $p_{[ijk]}$. Une pénalité $p_{[ijk]} = M$, où M est une constante assez élevée, est associée à tout virage $[e_{ij}v_je_{jk}]$ interdit dans G.

Une tournée admissible dans G est un cycle ou un circuit sans virage interdit, partant d'un sommet, revenant à ce même sommet et desservant les liens de R.

Résoudre le PPUV-PV revient à chercher une tournée admissible dont la longueur totale est minimum et telle que chaque élément de R est traversé au moins une fois par le véhicule.

Lorsque $A = \varnothing$ et $R = E$, ou que $E = \varnothing$ et $R = A$, ou encore $R = E \cup A$ et $A \neq \varnothing \neq E$, le PPUV-PV devient un problème du postier chinois avec pénalités pour les virages (PPC-PV). Lorsque $A = \varnothing$ et $R \subset E$, ou que $E = \varnothing$ et $R \subset A$, ou encore $R \subset E \cup A$ et $A \neq \varnothing \neq E$, le PPUV-PV devient un problème du postier rural avec pénalités pour les virages (PPR-PV). Martínez et Soler ont montré que le PPUV-PV est NP-difficile [Mar98] même pour les deux cas particuliers suivants: lorsque $E = \varnothing$, $R = A$ et G est eulérien; lorsque $A = \varnothing$, $R = E$, G est eulérien et $p_{[ijk]} = p_{[kji]}$ pour tout virage $[e_{ij}v_je_{jk}]$ dans G.

Les principales méthodes heuristiques et exactes développées pour la résolution des problèmes du postier chinois et rural avec pénalités pour les virages seront décrites dans cette section.

2.1 Le problème du postier chinois avec pénalités pour les virages

Le problème du postier chinois avec pénalités pour les virages (PPC-PV) est une généralisation du problème du postier chinois. Comme au chapitre précédent, les méthodes heuristiques et exactes pour le PPC-PV seront regroupées selon le type de graphe: non orienté, orienté et mixte.

2.1.1 Le PPC-PV dans un graphe non orienté

Soit $G = (V, E)$ un graphe non orienté connexe. Clossey [Clo98] propose d'appliquer d'abord à G les étapes 1 à 3 de l'algorithme PPC_NON_ORIENTÉ décrit pour le problème du postier chinois dans un graphe non orienté (cf. section 1.1.1 du

chapitre 1). Un tour eulérien est ensuite trouvé à l'aide d'une heuristique constructive ou d'une méthode exacte.

Une méthode heuristique est basée sur l'idée qu'en guidant de façon appropriée la construction des cycles à l'étape 4 de l'algorithme END-PAIRING, il est possible de minimiser la somme des pénalités associées aux virages. Soit $e_{ij} = (v_i, v_j)$ la dernière arête introduite dans le cycle C. Clossey propose les six stratégies suivantes pour déterminer la prochaine arête $e_{jk} = (v_j, v_k)$ non desservie et incidente à l'arête e_{ij} à introduire dans le cycle C.

Stratégie 1. Insérer l'arête e_{jk} selon l'ordre lexicographique.

Stratégie 2. Insérer l'arête e_{jk} qui minimise la pénalité $p_{[ijk]}$ associée au virage $[e_{ij}v_je_{jk}]$.

Stratégie 3. Insérer la paire d'arêtes e_{jk}, e_{kl} qui minimise la somme des pénalités $p_{[ijk]}$ et $p_{[jkl]}$ associées respectivement aux virages $[e_{ij}v_je_{jk}]$ et $[e_{jk}v_ke_{kl}]$.

Stratégie 4. Insérer l'arête e_{jk} qui minimise la somme des pénalités $p_{[ijk]}$ et $p_{[jkl]}$ associées respectivement aux virages $[e_{ij}v_je_{jk}]$ et $[e_{jk}v_ke_{kl}]$.

Stratégie 5. Insérer l'arête e_{jk} qui minimise le nombre d'arêtes e_{kl} non desservies et adjacentes à l'arête e_{jk}.

Stratégie 6. Insérer l'arête e_{jk} qui maximise le nombre d'arêtes e_{kl} non desservies et adjacentes à l'arête e_{jk}.

Clossey propose également de considérer les quatre façons de combiner les cycles C_1 et C_2 à l'étape 3 de l'algorithme END-PAIRING et de choisir la fusion dont la somme des pénalités associées aux virages est minimum. Par exemple, soient les cycles $C_1 = (a, b, d, a)$ et $C_2 = (b, c, f, e, b)$ construits sur le graphe de la figure 2.1 de la page suivante à l'étape 2 de l'algorithme END-PAIRING. Les quatre cycles suivants sont alors considérés: (a, b, c, f, e, b, d, a), (a, b, e, f, c, b, d, a), (a, d, b, c, f, e, b, a) et (a, d, b, e, f, c, b, a).

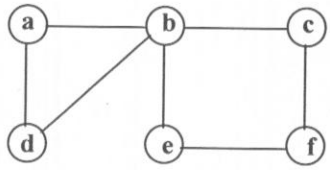


Figure 2.1: Graphe servant à illustrer la fusion de deux cycles pour le PPCNO-PV

La méthode exacte est basée sur la transformation d'un PTA en un problème du voyageur de commerce (PVC) asymétrique décrite dans [Lap97]. Soit $G = (V, A)$ un graphe orienté où $V = \{v_1, \ldots, v_n\}$ est l'ensemble des sommets et $A = \{(v_i, v_j) : v_i, v_j \in V\}$ est l'ensemble des arcs. À chaque arc est associé un coût non négatif c_{ij}. Le problème du voyageur de commerce (PVC) asymétrique consiste à trouver un circuit de coût minimum passant exactement une fois par chaque sommet. Le problème du postier chinois eulérien est transformé en un problème du voyageur de commerce (PVC) asymétrique lequel est ensuite résolu à l'aide de l'algorithme de Carpaneto et Toth [Car80]. Une autre méthode heuristique consiste à résoudre le PVC asymétrique à l'aide de l'heuristique PATCHING de Karp [Kar79]. L'algorithme suivant correspond à celui décrit dans [Lap97].

Algorithme TRANSFORMER_PTA_EN_PVC

Étape 1. Soit $G = (V, E \cup A)$ un graphe mixte.

Construire le graphe orienté $G' = (V, A \cup A')$ tel qu'à chaque arête (v_i, v_j) de E correspondent dans A' deux arcs de direction opposée et de coût c_{ij}.

Étape 2. Construire le graphe complet orienté $G'' = (V'', A'')$ où chaque sommet de V'' correspond à un arc de $A \cup A'$. Soit (v_j'', v_k'') un arc de A'' tel que le sommet v_j'' correspond à l'arc (v_i, v_j) dans G' et le sommet v_k'' correspond à l'arc (v_k, v_l) dans G', $i \neq k$ et $j \neq l$. Le coût d'un arc $(v_j'', v_k'') \in A''$ est égal à la somme du coût de l'arc (v_i, v_j) dans G' et de la longueur du plus court chemin pcc_{jk} reliant les sommets v_j et v_k dans G'.

Algorithme TRANSFORMER_PTA_EN_PVC (suite)

Étape 3. Identifier les ensembles disjoints de sommets S_1, ..., S_m de V'' où chaque ensemble est formé soit d'un sommet de V'' correspondant à un arc de A soit de deux sommets de V'' correspondant à la même arête de E.

Soit S_i un ensemble de sommets. Si $|S_i| = 2$, alors les coûts c_{ij}'' et c_{ji}'' des arcs reliant les deux sommets v_i'' et v_j'' de S_i égalent $-M$ où M est une constante positive élevée. De plus, si $|S_i| = 2$, permuter les coûts c_{ik}'' et c_{jk}'' des arcs reliant respectivement le sommet v_i'' de S_i au sommet v_k'' de S_k et le sommet v_j'' de S_i au sommet v_k'' de S_k.

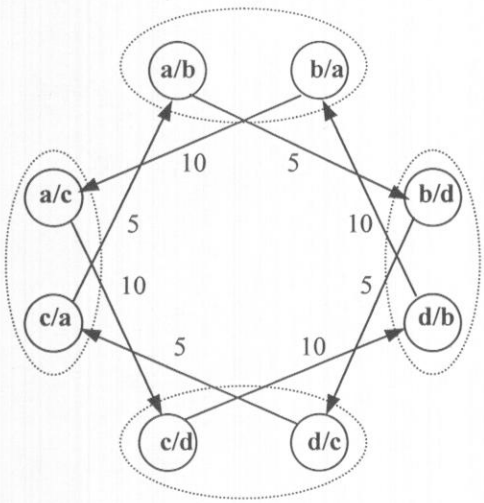
Dans le graphe initial de la figure 2.2 (a), les virages à droite reçoivent une pénalité de 5 et les virages à gauche une pénalité de 10.

(a) Graphe initial G (b) Graphe G'

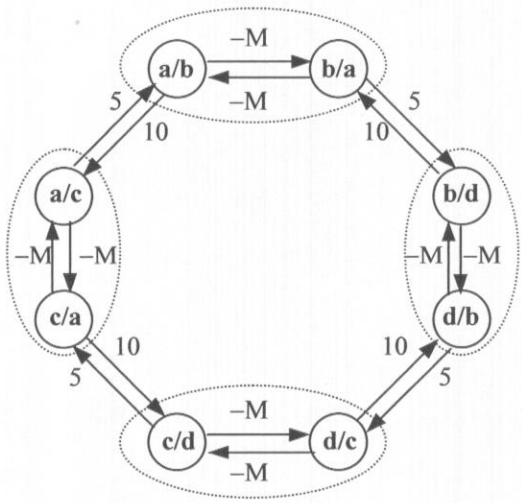
Figure 2.2: Transformation du PPC eulérien en un PVC asymétrique

Les arêtes (a, b), (b, d), (d, c) et (c, a) deviennent respectivement les sommets a/b et b/a, b/d et d/b, d/c et c/d, c/a et a/c formant ainsi quatre ensembles distincts tel qu'illustré à la figure 2.2 (c) de la page suivante. Les arcs du graphe G'' de la figure 2.2 (c) représentent les virages possibles dans le graphe initial G et les nombres situés près de ces arcs correspondent aux pénalités qui leur sont associées. Tous les autres arcs n'apparaissant pas sur le graphe G'' de la figure 2.2 (c) reçoivent un coût égal à M puisqu'ils ne correspondent pas à un virage possible dans le graphe G. Les coûts des deux arcs reliant les sommets d'un même ensemble sont ensuite fixés à $-M$ (figure 2.2

(d)) forçant ainsi l'algorithme à choisir deux sommets d'un même ensemble l'un à la suite de l'autre. Chaque arête du graphe initial G sera alors visitée une seule fois. Les coûts sur les arcs reliant les ensembles doivent cependant être modifiés pour tenir compte du fait que les sommets d'un même ensemble sont visités de façon consécutive. Par exemple, après avoir visité le sommet *a/b* sur le graphe de la figure 2.2 (d), l'algorithme est forcé de passer par le sommet *b/a* avant de visiter le sommet *b/d* de l'ensemble adjacent. Il faut donc remplacer le coût de l'arc (*b/a*, *b/d*) par celui de l'arc (*a/b*, *b/d*) pour prendre en considération le passage forcé de *a/b* vers *b/a*. En modifiant les coûts de tous les arcs reliant les ensembles, on obtient le graphe G'' tel qu'illustré à la figure 2.2 (d). Les arcs n'apparaissant pas sur le graphe G'' de la figure 2.2 (d) reçoivent un coût égal à M



(c) Graphe G'' après l'étape 2



(d) Graphe G'' après l'étape 3

Figure 2.2 (suite): Transformation du PPC eulérien en un PVC asymétrique

Toutes ces méthodes sont testées sur un ensemble d'instances générées aléatoirement. Étant donné que la transformation du PPC eulérien en un PVC asymétrique donne un graphe G'' de plus grande taille que celle du graphe initial G, la méthode exacte ne permet de résoudre que des problèmes de petite taille. Tel que décrit dans [Lap99], une transformation de ce type est d'un intérêt algorithmique

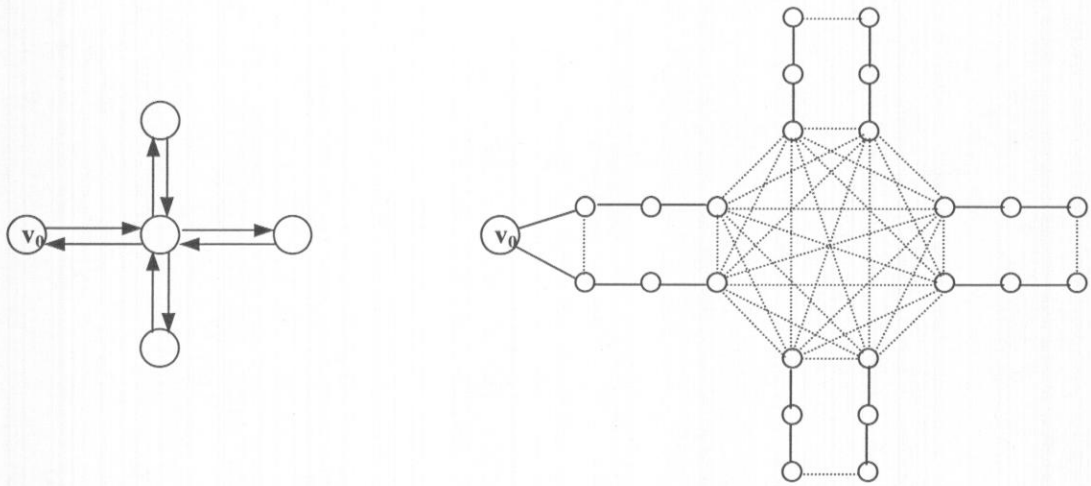
limité parce qu'elle introduit trop de dégénérescence dans le problème initial. Les résultats démontrent aussi que les stratégies 2 et 4 permettent d'obtenir des solutions de meilleure qualité que la méthode PATCHING.

Roy et Rousseau [Roy89] proposent d'associer des pénalités pour les changements et les traverses de rue dans le développement d'une heuristique pour le problème de tournées sur les arcs avec contrainte de capacité (PTAC) de la Société canadienne des postes (SCP).

Dans le PTAC de la SCP, $A = \varnothing$, $R = E$ et ce sont des personnes et non des véhicules qui desservent les clients. De plus, la limite sur la durée des tournées correspond à la contrainte de capacité.

Roy et Rousseau proposent une heuristique pour résoudre le PTAC de la SCP. Des sous-graphes connexes et pairs de G sont d'abord formés. Un sous-graphe G' de G est un graphe $G' = (V', E')$ où $V' \subset V$ et dont l'ensemble des arêtes est formé des arêtes de G ayant leurs extrémités dans V'. Un PPC-PV est défini dans chaque sous-graphe non orienté G' de G. Chaque sous-graphe G' de G est ensuite transformé en un sous-graphe orienté $G'' = (V', A)$. Chaque arête (v_i, v_j) de E' correspond à deux arcs de direction opposée dans A. Chaque sous-graphe G' de G est eulérien et $p_{[ijk]} = p_{[kji]}$ pour tout virage $[e_{ij}v_je_{jk}]$ dans G'. Un cycle eulérien minimisant les pénalités associées aux changements et aux traverses de rue est déterminé dans chaque sous-graphe en transformant le problème du postier chinois eulérien défini dans un sous-graphe G'' en un problème du voyageur de commerce (PVC) à l'aide de l'approche de Pearn, Assad et Golden [Pea86]. Roy et Rousseau utilisent un algorithme du plus proche voisin pour résoudre le PVC résultant de la transformation. Appliquée sur deux réseaux, la méthode a permis de réduire la durée des tournées de façon significative.

La figure 2.3 illustre la transformation du PPC eulérien en un PVC pour un sous-graphe donné G''. Les arcs dessinés en pointillés correspondent aux changements et aux traverses de rue auxquels sont associées des pénalités.



(a) Sous-graphe G'' (b) Graphe résultant de la transformation

Figure 2.3: Illustration de la transformation du PPC eulérien en un PVC

Soler [Sol95] étudie le problème du postier chinois non orienté sans virages en «U». Il montre que ce problème peut se résoudre en temps polynomial dans le cas où le graphe G est simple (i.e. lorsqu'il n'y a jamais plus d'une arête entre deux sommets).

2.1.2 Le PPC-PV dans un graphe orienté

Soit $G = (V, A)$ un graphe orienté fortement connexe. Si G est symétrique, Veerasamy [Vee99] montre que le problème de trouver un circuit eulérien en tenant compte des virages interdits dans G est NP-complet. Il montre également que si G est symétrique, le problème plus général de trouver un circuit eulérien en tenant compte des pénalités associées aux virages dans G est NP-difficile.

Soler [Sol95] étudie le problème du postier chinois orienté sans virages en «U». Il montre que ce problème peut se résoudre en temps polynomial lorsqu'une condition beaucoup plus restrictive que dans le cas non orienté est satisfaite. Le lecteur intéressé pourra consulter [Sol95].

Soler [Sol95] propose une méthode heuristique pour le problème du postier chinois orienté avec virages interdits. La méthode est basée sur la transformation du PPC orienté avec virages interdits en un PPR orienté. L'idée générale de cette transformation est de représenter chaque intersection par un sous-graphe. Cette façon de modéliser graphiquement une intersection a été suggérée à l'origine par Wattleworth et Shuldiner [Wat63] pour résoudre le problème de transbordement avec pénalités pour les virages. L'algorithme suivant unifie les principes de la transformation proposée par Wattleworth et Shuldiner [Wat63].

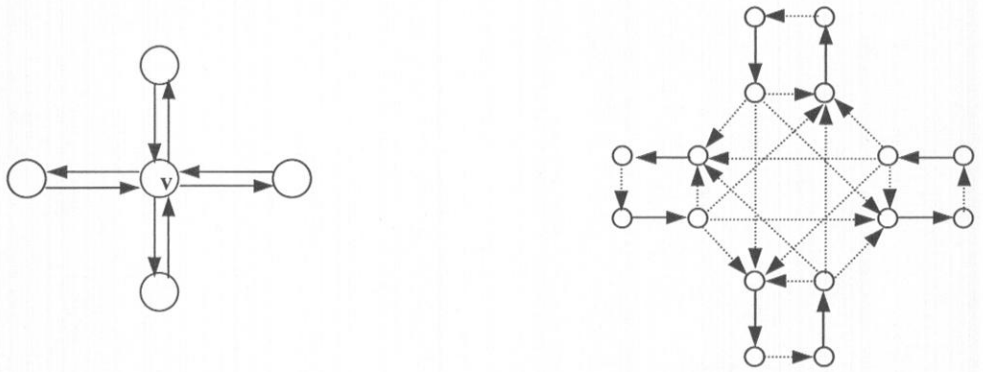
Algorithme TRANSFORMER_PTA_EN_PPR

Étape 1. Soit $G = (V, E \cup A)$ un graphe mixte.
Construire le graphe orienté $G' = (V, A \cup A')$ tel qu'à chaque arête (v_i, v_j) de E correspondent dans A' deux arcs de direction opposée et de coût c_{ij}.

Étape 2. Poser $G'' = G'$. Soit d_i le degré d'un sommet v_i de G''. Décomposer chaque sommet v_i de G'' en d_i sommets de façon à introduire dans G'' un sommet pour chacune des extrémités d'un arc de $A \cup A'$.

Étape 3. Soit S_i l'ensemble des d_i sommets correspondants au sommet v_i de G'. Soit T l'ensemble des virages possibles $[e_{ij}v_je_{jk}]$ résultant du passage de l'arc (v_i, v_j) à l'arc (v_j, v_k) au sommet v_j dans G'. Pour chaque virage $[e_{ij}v_je_{jk}]$ de T, ajouter à G'' un arc (v_m'', v_n'') tel que $v_m'', v_n'' \in S_j$, le sommet initial v_l'' de l'arc (v_l'', v_m'') appartient à S_i et le sommet terminal v_p'' de l'arc (v_n'', v_p'') appartient à S_k. Le coût de l'arc (v_m'', v_n'') de G'' est égal à la pénalité $p_{[ijk]}$ associée au virage $[e_{ij}v_je_{jk}]$ de T.

Dans le graphe initial G de la figure 2.4 (a), les pénalités associées aux traverses et aux virages à droite et à gauche sont respectivement 0, 5 et 10. Le sommet v est décomposé en huit sommets et des arcs de coût égal aux valeurs des pénalités pour les virages sont ajoutés en pointillés.



(a) Graphe initial G (b) Graphe G''

Figure 2.4: Représentation d'une intersection

Soler propose de résoudre le PPR orienté résultant de la transformation en utilisant diverses idées des heuristiques de Christofides et al. [Chr86] et de Ball et Magazine [Bal88] pour le PPR orienté. L'heuristique fournit une solution optimale pour chacune des instances testées.

2.1.3 Le PPC-PV dans un graphe mixte

À notre connaissance, les seules méthodes existantes permettant de résoudre le problème du postier chinois mixte avec pénalités pour les virages (PPCM-PV) sont celles développées par Martínez et Soler dans leur article sur le PPUV-PV [Mar98].

Martínez et Soler proposent une méthode exacte et une heuristique pour résoudre le PPUV-PV qui est une généralisation des problèmes du postier chinois et rural. Les deux méthodes sont basées sur une transformation polynomiale du PPUV-PV en un

PVC asymétrique obtenue à l'aide de l'algorithme présenté à la section 2.1.1. Le PVC asymétrique est ensuite résolu soit de façon exacte à l'aide de la méthode de séparation et d'évaluation progressive de Fischetti et Toth [Fis92] soit de façon approximative en utilisant l'heuristique PATCHING de Karp [Kar79]. Chaque méthode est testée sur un ensemble de 270 instances. Comme dans les expériences de Laporte [Lap97] et de Clossey [Clo98], les résultats montrent que la transformation fonctionne bien sur des graphes dont le nombre d'arêtes à desservir est relativement limité.

2.2 Le problème du postier rural avec pénalités pour les virages

2.2.1 Le PPR-PV dans un graphe non orienté

Le problème du postier rural avec pénalités pour les virages (PPR-PV) a également reçu peu d'attention, les seules méthodes connues à ce jour étant celles proposées par Martínez et Soler pour le PPUV-PV [Mar98] et décrites à la section précédente.

2.2.2 Le PPR-PV dans un graphe orienté

Une des premières applications du problème du postier rural avec pénalités pour les virages dans un graphe orienté (PPRO-PV) est décrite dans Bodin et Kursh [Bod78; Bod79] qui traitent de l'utilisation d'une flotte de balais mécaniques dans la ville de New York. Le réseau de la ville est représenté par un graphe fortement connexe et orienté $G = (V, A)$. Soit R un sous-ensemble de A des arcs à desservir. Étant donné une flotte homogène de balais mécaniques basés au dépôt, le problème étudié par Bodin et Kursh consistait à trouver un ensemble de tournées de durée limitée dont la longueur totale et le nombre de virages en «U» et à gauche sont minimisés, chaque arc de R étant traversé au moins une fois par un balai mécanique.

Bodin et Kursh ont développé une heuristique basée sur deux approches décrites dans Beltrami et Bodin [Bel74]. Soit m le nombre de balais mécaniques utilisés. La première approche, appelée *cluster-first–route-second*, consiste à partitionner le graphe G en m sous-graphes connexes et orientés. Un PPRO-PV est ensuite résolu sur chaque sous-graphe G' de G à l'aide de l'algorithme PPR-PV_ORIENTÉ décrit à la page suivante. La deuxième approche construit d'abord une tournée géante en résolvant un PPRO-PV sur le graphe G. Cette tournée est ensuite décomposée en m tournées admissibles en regard des contraintes de capacité et de limite de durée, d'où l'appellation *route-first–cluster-second.*

Dans la description de l'algorithme, le graphe G correspond soit au graphe représentant le réseau au complet si l'approche *route-first–cluster-second* est utilisée soit au sous-graphe sur lequel est défini un PPRO-PV si l'autre approche est appliquée. L'algorithme PPR-PV_ORIENTÉ comprend quatre étapes. La première étape consiste à transformer le graphe G en un graphe symétrique G'. Un circuit eulérien tenant compte des pénalités associées aux virages est ensuite déterminé dans chaque composante connexe de G' en résolvant un problème d'affectation à chaque sommet de G'. Le problème d'affectation est un cas particulier du problème de transport. Le réseau d'un problème général d'affectation comporte n sommets émetteurs qui chacun émettent exactement une unité de flot. Le réseau admet aussi n sommets récepteurs qui reçoivent chacun une unité de flot. Le but est d'affecter chaque sommet émetteur à un sommet récepteur de façon à ce que chaque sommet récepteur soit affecté à un seul sommet émetteur et que les coûts d'affectation soient minimisés. Ce problème apparaît de façon naturelle dans les entreprises qui doivent affecter leurs employés à différentes tâches.

La troisième étape consiste à relier les circuits formés dans chaque composante connexe. Le graphe résultant G' est finalement rendu fortement connexe.

L'algorithme PPR-PV_ORIENTÉ est similaire à ceux présentés dans [Bod78], [Ben99], [Tuc83] et [Ass95].

Algorithme PPR-PV_ORIENTÉ

Étape 1. *(Transformation de G en un graphe symétrique)*

Calculer la valeur $f_i = d^-(i) - d^+(i)$ pour chaque sommet $v_i \in G$.

Soit A l'ensemble des arcs de G. Déterminer l'ensemble des sommets émetteurs $I = \{v_i \in V_R : f_i > 0\}$ et l'ensemble des sommets récepteurs $J = \{v_j \in V_R : f_j < 0\}$. Construire le réseau $G_0 = (V_R, A \cup A_0)$ où A_0 est l'ensemble des arcs virtuels $(\cdot, v_i)$ associés aux sommets émetteurs v_i et des arcs virtuels $(v_j, \cdot)$ associés aux sommets récepteurs v_j. Attribuer à chaque arc virtuel $(\cdot, v_i)$ une disponibilité de valeur f_i et à chaque arc virtuel $(v_j, \cdot)$ une demande de valeur $|f_i|$. Attribuer à chaque arc de A une capacité infinie.

Trouver un flot compatible de coût minimum dans G_0 satisfaisant les demandes. Soit x_{ij} la valeur du flot traversant un arc de $A \cup A_0$.

Poser $G' = G$. Ajouter à G' x_{ij} copies de chaque arc (v_i, v_j) appartenant à A.

Étape 2. *(Construction des circuits)*

Soit A' l'ensemble des arcs de G'. Soit I_i l'ensemble des arcs $a_i = (v_j, v_i)$ incidents au sommet v_i dans G' et J_i l'ensemble des arcs $a_j = (v_i, v_j)$ émergents du sommet v_i dans G'.

Pour chaque sommet $v_j \in G'$, résoudre le problème d'affectation dans lequel chaque arc a_i de I_j représente un sommet émetteur et chaque arc a_j de J_j représente un sommet récepteur. Soit (v_i, v_j) et (v_j, v_k) deux arcs de G' qui désignent respectivement le sommet émetteur a_i et le sommet récepteur a_j. Le coût d'un arc (a_i, a_j) reliant un sommet émetteur a_i à un sommet récepteur a_j est égal à la pénalité $p_{[ijk]}$ associée au virage $[e_{ij}v_je_{jk}]$ résultant du passage de l'arc (v_i, v_j) à l'arc (v_j, v_k).

Étape 3. Soit x_{ij} la valeur du flot traversant un arc (a_i, a_j) dans la solution optimale du problème d'affectation. Si $x_{ij} = 1$, alors l'arc $a_i = (v_i, v_j)$ de I_j est affecté à l'arc $a_j = (v_j, v_k)$ de J_j et le virage $[e_{ij}v_je_{jk}]$ résultant du passage de l'arc (v_i, v_j) à l'arc (v_j, v_k) au sommet v_j est effectué dans G'.

Algorithme PPR-PV_ORIENTÉ (suite)

Étape 4. *(Combinaison des circuits)*

Combiner les circuits ayant un sommet en commun. Si un seul circuit est formé, STOP.

Étape 5. *(Combinaison des composantes connexes)*

Soient $C_1, C_2, \ldots, C_c$ les différentes composantes connexes de G' et soit PCC_{ij} la longueur d'un plus court chemin reliant v_i à v_j dans G. Soit pcc_{ij} la longueur du plus court chemin PCC_{ij} entre v_i et v_j dans G.

Construire un graphe complet $G'' = (V'', E'')$ tel qu'à chaque composante connexe C_p de G' corresponde dans V'' un sommet v_p'', $1 \leq p \leq c$, et tel qu'à chaque arête (v_p'', v_q'') de E'' est associé un coût c_{pq}'' tel que

$$c_{pq}'' = min\{pcc_{ij} + pcc_{ji} \mid v_i \in C_p \text{ et } v_j \in C_q\}.$$

Déterminer un arbre maximal de coût minimum dans G''.

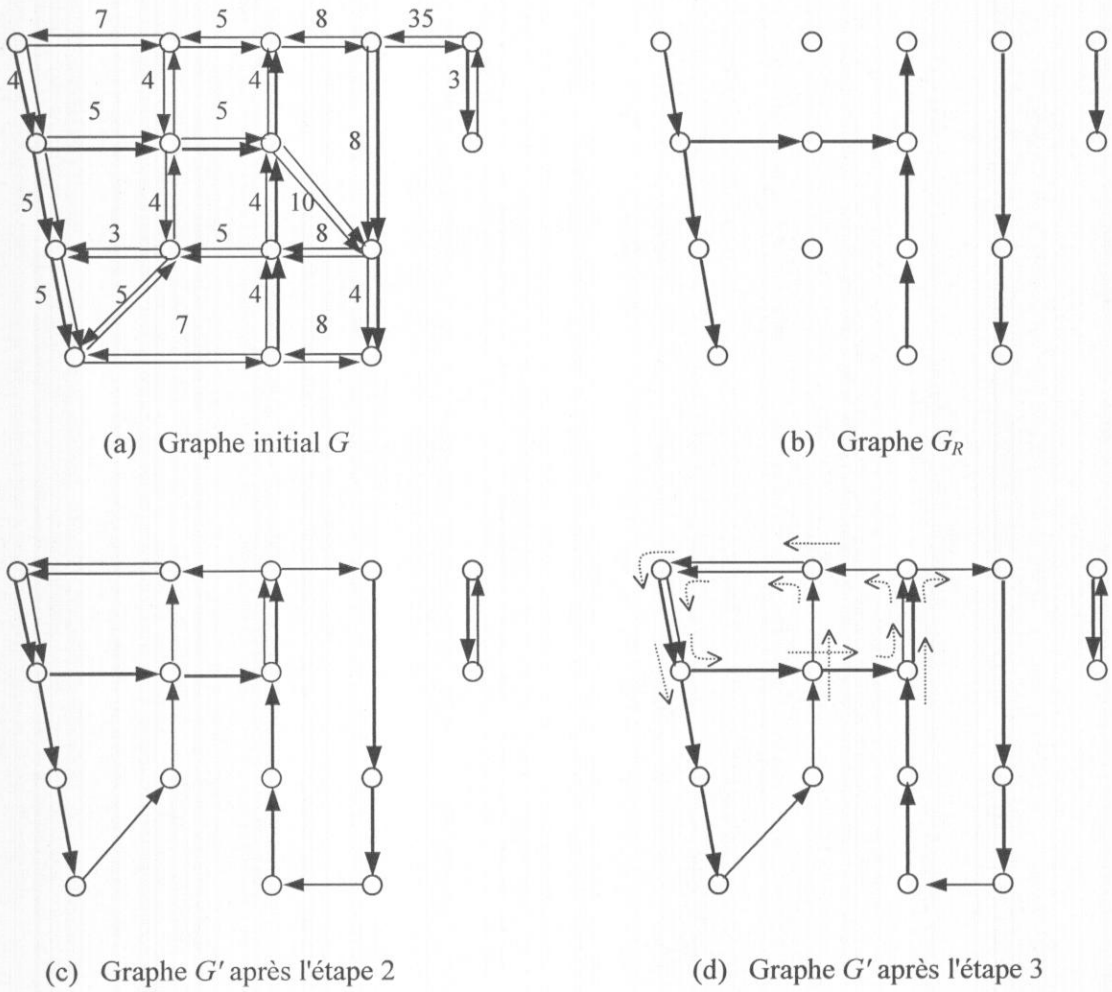
Pour chaque arête (v_p'', v_q'') de l'arbre optimal, considérer les sommets v_i de C_p et v_j de C_q tels que $c_{pq}'' = pcc_{ij} + pcc_{ji}$ et ajouter à G' tous les arcs de G appartenant à PCC_{ij} ainsi que tous les arcs de G appartenant à PCC_{ji}.

Retourner à l'étape 4.

La figure 2.5 de la page suivante illustre le fonctionnement de l'algorithme PPR-PV_ORIENTÉ. Cet exemple est tiré de [Bod79]. Le graphe initial G est représenté à la figure 2.5 (a). Les arcs à desservir sont dessinés en gras. Les nombres situés près des arcs correspondent aux coûts c_{ij} ($c_{ij} = c_{ji}$ pour toute paire d'arcs $(v_i, v_j), (v_j, v_i) \in A$). Les pénalités associées aux traverses, aux virages à droite, à gauche et aux virages en «U» sont respectivement de 0, 1, 4 et 8. Le sous-graphe G_R engendré par R comprend trois composantes connexes (figure 2.5 (b)).

Le graphe symétrique G' apparaît à la figure 2.5 (c). Le graphe G' est maintenant composé de deux composantes connexes. La figure 2.5 (d) représente le graphe G' obtenu à la fin de l'étape 3. Les flèches dessinées en pointillés indiquent les virages

effectués aux intersections. Trois circuits résultent de la solution au problème d'affectation.



(a) Graphe initial G

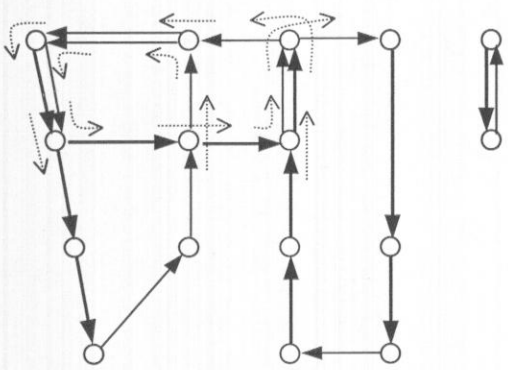
(b) Graphe G_R

(c) Graphe G' après l'étape 2

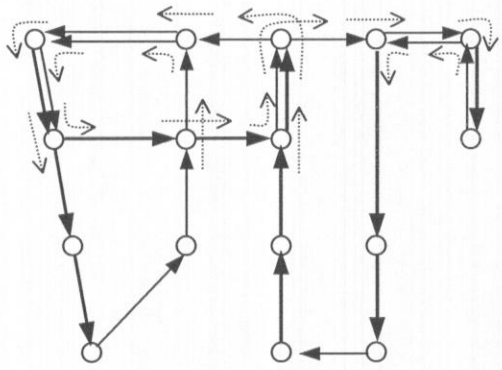
(d) Graphe G' après l'étape 3

Figure 2.5: Illustration de l'algorithme PPR-PV_ORIENTÉ

La figure 2.5 (e) de la page suivante représente le graphe G' obtenu après avoir combiné les deux circuits de la même composante connexe. Les deux composantes connexes sont finalement reliées pour former le graphe symétrique connexe et orienté G' de la figure 2.5 (f).



(e) Graphe G' après l'étape 4



(f) Graphe G' après l'étape 5

Figure 2.5 (suite): Illustration de l'algorithme PPR-PV_ORIENTÉ

Des études pilote montrent que l'approche *route-first–cluster-second* permet de générer des tournées comprenant moins de passages à vide que l'approche *cluster-first–route-second* et moins de virages en «U». Cependant, lorsque les tournées sont générées avec l'approche *route-first–cluster-second*, elles s'entrecroisent et se chevauchent alors qu'avec l'approche *cluster-first–route-second* les chevauchements sont considérablement réduits.

McBride [McB82] propose certains raffinements pour l'algorithme PPR-PV_ORIENTÉ de Bodin et Kursh afin de contrôler le nombre de virages à gauche et en «U». De tels virages peuvent être éliminés en ajoutant des passages à vide forcés dans le graphe initial G. Ces passages à vide forcés sont en fait des arcs artificiels ajoutés à l'ensemble R des arcs à desservir forçant ainsi l'algorithme à desservir obligatoirement ces arcs. Cette idée est illustrée à l'aide de l'intersection de la figure 2.6 (a). Tous les arcs représentés sont des arcs à desservir. Le sommet v étant déjà symétrique, aucun passage à vide n'est ajouté après l'étape 1 de l'algorithme. Deux virages à gauche sont nécessairement introduits au sommet v après l'étape 3 de l'algorithme. À la figure 2.6 (b), deux passages à vide forcés sont ajoutés à l'intersection initiale. L'algorithme doit alors couvrir les deux nouveaux arcs

artificiels à desservir et les deux virages à gauche sont remplacés par une traverse et deux virages à droite.

(a) Intersection initiale (b) Intersection finale

Figure 2.6: Exemple illustrant l'introduction de passages à vide forcés

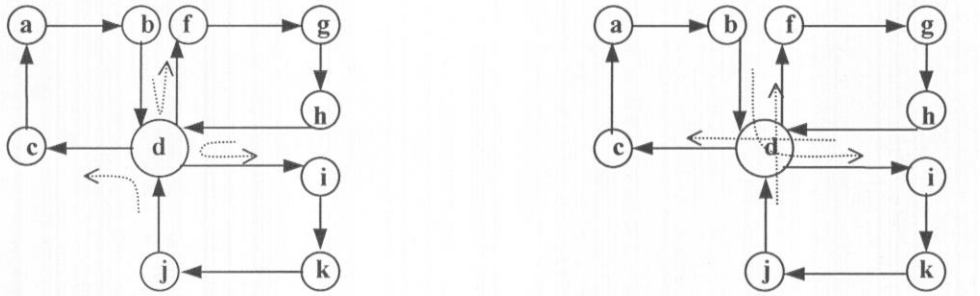
McBride propose aussi de contrôler le nombre de virages à gauche et en «U» lors de la combinaison des circuits à l'étape 4 de l'algorithme. Une façon de réduire le nombre de virages à gauche et en «U» est de prioriser la fusion de deux circuits ayant un arc en commun. À titre d'exemple, considérons les fusions de circuits illustrées à la figure 2.7. La combinaison des deux circuits de la figure 2.7 (a) remplace les deux virages à droite $[e_{ab}v_be_{bd}]$ et $[e_{hb}v_be_{bf}]$ au sommet b par les virages à gauche $[e_{ab}v_be_{bf}]$ et $[e_{hb}v_be_{bd}]$. Les deux circuits de la figure 2.7 (b) ayant un arc en commun, aucun virage n'est modifié par la fusion.

(a) Fusion de deux circuits (b) Fusion de deux circuits ayant un arc en commun

Figure 2.7: Combinaison de circuits ayant un arc en commun

Une autre façon de réduire le nombre de virages à gauche et en «U» est de relier, si possible, trois circuits ayant un sommet en commun. À titre d'exemple, considérons

les trois circuits $C_1 = (a, b, d, c, a)$, $C_2 = (f, g, h, d, f)$ et $C_3 = (i, k, j, d, i)$ de la figure 2.8 (a). Une première façon de relier ces trois circuits est de relier premièrement les circuits C_1 et C_2. Les virages $[e_{bd}v_de_{dc}]$ et $[e_{hd}v_de_{df}]$ sont alors remplacés par le virage en «U» $[e_{bd}v_de_{df}]$ et la traverse $[e_{hd}v_de_{dc}]$ pour former le nouveau circuit $C_4 = (a, b, d, f, g, h, d, c, a)$. La fusion des circuits C_3 et C_4 remplace ensuite les virages $[e_{hd}v_de_{dc}]$ et $[e_{jd}v_de_{di}]$ par le virage en «U» $[e_{hd}v_de_{di}]$ et le virage à gauche $[e_{jd}v_de_{dc}]$.



(a) Fusion de trois circuits

(b) Fusion de trois circuits ayant un sommet en commun

Figure 2.8: Combinaison de circuits ayant un sommet en commun

Une autre façon de relier les circuits C_1, C_2 et C_3 est illustrée à la figure 2.8 (b). Les trois circuits ayant un sommet en commun, ils peuvent alors être tous combinés en même temps pour former le circuit $(a, b, d, i, k, j, d, f, g, h, d, c, a)$. Il y a toujours deux façons de fusionner trois circuits. La fusion de coût minimum est alors choisie.

Benavent et Soler [Ben99] présentent et comparent deux méthodes constructives CONS1 et CONS2 pour résoudre le PPRO-PV ainsi qu'une méthode améliorative IMPROVE. Les méthodes constructives sont basées sur les principes de l'algorithme PPR-PV_ORIENTÉ de Bodin et Kursh. Deux modifications importantes de l'algorithme sont cependant proposées par les auteurs. La première modification consiste à générer des tournées exemptes de virages interdits contrairement aux tournées obtenues par la méthode de Bodin et Kursh à l'aide d'une opération similaire

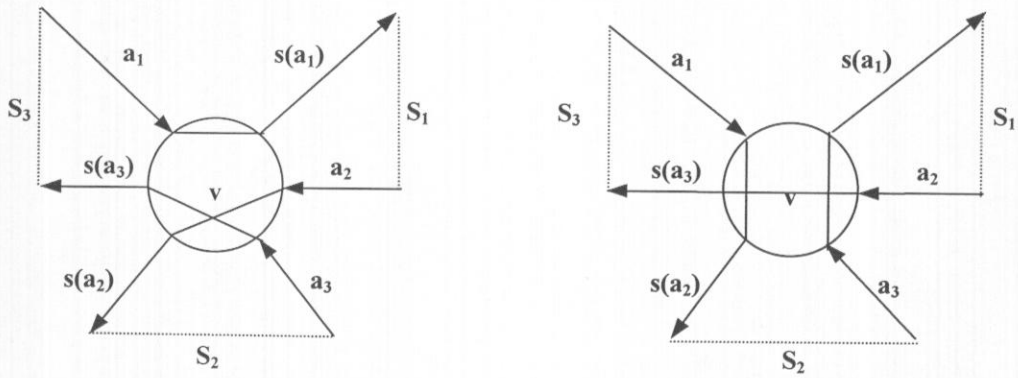
à la procédure RACCOR décrite à la section 1.2.2 du chapitre 1. Benavent et Soler proposent aussi d'améliorer l'étape 4 de l'algorithme PPR-PV_ORIENTÉ en appliquant les deux stratégies proposées par McBride soient la fusion des circuits ayant un arc en commun et la fusion de circuits ayant un sommet en commun.

La méthode constructive CONS1 utilise la méthode BALANCE_CONNECT de Ball et Magazine décrite à la section 1.2.2 du chapitre 1 pour le PPRD afin d'obtenir un graphe symétrique et connexe G'. Un problème d'affectation est ensuite résolu à chaque sommet de G' comme dans l'algorithme PPR-PV_ORIENTÉ dans le but de générer un ensemble de circuits qui seront fusionnés par la suite. Les virages interdits sont finalement éliminés.

La deuxième heuristique CONS2 consiste premièrement à résoudre un problème d'affectation impliquant tous les arcs requis de R au lieu de résoudre plusieurs problèmes d'affectation à chaque sommet comme dans la première heuristique. La solution du problème d'affectation fournit une borne inférieure au PPRO-PV. À partir de cette solution, un ensemble de circuits traversant au moins une fois chaque arc à desservir est construit. Les circuits sont ensuite fusionnés pour former une seule tournée sans introduire de virage interdit.

L'heuristique améliorative IMPROVE est basée sur deux opérations modifiant localement une solution du PPRO-PV. Une opération est similaire à la procédure RACCOR décrite à la section 1.2.2 du chapitre 1. Une autre opération consiste à modifier trois virages effectués au sommet v d'une tournée pour obtenir une nouvelle tournée qui diffère de la première uniquement par les virages effectués à ce sommet. À titre d'exemple, considérons le circuit suivant de la figure 2.9 (a): a_1, $s(a_1)$, S_1, a_2, $s(a_2)$, S_2, a_3, $s(a_3)$, S_3 où a_1, a_2 et a_3 sont des arcs, $s(a_1)$, $s(a_2)$ et $s(a_3)$ sont respectivement leurs arcs successeurs et S_1, S_2 et S_3 sont des sections du circuit. L'opération consiste à remplacer les trois virages $[a_1, v, s(a_1)]$, $[a_2, v, s(a_2)]$ et $[a_3, v, s(a_3)]$ par les virages $[a_1, v, s(a_2)]$, $[a_2, v, s(a_3)]$ et $[a_3, v, s(a_1)]$. La figure 2.9 (b)

illustre le nouveau circuit a_1, $s(a_2)$, S_2, a_3, $s(a_1)$, S_1, a_2, $s(a_3)$, S_3 traversant les mêmes arcs dans un ordre différent.



(a) Circuit initial

(b) Circuit résultant de l'opération

Figure 2.9: Exemple d'une opération modifiant trois virages à un sommet

Les heuristiques proposées ont été testées sur un ensemble de 30 instances comprenant jusqu'à 180 sommets et 666 arcs, six d'entre elles correspondant à des problèmes réels. Les résultats démontrent que l'heuristique constructive CONS2 suivie de l'heuristique améliorative IMPROVE permet d'obtenir les meilleurs résultats.

2.2.3 Le PPR-PV dans un graphe mixte

Dans [Bod89], Bodin, Fagin, Welebny et Greenberg adaptent l'algorithme PPR-PV_ORIENTÉ de Bodin et Kursh pour le développement d'un système de planification de la collecte des ordures. Le sous-graphe $G_R = (V_R, R)$ de G représentant le réseau de rues à desservir est d'abord divisé en deux sous-graphes: un sous-graphe non orienté G_1 et un sous-graphe orienté G_2. Il y a deux arcs à desservir (v_i, v_j) et (v_j, v_i) entre chaque paire de sommets v_i, v_j dans G_2. Le sous-graphe non orienté G_1 représente les rues qui doivent être desservies dans un sens ou dans l'autre, la collecte étant effectuée pour les deux côtés de la rue à la fois. Le sous-graphe

orienté G_2 permet de représenter les rues où les rebus sont ramassés en deux fois, sur un seul côté de la rue à la fois.

L'approche *route-first–cluster-second* est utilisée pour la construction des tournées. Un problème du postier chinois non orienté est d'abord résolu sur chaque composante connexe C_i de G_1 en appliquant premièrement les étapes 1 à 3 de l'algorithme PPC_NON_ORIENTÉ présenté à la section 1.1.1 du chapitre 1. Un cycle eulérien tenant compte des pénalités associées aux virages est ensuite déterminé dans chaque composante connexe augmentée C_i' de G_1 en adaptant l'étape 2 de l'algorithme PPR-PV_ORIENTÉ. Au lieu d'un problème d'affectation, un problème de couplage est résolu à chaque sommet de C_i' dans le but de générer un ensemble de circuits. Les arcs de G_2 ayant un sommet en commun avec un circuit T de G_1 sont ensuite insérés dans T jusqu'à ce qu'une seule tournée soit formée en ajoutant, si nécessaire, des liens qui n'appartiennent pas à l'ensemble R. La tournée géante est finalement partitionnée en un ensemble de tournées admissibles. Le système permet de réduire les coûts de manière significative.

Corberán et al. [Cor00b] présentent et comparent deux méthodes heuristiques et une méthode exacte pour le PPRM-PV et virages interdits. Une méthode heuristique et la méthode exacte utilisent la transformation d'un PTA en un PVC asymétrique décrite à section 2.1.1. Le PVC asymétrique est ensuite résolu soit de façon exacte à l'aide de la méthode de séparation et d'évaluation progressive de Fischetti et Toth [Fis92] soit de façon heuristique à l'aide de la méthode PATCHING de Karp [Kar79]. La deuxième heuristique comprend trois étapes distinctes. Toute solution au PPRM, sans tenir compte des pénalités associées aux virages et des virages interdits, dont l'écart par rapport à la meilleure solution obtenue est inférieur à un certain seuil est d'abord recherchée à l'aide de l'algorithme PPR_MIXTE présenté à la section 1.2.3 du chapitre 1. Les tournées obtenues sont ensuite rendues admissibles pour le PPRM-PV et virages interdits en résolvant un problème d'affectation à chaque sommet d'une tournée et en combinant les circuits ainsi formés. Il s'agit de la procédure CONS1 de

Benavent et Soler [Ben99]. Finalement, la méthode améliorative IMPROVE de Benavent et Soler [Ben99] est appliquée aux tournées admissibles. Chaque méthode heuristique est testée sur des nouveaux ensembles de données. La méthode heuristique basée sur la solution du PPRM à l'aide de l'algorithme PPR_MIXTE produit les meilleurs résultats.

Bousonville et Kopfer [Bou00] proposent un algorithme génétique pour le PPRM-PV. Les algorithmes génétiques, initialement proposés par Fogel et al. [Fog66], sont des méta-heuristiques dont les principes reposent sur une analogie avec l'évolution des espèces. Un algorithme génétique explore l'espace des solutions en analysant, à chaque itération, une population (i.e. un groupe de solutions) et non seulement une solution unique. L'idée est de combiner des solutions d'une population pour générer de nouvelles solutions composées des bonnes caractéristiques des solutions mères et exemptes de leurs mauvaises caractéristiques. En se basant sur les algorithmes génétiques proposés dans la littérature pour le problème du voyageur de commerce, Bousonville et Kopfer développent une approche hybride pour le PPRM-PV qui unifie les principes des algorithmes génétiques et qui utilise des procédures de recherche locale. L'algorithme est comparé à l'heuristique de Corberán et al. [Cor00b] pour le PPCM-PV. Les résultats montent que l'algorithme génétique produit des solutions de meilleure qualité mais nécessitant des temps de calcul pouvant aller jusqu'à dix heures lorsque la taille du problème et celle de la population deviennent importantes.

Il est rare que les méthodes décrites dans ce chapitre s'appliquent directement à des problèmes réels. Elles peuvent cependant servir de base à d'autres méthodes qui sont adaptées à de tels problèmes et qui tiennent compte des contraintes additionnelles pouvant s'ajouter. Plusieurs auteurs ont adapté certaines de ces méthodes à des contextes réels où le nombre de virages à gauche ou en «U» doit être réduit. Robinson, Ogawa et Frickenstein [Rob90] par exemple utilisent l'approche de Bodin et Kursh [Bod78;Bod79] tout en minimisant le nombre de virages en «U».

D'autres problèmes pratiques ont été résolus de façon à minimiser le nombre de certains types de virages notamment dans [Shu74], [Tuc79] et [Ben72].

D'autres problèmes de réseaux et d'optimisation combinatoire où la réduction et l'interdiction de certains types de virages sont prises en considération ont été étudiés dans la littérature.

Le problème du plus court chemin avec pénalités pour les virages a été étudié, entre autres, par Kirby et Potts [Kir69] qui présentent une excellente revue de ce problème. La considération des pénalités pour les virages dans le problème de l'allocation et de l'estimation du trafic a été étudiée notamment par Woods et al. [Woo69], Bronzini [Bro70] et Davinroy et al. [Dav69]. Namkoong et al. [Nam98] modifient l'algorithme de Dijsktra pour résoudre le problème du plus court chemin dans un réseau avec interdiction pour certains virages à gauche conjointement avec l'autorisation pour les virages en «U» et en «P». Singer [Sin74] propose une méthode de résolution utilisant l'algorithme TRANSFORMER_PTA_EN_PPR présenté à la section 2.1.2. Lee et al. [Lee97] développent un modèle graphique avec pénalités pour les virages pour le problème de tournées de véhicules téléguidés et automatisés (en anglais, Automated Guided Vehicles: AGV). Allen [All90] présente une méthode basée sur des modèles théoriques d'estimation des délais aux intersections pour simuler les pénalités pour les virages. Boroujerdi et Uhlmann [Bor98a; Bor98b] et Park et al. [Par98] modifient efficacement l'algorithme de Dijkstra pour tenir compte des virages interdits.

Laporte et al. [Lap89] ont développé un algorithme exact qui respecte la réglementation de la circulation et de l'union postale dans le problème de la levée des boîtes aux lettres de la Société canadienne des postes. Le problème de la levée des boîtes aux lettres est un problème de tournées sur les sommets où les sommets représentent les boîtes aux lettres à être visitées par les véhicules. Plusieurs règlements de la circulation tels l'interdiction des virages en «U» et des virages à

gauche doivent être respectés dans la détermination des tournées. Une difficulté reliée à ce problème concerne le calcul des plus courts chemins entre les boîtes aux lettres. À titre d'exemple, considérons le réseau de la figure 2.10 dans lequel les boîtes aux lettres A et B sont localisées. Le calcul du plus court chemin de A à B dépend du point d'origine du véhicule avant d'arriver à la boîte aux lettres A. Si le véhicule utilise le chemin P1, les boîtes aux lettres A et B sont alors visitées sur la même rue. Si le véhicule emprunte le chemin P2, la boîte aux lettres A est alors visitée le long du segment AD et le véhicule doit faire le tour du bloc pour atteindre la boîte aux lettres B. Pour calculer la longueur des chemins P1 et P2, Laporte et al. proposent de remplacer chaque boîte aux lettres par une copie sur chaque segment adjacent à l'intersection où est localisée la boîte aux lettres. Sur la figure 2.10, les boîtes aux lettres A et B sont donc remplacées respectivement par les boîtes aux lettres A1, A2 et B1, B2. Ainsi, la longueur du chemin P1 correspond à la longueur du plus court chemin de A2 à B2 et celle du chemin P2 correspond à la longueur du plus court chemin de A1 à B2.

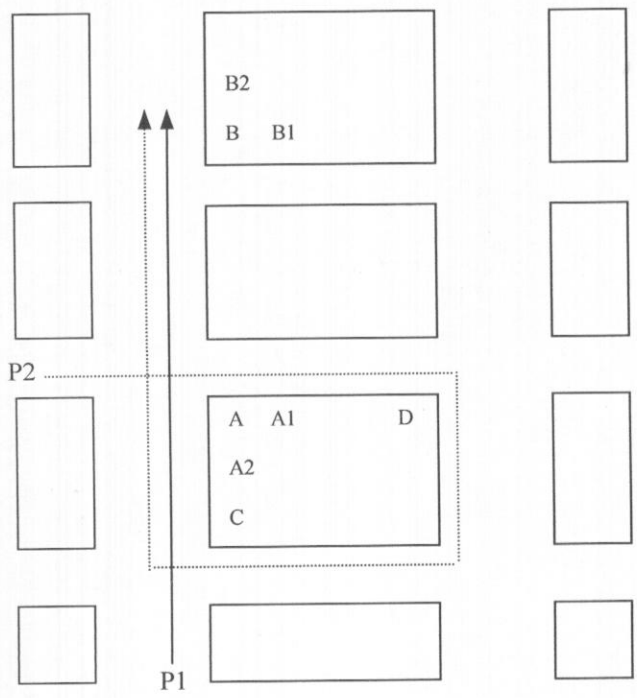


Figure 2.10: Réseau illustrant le calcul des plus courts chemins

Chapitre 3

Formulation du PPCM-PV

Dans ce chapitre, la définition du PPCM-PV est basée sur celle du PPUV-PV proposée par Martínez et Soler dans [Mar98].

Le problème du postier chinois mixte avec pénalités pour les virages (PPCM-PV) est une généralisation du problème du postier chinois mixte. Le PPCM-PV est défini sur un graphe $G = (V, E \cup A)$ fortement connexe où $V = \{v_0, v_1, \ldots, v_n\}$ est l'ensemble des sommets de G, $E \subseteq \{(v_i, v_j) : i \neq j\}$ est l'ensemble des arêtes du graphe et $A \subseteq \{(v_i, v_j) : i \neq j\}$ l'ensemble des arcs. Les liens de $E \cup A$ désignent les clients à desservir. Le sommet v_0 représente le sommet de départ et d'arrivée d'une tournée et les autres sommets représentent les intersections de liens.

À chaque élément (v_i, v_j) de $E \cup A$ est associé un coût non négatif c_{ij} représentant la longueur du lien (v_i, v_j) de $E \cup A$. Tous les coûts associés aux arêtes sont symétriques (i.e. $c_{ij} = c_{ji}$ pour tout $(v_i, v_j) \in E$). Un virage $[e_{ij}v_je_{jk}]$ résultant du passage de (v_i, v_j) à (v_j, v_k) au sommet v_j de G est associé à chaque paire d'éléments $e_{ij} = (v_i, v_j)$, $e_{jk} = (v_j, v_k)$ de $E \cup A$. L'élément (v_i, v_j) représente le lien initial du virage $[e_{ij}v_je_{jk}]$ et l'élément (v_j, v_k) le lien terminal. Si e_{ij} et $e_{jk} \in E$, alors à cette paire est également associé un virage $[e_{kj}v_je_{ji}]$ résultant du passage de (v_k, v_j) à (v_j, v_i) au sommet v_j. De plus, à chaque arête $e_{ij} = (v_i, v_j) \in E$ correspondent deux virages en «U»: le virage $[e_{ji}v_ie_{ij}]$ au sommet v_i et le virage $[e_{ij}v_je_{ji}]$ au sommet v_j. Une pénalité non négative $p_{[ijk]}$ est associée à chaque virage $[e_{ij}v_je_{jk}]$ dans G et une pénalité

$p_{[ijk]} = M$, où M est une constante élevée, est associée à chaque virage $[e_{ij}v_je_{jk}]$ interdit dans le graphe G.

Une tournée admissible pour le PPCM-PV est un circuit sans virage interdit partant du sommet v_0, revenant au sommet v_0 et desservant au moins une fois chaque lien du graphe. La longueur d'une tournée admissible est la somme des coûts associés aux liens traversés par la tournée et des pénalités associées aux virages faisant partie de la tournée. Le problème consiste à construire une tournée admissible dont la longueur totale est minimum.

Le PPCM-PV peut être formulé comme un programme linéaire en nombres entiers. Définissons $A^+(v_k) = \{(v_i, v_j) \in A : v_i = v_k\}$, $A^-(v_k) = \{(v_i, v_j) \in A : v_j = v_k\}$ et $V_k = \{v_i \in V : (v_i, v_k) \in E \text{ ou } (v_k, v_i) \in E\}$, l'ensemble de tous les sommets reliés au sommet v_k par une arête. Définissons aussi $\Gamma^+(v_k)$ l'ensemble des successeurs du sommet v_k de V et $\Gamma^-(v_k)$ l'ensemble des prédécesseurs du sommet v_k de V. Le sommet v_i est un successeur de v_k s'il existe un lien (v_k, v_i) ayant son extrémité initiale en v_k et son extrémité terminale en v_i. Le sommet v_i est un prédécesseur de v_k s'il existe un lien (v_i, v_k) ayant son extrémité initiale en v_i et son extrémité terminale en v_k. Soit $T = \{[e_{ij}v_je_{jk}] : (v_i, v_j), (v_j, v_k) \in E \cup A\}$ l'ensemble des virages permis dans G.

Le modèle comprend des variables x_{ij} et y_{ij} qui dénotent respectivement le nombre de fois que l'arc (v_i, v_j) est traversé et le nombre de fois que l'arête (v_i, v_j) est traversée de v_i à v_j. Il y a donc deux variables y_{ij} et y_{ji} associées à chaque arête (v_i, v_j) de E. Une variable $n_{[ijk]}$ est également introduite afin de représenter le nombre de fois que le virage permis $[e_{ij}v_je_{jk}]$ de T du lien inital (v_i, v_j) au lien terminal (v_j, v_k) au sommet v_j de G apparaît dans une solution optimale.

Le modèle mathématique du PPCM-PV présenté à la page suivante est basé sur la formulation du PPCM proposée par Christofides et al. dans [Chr84].

Minimiser

$$\sum_{(v_i,v_j)\in A} c_{ij}x_{ij} + \sum_{(v_i,v_j)\in E} c_{ij}(y_{ij}+y_{ji}) + \sum_{[e_{ij}v_je_{jk}]\in T} p_{[ijk]}n_{[ijk]} \tag{3.1}$$

sous les contraintes

$$\sum_{(v_i,v_j)\in A^+(v_k)} x_{ij} - \sum_{(v_i,v_j)\in A^-(v_k)} x_{ij} + \sum_{v_j\in V_k} y_{kj} - \sum_{v_j\in V_k} y_{jk} = 0 \qquad (v_k \in V) \tag{3.2}$$

$$y_{ij} + y_{ji} \geq 1 \qquad ((v_i,v_j)\in E) \tag{3.3}$$

$$x_{ij} \geq 1 \qquad ((v_i,v_j)\in A) \tag{3.4}$$

$$\sum_{v_k\in\Gamma^-(v_i)} n_{[kij]} - x_{ij} - y_{ij} = 0 \qquad (v_i \in V, v_j \in \Gamma^+(v_i)) \tag{3.5}$$

$$\sum_{v_k\in\Gamma^+(v_j)} n_{[ijk]} - x_{ij} - y_{ij} = 0 \qquad (v_i \in V, v_j \in \Gamma^+(v_i)) \tag{3.6}$$

$$x_{ij}, y_{ij}, y_{ji}, n_{[ijk]} \geq 0 \qquad ((v_i,v_j)\in A\cup E, [e_{ij}v_je_{jk}]\in T) \tag{3.7}$$

$$x_{ij}, y_{ij}, y_{ji}, n_{[ijk]} \text{ entiers} \qquad ((v_i,v_j)\in A\cup E, [e_{ij}v_je_{jk}]\in T) \tag{3.8}$$

Les contraintes (3.2) indiquent que chaque sommet est symétrique. Les contraintes (3.3) et (3.4) exigent que chaque lien soit traversé au moins une fois. Les contraintes (3.5) assurent que le nombre de virages réalisés à une intersection donnée et ayant le même lien terminal soit égal au nombre de liens traversés, incluant les copies, de l'intersection au sommet terminal du lien terminal. Les contraintes (3.6) garantissent que le nombre de virages réalisés à une intersection donnée et ayant le même lien initial soit égal au nombre de liens traversés, incluant les copies, du sommet initial du lien initial à l'intersection.

Lorsque les pénalités associées aux virages sont nulles, le PPCM-PV devient alors un problème du postier chinois mixte. Puisque le PPCM est NP-difficile

[Pap76] et qu'il est un cas particulier du PPCM-PV, le PPCM-PV est donc NP-difficile. Rappelons que Martínez et Soler ont montré que le problème du postier à un véhicule avec pénalités pour les virages (PPUV-PV) défini au début du chapitre 2 est NP-difficile [Mar98] même pour les cas particuliers suivants: lorsque $E = \varnothing$, $R = A$ et G est eulérien; lorsque $A = \varnothing$, $R = E$, G est eulérien et $p_{[ijk]} = p_{[kji]}$ pour tout virage $[e_{ij}v_je_{jk}]$ dans G.

Chapitre 4

Méthodes de résolution pour le PPCM-PV

Le problème du postier chinois dans un graphe mixte $G = (V, E \cup A)$ consiste à déterminer une tournée de longueur totale minimum et desservant les clients représentés par les liens du graphe. Plusieurs heuristiques et des méthodes exactes ont été développées pour résoudre ce problème. Le PPCM-PV (Problème du postier chinois mixte avec pénalités pour les virages) est un problème plus général que le PPCM. Ce problème consiste à servir un ensemble de clients représentés par les liens d'un réseau de façon à minimiser la longueur totale de la tournée ainsi que le nombre de virages d'un certain type dans la tournée accomplie. Au chapitre 2, nous avons vu que très peu de méthodes ont été développées pour tenter de le résoudre.

Dans ce chapitre, nous proposons trois nouvelles heuristiques pour résoudre le PPCM-PV. Ces heuristiques sont des adaptations de méthodes développées pour des PTA décrits au premier chapitre. La première heuristique est une méthode constructive et les deux autres sont des méthodes de recherche à voisinage variable.

4.1 Algorithme MIXTE

La première méthode consiste à transformer le graphe G en un graphe orienté pair et symétrique G' et à déterminer un circuit dans G' en tenant compte des pénalités

pour les virages. L'algorithme PPC_MIXTE1 présenté à la section 1.1.3 du premier chapitre est appliqué à G pour obtenir un graphe orienté pair et symétrique G'. Cet algorithme est utilisé pour sa simplicité car il produit un graphe symétrique et orienté contrairement à l'algorithme MIXED2 [Fre76] qui produit un graphe pair et équilibré mais pas nécessairement orienté nécessitant alors d'orienter certaines arêtes à l'aide de l'algorithme PPC_MIXTE_PAIR_ÉQUILIBRÉ ou de celui de Ford et Fulkerson [For62] (cf. section 1.1.3, chap.1).

Un circuit eulérien est déterminé dans G' en combinant les principes de l'algorithme END_PAIRING et de la stratégie 4 développée par Clossey pour le PPC-PV dans un graphe non orienté (cf. section 2.1.1, chap. 2). Lors de la construction d'un circuit, le prochain arc non couvert à introduire dans le circuit est celui qui minimise la somme des pénalités associées aux k virages qui seraient réalisés à partir du dernier arc introduit dans le circuit si k arcs non couverts étaient insérés dans le circuit. Il s'agit d'une généralisation de la stratégie 4 proposée par Clossey. En général, la valeur de k est petite. Plusieurs valeurs de k ont été testées. Les résultats sont présentés au chapitre 5.

Nous proposons également les deux stratégies suivantes afin de choisir le sommet de départ du deuxième circuit à l'étape 2 de l'algorithme END_PAIRING:

Stratégie 1. Choisir le prochain sommet selon l'ordre lexicographique.

Stratégie 2. Choisir le prochain sommet qui minimise la somme des pénalités associées aux k virages suivants.

Au prochain chapitre, des tests ont été effectués afin de déterminer la meilleure stratégie pour la sélection du sommet de départ du deuxième circuit.

L'algorithme MIXTE est décrit à la page suivante.

Algorithme MIXTE

Étape 1. *(Construction d'un graphe orienté pair et symétrique)*

Construire un graphe orienté pair et symétrique G' en appliquant l'algorithme PPC_MIXTE1 (cf. section 1.1.3, chap.1).

Étape 2. *(Construction d'un circuit eulérien dans G')*

Soit A' l'ensemble des arcs de G'.

a) Choisir un arc $(v_j, v_{j+1}) \in A'$ pour construire le premier circuit C_1.

b) Le prochain arc (v_{j+1}, v_{j+2}) à insérer est celui qui minimise la somme S des pénalités associées aux k virages suivants calculée comme suit:

$$S = p_{[j,j+1,j+2]} + \sum_{\varepsilon=1}^{k-1} p_{[j+\varepsilon,j+\varepsilon+1,j+\varepsilon+2]}$$

c) Continuer d'insérer des arcs de cette façon jusqu'à ce qu'un premier circuit C_1 soit construit. Si C_1 couvre chaque arc de A', STOP.

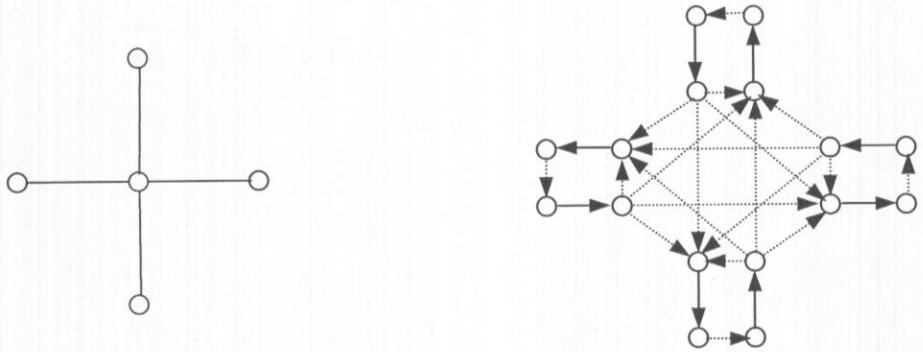
d) Choisir un sommet v_{j+1} appartenant à C_1 et incident à un arc qui ne figure pas dans C_1. À partir du sommet v_{j+1}, construire un deuxième circuit C_2 en utilisant la méthode décrite en b) et tel que C_1 et C_2 n'aient pas d'arc commun.

e) Fusionner les circuits C_1 et C_2 pour former le circuit C_1. Si C_1 couvre chaque arc de A', STOP. Sinon, retourner à d).

4.2 Algorithme RURAL

La deuxième méthode de résolution proposée pour le PPCM-PV comprend trois phases: une phase permettant de représenter graphiquement tous les virages possibles à une intersection donnée, une phase pour générer une solution initiale s_0 et une phase pour améliorer la solution initiale.

La première phase consiste à remplacer chaque intersection du graphe G par un sous-graphe afin de représenter graphiquement tous les virages permis au sommet (figure 4.1). Une pénalité est alors associée à chaque virage permis dans G. Cette phase transforme en fait le PPC-PV dans un graphe mixte $G = (V, E \cup A)$ en un problème du postier rural dans un graphe orienté $G' = (V', A')$.



(a) Intersection type (b) Sous-graphe

Figure 4.1: Modélisation graphique d'une intersection type

À chaque arête de G correspondent deux arcs de direction opposée dans G'. Étant donné que chaque arête doit être desservie au moins une fois, un seul des deux arcs relatifs à une arête doit être élément de l'ensemble R des arcs à desservir. Les arêtes de G sont donc orientées de façon arbitraire et l'arc associé à cette orientation dans G' est élément de R.

L'algorithme TRANSFORMER_PTA_EN_PPR décrit à la section 2.1.2 du chapitre 2 peut être appliqué à G pour effectuer la première phase de l'algorithme RURAL. La taille du graphe G' est cependant considérable. En effet, le nombre de sommets $|V'|$ dans le graphe G' est égal $2 \cdot |A| + 4 \cdot |E|$ et le nombre d'arcs $|A'|$ est égal $|A| + 2 \cdot |E| + n$ où n est le nombre de virages permis dans le graphe initial G.

La deuxième phase de l'algorithme RURAL consiste à résoudre le problème du postier rural défini dans le graphe orienté G' en appliquant les étapes 3, 4 et 5 de l'algorithme BALANCE_CONNECT de Ball et Magazine [Bal88] présenté à la section 1.2.2 du chapitre 1. Étant donné que le graphe G' est déjà composé uniquement des sommets appartenant à V_R, il n'est donc pas nécessaire d'effectuer l'étape 1 de l'algorithme BALANCE_CONNECT.

Dans la troisième phase de l'algorithme RURAL, nous avons développé une méthode de descente à voisinage variable pour améliorer la solution au PPR orienté obtenue lors de la seconde étape. La méthode améliorative fait appel aux procédures de post-optimisation de Hertz et al. [Her99] pour le PPR non orienté qui ont été modifiées par Mittaz [Mit99] pour le cas orienté (cf. section 1.2.2 du chapitre 1).

La méthode de descente à voisinage variable est une méta-heuristique qui contrôle et guide une heuristique interne de recherche locale pour lui permettre de franchir l'obstacle de l'optimalité locale. Tel que mentionné à la section 1.2.3 du premier chapitre, il est possible de rester bloqué en un minimum local qui peut être éloigné d'une solution optimale si une méthode de descente est utilisée comme méthode de recherche locale. La méthode de descente à voisinage variable permet d'éviter d'être piégé en un minimum local en explorant successivement plusieurs types de voisinages. Un minimum local pour un type de voisinage ne l'est pas nécessairement pour un autre. Ainsi, le risque d'être bloqué en un minimum local éloigné d'un optimum global peut être évité.

La méthode de descente à voisinage variable a été introduite par Hansen et Mladenović [Han97]. Soit S l'ensemble des solutions à un PPCM-PV. Un voisinage $N(s)$ d'une solution $s \in S$ est composé des solutions admissibles pouvant être obtenues en modifiant certaines caractéristiques de la solution s. Par exemple, une heuristique 2-opt pour le PVC remplace à chaque itération deux arêtes non adjacentes par deux arêtes différentes. Soit $N = \{N_1, N_1, \ldots, N_K\}$ un ensemble de différents types

de voisinage. Soit $N_k(s)$ le $k^{ème}$ type de voisinage de s ($k = 1, \ldots, K$). La méthode de descente à voisinage variable consiste à effectuer successivement une série de descentes en changeant de type de voisinage entre chaque application de la méthode descente.

Description générale d'une méthode de descente à voisinage variable

Étape 1. *(Initialisation)*

Sélectionner une structure de voisinages $\mathcal{N} = \{N_1, N_2, \ldots, N_K\}$ qui seront utilisés dans la descente.

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

Étape 2. Poser $k = 1$ et $s'' = s$.

Tant que $k \leq K$ faire

(Exploration de voisinage)

Déterminer $s' \in N_k(s)$ telle que $f(s') = \min f(s''')$ sous la contrainte que $s''' \in N_k(s)$.

(Déplacement ou non)

Si $f(s') < f(s)$, poser $s = s'$. Sinon, poser $k = k + 1$.

Étape 3. Si $f(s) < f(s'')$, retourner à l'étape 2. Sinon, s est la solution fournie par l'algorithme.

La structure de voisinages utilisés dans la descente comprend quatre types de voisinage. Le premier type de voisinage de s, $N_1(s)$, est obtenu en remplaçant un arc à desservir de R correspondant à une arête de G par l'arc correspondant à l'autre orientation de l'arête dans G. Le principe utilisé pour remplacer un tel arc de R consiste à utiliser la procédure DROP adaptée par Mittaz pour le PPR orienté (cf. section 1.2.2 du chapitre 1) afin de supprimer le service de cet arc dans la tournée. La procédure ADD_ORIENTÉ décrite à la section 1.2.2 du premier chapitre permet ensuite d'insérer dans T l'autre arc afin qu'il puisse être desservi.

Le second voisinage de s, $N_2(s)$, consiste à enlever et à rajouter successivement dans la tournée tous les arcs de R tout en raccourcissant la tournée après chacune de ces opérations. Il s'agit de la procédure de post-optimisation DROP_ADD proposée par Hertz et al. [Her99] qui a été adaptée par Mittaz [Mit99] pour le PPR orienté (cf. section 1.2.2).

Les deux derniers voisinages sont basés sur la méthode 2-OPT de Hertz et al. [Her99] décrite à la section 1.2.2 du premier chapitre. Le voisinage $N_3(s)$ est obtenu en retirant trois arcs non adjacents qui sont remplacés par trois plus courtes chaînes. La procédure RACCOR (cf. section 1.2.2 du chapitre 1) est ensuite appliquée à la tournée résultante. Si l'un ou l'autre des trois arcs retirés sont des éléments de R, la tournée n'est plus admissible. La procédure ADD_ORIENTÉ est alors appliquée à chacun de ces arcs. L'algorithme 3-OPT présente de manière détaillée le passage d'une solution s à une solution $s' \in N_3(s)$.

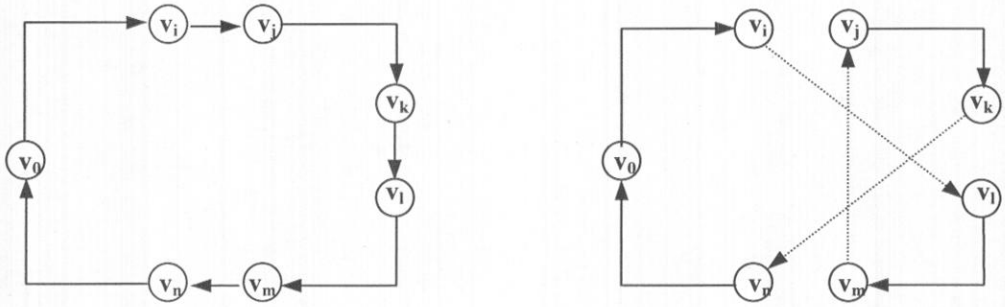
Algorithme 3-OPT

Étape 1. Choisir trois arcs (v_i, v_j), (v_k, v_l) et (v_m, v_n) dans la tournée T.
Remplacer (v_i, v_j), (v_k, v_l) et (v_m, v_n) par les plus courtes chaînes SC_{il}, SC_{kn} et SC_{mj} dans G'.
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 arcs de $R(T_1)$ de même longueur ou plus courte que T_1 en appliquant RACCOR à T_1.

Étape 2. Si l'un ou l'autre des (v_i, v_j), (v_k, v_l), (v_m, v_n) sont des arcs de $R(T)$ non couverts par T_2, construire une tournée T_3 desservant les clients de $R(T)$ en utilisant ADD_ORIENTÉ. Si T_3 est plus courte que T, poser $T = T_3$.

La figure 4.2 illustre le remplacement de trois arcs (v_i, v_j), (v_k, v_l) et (v_m, v_n) par les plus courtes chaînes SC_{il}, SC_{kn} et SC_{mj} à l'étape 1 de l'algorithme 3-OPT. Le sommet

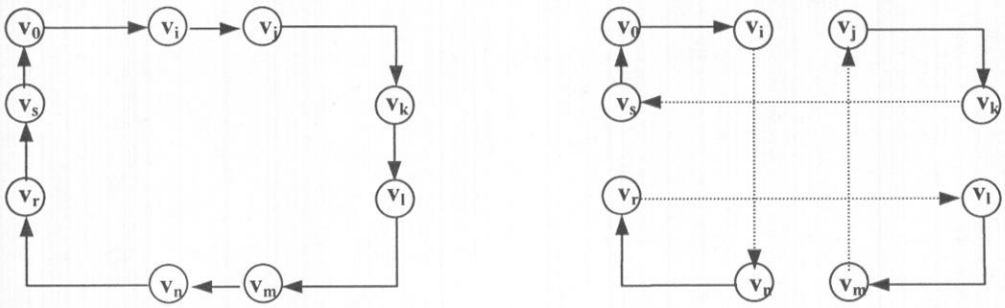
v_0 correspond au point de départ et d'arrivée de la tournée. Les plus courtes chaînes sont représentées en pointillés à la figure 4.2 (b).



(a) Tournée initiale (b) Nouvelle tournée

Figure 4.2: Remplacement de trois arcs par trois plus courtes chaînes

Le voisinage $N_4(s)$ est obtenu de façon analogue sauf que le nombre d'arcs non adjacents à retirer est de quatre. La figure 4.3 illustre le remplacement de quatre arcs (v_i, v_j), (v_k, v_l), (v_m, v_n) et (v_r, v_s) par les plus courtes chaînes SC_{in}, SC_{ks}, SC_{mj} et SC_{rl}.



(a) Tournée initiale (b) Nouvelle tournée

Figure 4.3: Remplacement de quatre arcs par quatre plus courtes chaînes

Étant donné que la taille des voisinages $N_3(s)$ et $N_4(s)$ d'une solution s peut être très importante, seule une portion des voisinages est explorée. La taille des portions N_3' et N_4' des voisinages $N_3(s)$ et $N_4(s)$ est déterminée au chapitre 5.

Les voisinages $N_1(s)$, $N_2(s)$, $N_3(s)$ et $N_4(s)$ sont numérotés du plus étroit au plus large. Étroit dans le sens où un voisinage ne modifie que très légèrement la solution courante. Large dans le sens où une solution voisine peut être complètement différente de la solution courante. L'ordre dans lequel les quatre voisinages sont considérés peut influencer la qualité des solutions. Une bonne règle empirique consiste à ranger les voisinages par ordre de taille non-décroissante [Han97]. Les résultats de tests numériques effectués en ordonnant les voisinages de trois manières différentes sont présentés au chapitre 5.

L'algorithme DVV résume la troisième phase de l'algorithme RURAL.

Algorithme DVV

Étape 1. *(Initialisation)*

Soit s_0 la solution initiale au PPR orienté.

Poser $s = s_0$.

Générer un ordre de voisinages: N_{O1}, N_{O2}, N_{O3} et N_{O4} où O1, O2, O3 et O4 sont respectivement les numéros des voisinages dans l'ordre établi.

Étape 2. Poser $k = 1$ et $s'' = s$.

Tant que $k \leq 4$ faire

(Exploration de voisinage)

Déterminer $s' \in N_{Ok}'(s)$ telle que $f(s') = \min f(s''')$ sous la contrainte que $s''' \in N_{Ok}'(s)$.

(Déplacement ou non)

Si $f(s') < f(s)$, poser $s = s'$. Sinon, poser $k = k + 1$.

Étape 3. Si $f(s) < f(s'')$, retourner à l'étape 2. Sinon, s est la solution fournie par l'algorithme.

Trois variantes de l'algorithme DVV seront testées au chapitre 4. Une première variante consiste à supprimer l'étape 3 de la DVV. La seconde variante est une stratégie de *First Improvement* qui revient à passer au voisinage suivant aussitôt que la première amélioration de la solution s est obtenue lors de la descente à l'étape 2. Hansen et Mladenović [Han99] ont montré que lors de l'application de l'heuristique 2-opt au PVC, les sélections de la meilleure amélioration à chaque itération donnent de moins bons résultats, en moyenne, que la sélection de la première amélioration. La dernière variante de la DVV consiste à réamorcer la DVV avec le voisinage $N_1(s)$ aussitôt qu'une meilleure solution est trouvée lors de la descente avec un voisinage donné plutôt que de poursuivre l'exploration des voisinages en séquence [Han00].

L'algorithme RURAL peut maintenant être décrit.

Algorithme RURAL

Étape 1. *(Orientation des arêtes)*
Orienter de façon arbitraire les arêtes de G.

Étape 2. *(Transformation du PPC en un PPR)*
Construire un graphe orienté $G' = (V', A')$ en appliquant l'algorithme TRANSFORMER_PTA_EN_PPR au graphe G (cf. section 1.2.1.2, chap.1). Les arcs de A' correspondant aux arcs et arêtes orientées à desservir de G sont des éléments de R.

Étape 3. *(Construction de la solution initiale)*
Appliquer les étapes 3, 4 et 5 de l'algorithme BALANCE_CONNECT à G' (cf. section 1.2.2, chap. 1) pour obtenir une tournée initiale s_0.

Étape 4. *(Descente à voisinage variable)*
Appliquer à s_0 l'algorithme DVV décrit dans cette section.

Étant donné que la première phase de l'algorithme transforme un PTA avec pénalités pour les virages en un PPR orienté, l'algorithme RURAL permet de

résoudre tous les problèmes du postier chinois et rural avec pénalités pour les virages. Cependant, ce type de transformation augmente considérablement la taille du graphe initial G.

4.3 Algorithme AFFECTATION

La méthode AFFECTATION est une adaptation au cas mixte de l'algorithme CONS2 développé par Benavent et Soler [Ben99] pour le problème du postier rural orienté avec pénalités pour les virages. La méthode consiste premièrement à affecter chaque lien du graphe G à un autre lien en résolvant un problème d'affectation dans G. Le coût d'affecter un lien (v_i, v_j) à un autre lien (v_k, v_l) est égal à la longueur du plus court chemin avec pénalités pour les virages PCC_{ijkl} reliant le lien (v_i, v_j) et le lien (v_k, v_l) dans G. Soit $suc(v_i)$ le successeur du sommet v_i de G et $pre(v_i)$ le prédécesseur du sommet v_i de G. La longueur $L(PCC_{ijkl})$ du plus court chemin avec pénalités pour les virages entre deux liens (v_i, v_j), (v_k, v_l) de G est définie comme suit:

$$L(PCC_{ijkl}) = c(v_i, v_j) + p_{[ijsuc(j)]} + L(PCC_{jk}) + p_{[pre(k)kl]}$$

Le coût du dernier arc (v_k, v_l) n'est pas considéré dans le calcul de la longueur L. Le calcul de la longueur du plus court chemin PCC_{jk} du sommet v_j au sommet v_k dans G tient compte des pénalités pour les virages. Toutes les pénalités associées aux virages du plus court chemin PCC_{ijkl} sont aussi considérées dans le calcul de la longueur du plus court chemin.

Le calcul de la longueur du plus court chemin avec pénalités pour les virages entre deux liens (v_i, v_j), (v_k, v_l) de G dépend aussi du type des liens impliqués dans l'affectation. Par exemple, la longueur du plus court chemin reliant un arc (v_i, v_j) et une arête (v_k, v_l) dans G correspond au $\min\{L(PCC_{ijkl}), L(PCC_{ijlk})\}$.

La solution au problème d'affectation permet dans un deuxième temps de construire un ensemble de circuits traversant au moins une fois chaque lien de G. Cependant, il est possible que certains circuits ne puissent pas être formés si les deux orientations d'une même arête sont impliquées dans la solution au problème d'affectation. À titre d'exemple, considérons les deux affectations représentées à la figure 4.4. Les plus courts chemins avec pénalités pour les virages sont dessinés en pointillés. L'arête (v_i, v_j) est affectée à un arc (v_k, v_l) et un arc (v_m, v_n) est affecté à l'autre orientation (v_j, v_i) de la même arête. Le calcul du minimum entre $L(PCC_{mnij})$ et $L(PCC_{jikl})$ permet alors de ne considérer qu'une seule orientation de l'arête (v_i, v_j) dans les deux affectations. Si le minimum est égal à $L(PCC_{mnij})$, alors l'arc (v_m, v_n) est affecté à l'arête orientée (v_i, v_j) et l'affectation de l'arête (v_i, v_j) à l'arc (v_k, v_l) demeure inchangée. Si le minimum est égal à $L(PCC_{jikl})$, alors l'arête orientée (v_j, v_i) est affectée à l'arc (v_k, v_l) et l'affectation de l'arc (v_m, v_n) à l'arête orientée (v_j, v_i) reste la même.

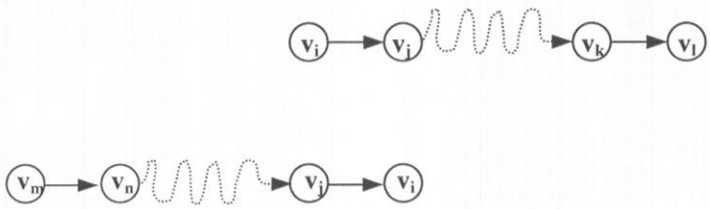


Figure 4.4: Exemple de deux affectations impliquant la même arête

Une méthode de descente à voisinage variable est ensuite appliquée à la solution au problème d'affectation. Étant donné un nombre k fixé ($2 \leq k \leq |A|+|E|$), le $k^{\text{ème}}$ type de voisinage de s, $N_k(s)$, est obtenu en permutant k affectations de s. Soit les deux affectations suivantes ($k = 2$): le lien (v_i, v_j) est affecté au lien (v_k, v_l) et le lien (v_m, v_n) est affecté au lien (v_p, v_q). Permuter les deux affectations revient à affecter le lien (v_i, v_j) au lien (v_p, v_q) et à affecter le lien (v_m, v_n) au lien (v_k, v_l). Un autre type de voisinage, appelé N_1, est aussi utilisé. Ce voisinage consiste simplement à inverser l'orientation d'une arête (v_i, v_j) de E comme le voisinage N_1 dans la DVV de

l'algorithme RURAL à la section 4.1. La taille du voisinage $N_k(s)$ peut cependant être très importante. Le nombre d'ensembles différents de k affectations est égal à $(|A| + |E|)! / (|A| + |E| - k)!\ k!$. Pour un ensemble donné de k affectations, il y a $k!$ façons différentes de les affecter. La taille du voisinage $N_k(s)$ est donc de l'ordre de $(|A|+|E|)^k$. Par conséquent, seule une portion N_k' du voisinage $N_k(s)$ est explorée. Pour une valeur de k fixée, m ensembles de k affectations sont générés aléatoirement. Pour un ensemble donné de k affectations, n façons différentes d'affecter les liens entre eux sont générées aléatoirement. De cette façon, $m \cdot n$ solutions sont générées à chaque itération de la méthode de descente. Les valeurs de m et de n sont fixées au chapitre suivant. Les voisinages N_1 et N_k ($2 \leq k \leq |A|+|E|$) peuvent être ordonnés de différentes manières. Trois ordres seront testés au chapitre 5.

L'algorithme DVV2 décrit la méthode de descente à voisinage variable utilisée.

Algorithme DVV2

Étape 1. *(Initialisation)*

Soit s_0 la solution initiale au problème d'affectation. Poser $s = s_0$.

Générer un ordre de voisinages: N_{O1}, N_{O2}, …, $N_{O|A|+|E|}$ où O1, O2, …, O$|A|+|E|$ sont respectivement les numéros des voisinages dans l'ordre établi.

Étape 2. Poser $k = 1$ et $s'' = s$.

Tant que $k \leq |A|+|E|$ faire

(Exploration de voisinage)

Déterminer $s' \in N_{Ok}'(s)$ telle que $f(s') = \min f(s''')$ sous la contrainte que $s''' \in N_{Ok}'(s)$.

(Déplacement ou non)

Si $f(s') < f(s)$, poser $s = s'$. Sinon, poser $k = k + 1$.

Étape 3. Si $f(s) < f(s'')$, retourner à l'étape 2. Sinon, s est la solution fournie par l'algorithme.

La dernière étape de la méthode AFFECTATION consiste à combiner les circuits formés à la deuxième étape à l'aide de deux stratégies proposées par McBride [McB82] et présentées à la section 2.2.2 du chapitre 2.

L'algorithme AFFECTATION peut maintenant être résumé.

Algorithme AFFECTATION

Étape 1. *(Construction des circuits)*

Résoudre le problème d'affectation dans G de façon à affecter chaque lien de $E \cup A$ à un autre lien de $E \cup A$. Soit $L(PCC_{ijkl})$ la longueur du plus court chemin avec pénalités pour les virages et sans virage interdit entre deux liens (v_i, v_j) et (v_k, v_l). Soit $suc(v_i)$ le successeur du sommet v_i dans G et soit $pre(v_i)$ le prédécesseur du sommet v_i dans G. La longueur $L(PCC_{ijkl})$ est égale à la somme $c(v_i, v_j) + p_{[ijsuc(j)]} + L(PCC_{jk}) + p_{[pre(k)kl]}$. Le coût d'affecter un lien (v_i, v_j) à un lien (v_k, v_l) est défini comme suit:

$$L(PCC_{ijkl}) = \begin{cases} L(PCC_{ijkl}) & \text{si } (v_i, v_j), (v_k, v_l) \in A \\ \min\{L(PCC_{ijkl}), L(PCC_{ijlk})\} & \text{si } (v_i, v_j) \in A \text{ et } (v_k, v_l) \in E \\ \min\{L(PCC_{ijkl}), L(PCC_{jikl})\} & \text{si } (v_i, v_j) \in E \text{ et } (v_k, v_l) \in A \\ \min\{L(PCC_{ijkl}), L(PCC_{ijlk}), L(PCC_{jikl}), L(PCC_{jilk})\} & \\ \qquad \text{si } (v_i, v_j), (v_k, v_l) \in E & \\ \infty \qquad \text{si } v_i = v_k \text{ et } v_j = v_l & \end{cases}$$

Étape 2. Si une arête (v_i, v_j) est affectée à un lien (v_k, v_l) et qu'un lien (v_m, v_n) est affecté à l'autre orientation (v_j, v_i) de la même arête dans la solution optimale du problème d'affectation, calculer la valeur $p = \min\{L(PCC_{mnij}), L(PCC_{jikl})\}$.

Si $p = L(PCC_{mnij})$, alors affecter le lien (v_m, v_n) à l'arête (v_i, v_j).

Si $p = L(PCC_{jikl})$, alors affecter l'arête orientée (v_j, v_i) au lien (v_k, v_l).

Soit s_0 la solution au problème d'affectation.

Algorithme AFFECTATION (suite)

Étape 3. Appliquer l'algorithme DVV2 à la solution s_0.

Étape 4. Construire le graphe orienté $G' = (V, A \cup A')$ en orientant les arêtes de G d'après la solution au problème d'affectation de l'étape 3.

Soit A'', un ensemble d'arcs. Poser $A'' = \varnothing$. Répéter les étapes suivantes jusqu'à ce que tous les arcs de $A \cup A'$ appartiennent à A''.

a) Choisir un arc $a = (v_i, v_j)$ tel que $(v_i, v_j) \in A \cup A'$ et $(v_i, v_j) \notin A''$.
Poser $a_i = a$. Poser $i = 0$.

b) Soit $q(a_i)$, l'arc auquel l'arc a_i de G' est affecté dans la solution au problème d'affectation à l'étape 3. Ajouter à A'' les arcs du plus court chemin entre l'arc a_i et l'arc $q(a_i)$ déterminé à l'étape 1. L'arc $q(a_i)$ n'est pas ajouté à A''. Poser $a_{i+1} = q(a_i)$. Poser $i = i + 1$.

c) Si $a_i = a_0$, un circuit a été construit.
Sinon, retourner à b).

d) Si tous les arcs de $A \cup A'$ appartiennent à A'', STOP. Sinon, retourner à a).

Étape 5. *(Combinaison des circuits ayant un arc en commun)*

Si un seul circuit a été formé à l'étape 4, STOP.

Soient $C_1, C_2, \ldots, C_c$, les différents circuits formés à l'étape 4. Combiner les circuits ayant un arc commun. Si un seul circuit est formé, STOP.

Étape 6. *(Combinaison des circuits ayant un sommet en commun)*

Soient $C_1, C_2, \ldots, C_c$, les différents circuits formés.

Construire le graphe non orienté $H = (W, B)$ tel qu'à chaque circuit C_i correspond un sommet w_i dans W et tel qu'une arête $(w_i, w_j) \in B$ si et seulement si les circuits C_i et C_j ont au moins un sommet v en commun dans G. Le coût c_{ij} associé à l'arête (w_i, w_j) est calculé comme suit:

$c_{ij} = \min\{p_{[asuc(b)]} + p_{[bsuc(a)]} - p_{[asuc(a)]} - p_{[bsuc(b)]} : a \in C_i, b \in C_j\}$ où a et b sont des arcs incidents au sommet v et $suc(a)$ et $suc(b)$ représentent respectivement les successeurs des arcs a et b dans les circuits C_i et C_j.

Algorithme AFFECTATION (suite)

Étape 7. Déterminer un arbre de recouvrement de coût minimum dans H.
Pour chaque arête (w_i, w_j) appartenant à l'arbre optimal, identifier les arcs $a \in C_i$, $b \in C_j$ pour lesquels le minimum est atteint lors du calcul de c_{ij}. Si les circuits C_i et C_j n'ont pas encore été reliés, remplacer les virages $[a, s(a)]$ et $[b, s(b)]$ par les virages $[a, s(b)]$ et $[b, s(a)]$ afin de combiner les circuits C_i et C_j. Mettre à jour $s(a)$ et $s(b)$.

Étape 8. Si le nombre de circuits est égal à 1, STOP. Sinon, retourner à l'étape 6.

Trois variantes de la DVV2 seront testées au chapitre 5. Ce sont les mêmes variantes décrites à la section 4.2 pour la DVV de l'algorithme RURAL.

Chapitre 5

Résultats numériques

Dans ce chapitre sont présentés les résultats obtenus à l'aide des trois méthodes décrites au chapitre précédent. Ce chapitre comprend quatre sections. Une borne inférieure pour le PPCM-PV est proposée dans la première section. Dans la deuxième section, les caractéristiques des problèmes utilisés pour tester les trois méthodes sont détaillées. Une troisième section est consacrée à la présentation des résultats issus de certains tests effectués sur les trois algorithmes MIXTE, RURAL et AFFECTATION. À la dernière section, ces algorithmes sont comparés aux algorithmes proposés par Corberán et al. [Cor00b] pour la résolution du PPRM-PV.

5.1 Une borne inférieure pour le PPCM-PV

Le problème du postier chinois avec pénalités pour les virages est un problème NP-difficile [Mar98]. Afin d'évaluer la qualité des trois heuristiques proposées au chapitre précédent, il est utile de disposer d'une borne inférieure. Benavent et Soler [Ben99] ont développé une borne inférieure pour le problème du postier rural dans un graphe orienté avec pénalités pour les virages et sans virage interdit (PPRO-PV). Cette borne est obtenue en résolvant un problème d'affectation de façon à affecter chaque arc à desservir de R à un autre arc à desservir. Le coût d'affecter un arc (v_i, v_j) de R à un autre arc (v_k, v_l) de R est égal à la longueur du plus court chemin avec pénalités pour les virages et sans virage interdit PCC_{ijkl} entre l'arc (v_i, v_j) et l'arc (v_k, v_l) dans G.

Nous allons nous baser sur la borne inférieure décrite pour le PPRO-PV dans [Ben99] afin de développer une borne inférieure pour le PPCM-PV. La solution au problème d'affectation obtenue après la première étape de l'algorithme AFFECTATION présenté à la section 4.2 du chapitre précédent correspond en fait à notre adaptation de la borne inférieure proposée par Benavent et Soler pour le PPRO-PV au cas mixte. Il s'agit de résoudre le problème d'affectation dans le graphe mixte G de façon à affecter chaque lien de $E \cup A$ à un autre lien de $E \cup A$. Le coût d'affecter un lien (v_i, v_j) à un lien (v_k, v_l) correspond à la longueur $L(PCC_{ijkl})$ du plus court chemin avec pénalités pour les virages et sans virage interdit entre (v_i, v_j) et (v_k, v_l) définie comme suit:

$$L(PCC_{ijkl}) = \begin{cases} L(PCC_{ijkl}) & \text{si } (v_i, v_j), (v_k, v_l) \in A \\ \min\{L(PCC_{ijkl}), L(PCC_{ijlk})\} & \text{si } (v_i, v_j) \in A \text{ et } (v_k, v_l) \in E \\ \min\{L(PCC_{ijkl}), L(PCC_{jikl})\} & \text{si } (v_i, v_j) \in E \text{ et } (v_k, v_l) \in A \\ \min\{L(PCC_{ijkl}), L(PCC_{ijlk}), L(PCC_{jikl}), L(PCC_{jilk})\} & \text{si } (v_i, v_j), (v_k, v_l) \in E \\ \infty & \text{si } v_i = v_k \text{ et } v_j = v_l \end{cases}$$

où

$$\begin{aligned} L(PCC_{ijkl}) &= c_{ij} + p_{[ijsuc(j)]} + L(PCC_{jk}) + p_{[pre(k)kl]}, \\ suc(v_i) &= \text{successeur du sommet } v_i \text{ dans } G, \\ pre(v_i) &= \text{prédécesseur du sommet } v_i \text{ dans } G. \end{aligned}$$

Toute solution au problème d'affectation peut s'interpréter comme un ensemble de circuits recouvrant tous les arcs et toutes les arêtes de G au moins une fois. En effet, considérons le lien (v_k, v_l), affecté au lien (v_i, v_j), comme étant le successeur du lien (v_i, v_j). En suivant récursivement chaque lien par son successeur dans la solution au problème d'affectation, un ensemble de circuits recouvrant tous les liens de G au moins une fois est construit.

Toute solution au problème d'affectation ne représente pas nécessairement une solution admissible au PPCM-PV. Certaines arêtes peuvent servir de lien prédécesseur dans une direction, et de lien successeur dans l'autre direction. De plus, des circuits peuvent être formés.

5.2 Problèmes tests

Deux ensembles de problèmes tests sont utilisés. Le premier groupe est un ensemble de problèmes tirés de la littérature et le second est composé de nouveaux problèmes générés aléatoirement.

Problèmes de Corberán, Martí, Martínez et Soler

Le premier groupe de problèmes tests correspond aux instances générées par Corberán, Martí, Martínez et Soler [Cor00b] pour le problème du postier rural mixte avec pénalités pour les virages et sans virage interdit. Dans certaines des 279 instances générées, l'ensemble des liens à desservir est égal à l'ensemble des arêtes et des arcs (i.e. $R = E \cup A$). Il s'agit de 93 instances possédant entre 40 et 200 sommets, le nombre d'arcs varie entre 90 et 440 et le nombre d'arêtes varie entre 10 et 100.

Problèmes générés aléatoirement

Contrairement aux problèmes de Corberán, Martí, Martínez et Soler [Cor00b], tous les graphes que nous avons générés sont rectilinéaires. Ces instances représentent les réseaux des régions urbaines Nord-Américaines dont les routes sont généralement parallèles et perpendiculaires.

Les nouvelles instances ont été générées à l'aide de la méthode décrite à la page suivante.

Description de la méthode utilisée pour générer les instances

Étape 1. Poser $E = \varnothing$ et $A = \varnothing$.

Étape 2. Générer une grille comprenant $n = p \times q$ intersections où p est le nombre de colonnes et q est le nombre de lignes dans la grille. Chaque intersection dans la grille représente un sommet du graphe mixte G. Soit l'ensemble des sommets $V = \{v_{ij} = (i, j) : i = 0, 1,..., p\text{-}1 \text{ et } j = 0, 1,..., q\text{-}1\}$ où (i, j) est la coordonnée du sommet v_{ij}. Choisir aléatoirement le sommet de départ et d'arrivée v_0.

Étape 3. Insérer aléatoirement dans la grille un arbre d'arêtes pour s'assurer que le graphe G soit fortement connexe. Ajouter à E les arêtes de l'arbre et retirer de V les sommets qui représentent l'intersection d'une seule traverse.

Étape 4. Générer aléatoirement selon la loi de Bernoulli de paramètre π_1 un lien entre deux sommets sans qu'il n'y ait jamais plus d'un lien pour les relier de façon à construire l'ensemble $L = \{(v_{ij}, v_{kl}) : i = k \text{ et } |l - j| = 1 \text{ ou } j = l \text{ et } |i - k| = 1\}$. Générer aléatoirement selon la loi de Bernoulli de paramètre π_2 le type de lien de L. Si le lien est un arc, déterminer aléatoirement selon la loi de Bernoulli de paramètre $\pi_3 = 0{,}5$ la direction de l'arc (arc (v_i, v_j) ou arc (v_j, v_i)) et ajouter cet arc à A. Si le lien est une arête, ajouter à E l'arête générée.

Étape 5. Déterminer aléatoirement selon la loi uniforme discrète de paramètres 1 et 100 la longueur d'un arc ou d'une arête.

Étape 6. Associer une pénalité à chaque virage de G selon la structure suivante des pénalités pour les virages:

traverse:	0
droite:	1
gauche:	5
en «U»:	11

La structure des pénalités pour les virages est déterminée d'après les structures les plus utilisées dans la littérature pour la collecte des ordures et les tournées de balai mécanique [McB82].

Huit familles de graphes sont générés selon les combinaisons de valeurs p et q du tableau 5.1.

n	20	30	40	50	75	100	150	200
$p \times q$	4×5	5×6	5×8	5×10	5×15	10×10	10×15	10×20

Tableau 5.1: Familles de graphes générés

Pour chaque famille de graphe, 10 graphes sont générés avec un nombre croissant de liens et un nombre croissant d'arcs. La probabilité qu'un lien soit généré entre deux sommets est égale à π_1 et la probabilité qu'un lien généré soit un arc est égale à π_2.

No	1	2	3	4	5	6	7	8	9	10
π_1	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95	1
π_2	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95	1

Tableau 5.2: Probabilités π_1 et π_2

Le nombre de sommets des 80 instances varie entre 16 et 267, le nombre d'arêtes varie entre 15 et 314 et le nombre d'arcs varie entre 3 et 161. Le générateur de nombres pseudo-aléatoires est initialisé avec une semence différente pour chaque instance. Pour une instance donnée, la semence est égale à $\pi_1 pq + \pi_2 pq + k$ où $k = 1, 2, \ldots, 80$ représente le numéro de l'instance à générer.

5.3 Tests numériques

L'objectif des tests qui suivent est d'ajuster les paramètres qui contrôlent les algorithmes MIXTE, RURAL et AFFECTATION et d'identifier la meilleure variante de la DVV pour les algorithmes RURAL et AFFECTATION selon la qualité des solutions obtenues et le temps de calcul. Les critères utilisés afin d'identifier la meilleure variante d'une DVV sont la restriction de la taille des voisinages ainsi que leur influence sur les performances d'un algorithme. La taille des voisinages peut être limitée en fixant les valeurs des paramètres qui l'influencent. Pour tester l'impact d'un voisinage sur les performances d'une méthode, une variante utilisant le voisinage et une autre variante ne faisant pas appel à ce voisinage sont traitées.

5.3.1 Algorithme RURAL

L'influence des voisinages N_2, N_3 et N_4 sur les performances de l'algorithme RURAL a été testée en traitant 40 nouvelles instances avec trois variantes de l'algorithme RURAL où l'étape 4 est modifiée. La première variante utilise uniquement le voisinage N_1 lors de l'application de la DVV, la deuxième considère les voisinages N_1 et N_2 et la dernière alterne entre les quatre voisinages N_1, N_2, N_3 et N_4. Pour les voisinages N_3 et N_4, le nombre r d'ensembles différents de 3 ou 4 arcs à retirer est de un. Les caractéristiques des 40 nouvelles instances avec $p = q = 5$ apparaissent au tableau 4.3 de la page suivante. Étant donné que la première étape de l'algorithme RURAL augmente considérablement la taille d'un graphe, seules des instances de petite taille ont été testées.

À toute solution admissible s du PPCM-PV dans $G = (V, E \cup A)$, nous pouvons associer un graphe augmenté $G_a = (V, E \cup A \cup E_a \cup A_a)$. L'ensemble E_a 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. De même, A_a contient autant de copies de l'arc (v_i, v_j) de A que le nombre de fois où (v_i, v_j) est traversé dans s sans être desservi. Soit $c(E \cup A)$ le

coût total associé aux liens de $E \cup A$ et $c(E_a \cup A_a)$ le coût total des liens de $E_a \cup A_a$. Notons $c(T)$ la somme des pénalités associées aux virages de s où T est l'ensemble des virages de s.

π_1	π_2	*No*	$\|V\|$	$\|E\|$	$\|A\|$	π_1	π_2	*No*	$\|V\|$	$\|E\|$	$\|A\|$
		1	15	15	2			21	18	19	5
		2	18	18	1			22	17	17	6
		3	12	11	2			23	17	20	4
		4	15	16	0			24	16	19	3
0,55	0,55	5	18	18	5	1	0,55	25	16	18	4
		6	17	17	6			26	16	18	2
		7	17	18	1			27	20	24	5
		8	19	21	4			28	16	17	5
		9	16	17	3			29	18	18	7
		10	19	19	3			30	17	18	5
		11	17	16	5			31	15	14	5
		12	17	16	4			32	15	14	6
		13	17	16	3			33	21	20	12
		14	18	17	5			34	17	16	8
0,55	1	15	19	18	7	1	1	35	20	19	10
		16	20	19	2			36	21	20	11
		17	20	19	2			37	20	19	9
		18	14	13	1			38	20	19	10
		19	20	19	6			39	16	15	7
		20	22	21	9			40	22	21	12

Tableau 5.3: Caractéristiques des 40 nouvelles instances pour l'algorithme RURAL

Dans les différents tableaux de résultats de ce chapitre, $E1_{moyen}$ représente l'écart moyen par rapport à la borne inférieure $BI1$ définie à la section 5.1 et $ET1_{moyen}$ représente l'écart-type moyen des écarts relatifs par rapport à $BI1$. Les écarts relatifs $E1$ sont calculés selon la formule:

$$E1 = \frac{\left(c(E \cup A) + c(E_a \cup A_a) + c(T) - BI1\right)}{BI1}$$

$E2_{moyen}$ représente l'écart moyen par rapport à la borne inférieure $BI2$ obtenue en soustrayant de $BI1$ le coût total $c(E \cup A)$ des liens de $E \cup A$. Ainsi, les écarts relatifs $BI2$ sont calculés selon la formule:

$$E2 = \frac{\left(c(E_a \cup A_a) + c(T) - BI2\right)}{BI2}$$

$ET2_{moyen}$ représente l'écart-type moyen des écarts relatifs par rapport à $BI2$. T_{moyen} indique le temps nécessaire, en moyenne, pour trouver la solution fournie par l'algorithme et TT_{moyen} indique le temps total moyen d'exécution. Tous les algorithmes ont été programmés en langage C et les tests ont été effectués sur un ordinateur Pentium de 200 MHz de fréquence ayant 64 Moctets de mémoire vive. Les temps de calcul sont exprimés en secondes CPU.

Le tableau 5.4 montre que l'utilisation des voisinages N_3 et N_4 permet d'améliorer la qualité des solutions obtenues. Les meilleurs écarts relatifs moyens sont indiqués en gras. Les temps totaux moyens de calcul augmentent cependant de 13% lorsque les voisinages N_3 et N_4 sont utilisés. On remarque que la meilleure solution est trouvée au 7/10 de la recherche environ peu importe les voisinages utilisés.

	Algorithme RURAL		
	avec N_1	avec N_1, N_2	avec N_1, N_2, N_3 et N_4 ($r = 1$)
$E1_{moyen}$	0,373	0,373	**0,345**
$ET1_{moyen}$	0,179	0,179	0,169
$E2_{moyen}$	6,964	6,964	**6,667**
$ET2_{moyen}$	7,953	7,953	7,747
T_{moyen}	39,38	36,67	42,15
TT_{moyen}	58,27	54,32	62,54

Tableau 5.4: Influence des voisinages N_2, N_3 et N_4 sur le comportement de RURAL

Le tableau 5.4 montre également que le voisinage N_2 n'a aucune influence sur la qualité des solutions avec la première variante. Le voisinage N_2 consiste à appliquer la procédure de post-optimisation DROP_ADD. Tous les arcs de R sont enlevés et rajoutés successivement dans la tournée qui est raccourcie après chacune de ces opérations. Le graphe orienté G' obtenu après la transformation de l'étape 1 possède une structure assez particulière. En effet, il n'y a pas d'arcs adjacents appartenant à R ni d'arcs adjacents n'appartenant pas à R dans G'. Autrement dit, l'ensemble R des arcs à desservir dans G' comprend $|A| + 2 \cdot |E|$ composantes connexes chacune avec un seul arc. Étant donné qu'un chemin inutile dans une tournée comprend un seul arc (cet arc correspond à un virage possible dans le graphe initial G), la procédure RACCOR parviendra rarement à réduire la longueur d'une tournée en raccourcissant un chemin inutile. Cela peut expliquer la similitude des résultats obtenus avec les deux premières variantes.

Les voisinages N_3 et N_4 sont des voisinages larges qui peuvent modifier une tournée de manière importante. Ainsi, ces voisinages permettent d'explorer des régions éloignées de l'espace des solutions que les voisinages N_1 et N_2 ne peuvent pas atteindre.

Nous avons testé une autre variante en considérant les voisinages N_1, N_3 et N_4 avec $r = 1$ où r est le nombre d'ensembles différents de 3 ou 4 arcs à retirer. Les résultats démontrent que des solutions de moins bonne qualité sont obtenues. Pour la suite de l'analyse, les voisinages N_1, N_2, N_3 et N_4 sont tous explorés à l'étape 4 de l'algorithme RURAL

Nous avons également testé l'influence de la taille des voisinages N_3 et N_4 sur les performances de l'algorithme RURAL en faisant varier le paramètre r. Des valeurs de r supérieures à 100 conduiraient à des temps de calcul jugés trop importants pour des graphes de petite taille.

	Algorithme RURAL					
	$r = 1$	$r = 5$	$r = 10$	$r = 20$	$r = 50$	$r = 100$
$E1_{moyen}$	0,345	0,351	0,370	0,358	0,376	**0,337**
$ET1_{moyen}$	0,169	0,186	0,213	0,191	0,200	0,202
$E2_{moyen}$	6,667	6,585	7,020	7,099	6,766	**6,294**
$ET2_{moyen}$	7,747	7,428	7,839	8,309	7,322	7,014
T_{moyen}	42,15	55,73	62,49	88,58	99,37	258,33
TT_{moyen}	62,54	83,61	108,38	153,19	259,27	483,25

Tableau 5.5: Influence de la taille des voisinages sur le comportement de RURAL

Le tableau 5.5 montre que la qualité des solutions obtenues peut être légèrement améliorée avec des valeurs de r supérieures à 1. En considérant 100 ensembles différents de 3 ou 4 arcs à retirer, la qualité des solutions obtenues est supérieure en moyenne. Cependant, en fixant r à 1, les temps totaux moyens d'exécution sont réduits d'environ 87%. Pour la suite des tests, on pose $r = 1$.

Nous avons comparé les résultats obtenus avec l'algorithme RURAL en ordonnant les voisinages N_1, N_2, N_3 et N_4 de trois manières différentes. L'ordre O1 consiste à examiner les voisinages les plus étroits en premiers, puis les voisinages larges. L'ordre O1 est le suivant: N_1, N_2, N_3 et N_4. Le deuxième ordre correspond à l'ordre inverse du premier et le dernier ordre est choisi aléatoirement.

	Algorithme RURAL		
	Ordre O1	Ordre O2	Ordre O3
$E1_{moyen}$	**0,345**	0,374	0,561
$ET1_{moyen}$	0,169	0,173	0,258
$E2_{moyen}$	**6,667**	7,095	9,761
$ET2_{moyen}$	7,747	8,047	10,945
T_{moyen}	42,15	38,43	16,68
TT_{moyen}	62,54	57,15	29,56

Tableau 5.6: Influence de l'ordre des voisinages sur le comportement de RURAL

D'après les résultats du tableau 5.6, l'ordre O1 est celui qui permet d'obtenir les meilleurs résultats.

Nous avons finalement testé trois variantes de l'algorithme RURAL avec les voisinages N_1, N_2, N_3 et N_4 dans l'ordre, avec $r = 1$ pour N_3 et N_4. Les trois variantes consistent à modifier une étape donnée de la DVV. Dans la première version, la troisième étape de la DVV est supprimée. Cette version revient alors à effectuer successivement les quatre méthodes de descente associées à chacun des voisinages N_1, N_2, N_3 et N_4. Dans la seconde version, la stratégie *First Improvement* est utilisée [Han99]. Cette stratégie revient à passer d'un voisinage donné au voisinage suivant aussitôt que la solution s est améliorée lors de la descente à l'étape 2 de la DVV. Finalement, dans la dernière variante, la DVV est réamorcée avec le voisinage N_1 aussitôt qu'une meilleure solution est trouvée lors de la descente avec un voisinage donné plutôt que de poursuivre l'exploration des voisinages en séquence. Le principe de cette version exploite l'idée que les voisinages N_3 et N_4 peuvent servir à modifier une solution courante afin de donner de nouvelles opportunités de recherche pour appliquer à nouveau le voisinage N_1.

	Algorithme RURAL			
	DVV	Version 1	Version 2	Version 3
$E1_{moyen}$	**0,345**	0,383	0,350	0,347
$ET1_{moyen}$	0,169	0,203	0,169	0,186
$E2_{moyen}$	6,667	6,960	**6,299**	6,807
$ET2_{moyen}$	7,747	7,838	6,778	7,960
T_{moyen}	42,15	36,28	28,32	39,58
TT_{moyen}	62,54	50,71	45,39	61,93

Tableau 5.7: Résultats obtenus avec trois versions de l'algorithme RURAL

Dans le tableau 5.7, les résultats de la colonne DVV correspondent aux résultats obtenus sans aucune modification apportée à la DVV. Les résultats obtenus avec la deuxième version sont similaires à ceux obtenus avec la DVV originale. En utilisant

la version 2, les temps totaux moyens d'exécution sont réduits d'environ 30%. Pour la comparaison des résultats à la section 4, seule la variante 2 sera utilisée.

5.3.2 Algorithme AFFECTATION

Comme pour l'algorithme RURAL, l'influence des voisinages N_2, ..., N_k sur les performances de l'algorithme AFFECTATION a été testée en modifiant le nombre de voisinages considérés lors de la descente à voisinage variable. Pour cela, 40 nouvelles instances ont été générées (tableau 5.8). Une première variante utilise uniquement le voisinage N_1 lors de l'application de la DVV. Une autre considère les voisinages N_1 et N_2 en générant toutes les paires possibles d'affectation pour le voisinage N_2. La dernière variante alterne entre les voisinages N_1, N_2, ..., N_k décrits à la section 3 du chapitre précédent avec $m = n = 1$.

π_1	π_2	*No*	$\|V\|$	$\|E\|$	$\|A\|$	π_1	π_2	*No*	$\|V\|$	$\|E\|$	$\|A\|$
		1	68	73	18			21	72	92	23
		2	73	82	18			22	67	85	19
		3	72	81	10			23	66	79	24
		4	71	79	12			24	73	88	30
0,55	0,55	5	68	74	18	1	0,55	25	70	81	30
		6	75	88	10			26	62	76	17
		7	66	75	9			27	69	84	27
		8	66	77	8			28	68	81	24
		9	74	88	15			29	72	93	21
		10	69	78	14			30	63	82	14
		11	74	73	26			31	75	74	48
		12	78	77	22			32	72	71	42
		13	73	72	22			33	63	62	33
		14	72	71	23			34	69	68	39
0,55	1	15	68	67	21	1	1	35	74	73	49
		16	70	69	28			36	66	65	35
		17	77	76	30			37	79	78	56
		18	66	65	15			38	70	69	40
		19	74	73	24			39	71	70	44
		20	81	80	35			40	73	72	44

Tableau 5.8: Caractéristiques des 40 instances pour AFFECTATION (p=q=10)

Dans le tableau 5.9, N_{best} représente le nombre de fois qu'une variante produit une solution de coût inférieur ou égal à celui de toute autre variante (excluant les coûts fixes). $N_{improve}$ correspond au nombre de fois que la descente à voisinage variable améliore la solution au problème d'affectation obtenue après la première étape de l'algorithme AFFECTATION.

Le tableau 5.9 montre que les meilleurs résultats sont obtenus en utilisant les voisinages N_1 et N_2. Rappelons que pour cette variante, le voisinage N_2 consiste à examiner tous les ensembles possibles de paires d'affectation. En fait, les meilleures solutions sont obtenues en considérant ces deux voisinages pour les 40 problèmes traités. Les temps totaux moyens de calcul sont cependant non négligeables lorsque les voisinages N_1 et N_2 sont utilisés dans l'algorithme DVV2. On remarque que les solutions produites sont trouvées, en général, près de la fin de la recherche.

	Algorithme AFFECTATION		
	avec N_1	avec N_1, N_2	avec N_1, …, N_k ($m = n = 1$)
$E1_{moyen}$	0,266	**0,230**	0,266
$ET1_{moyen}$	0,122	0,101	0,122
$E2_{moyen}$	4,734	**3,956**	4,734
$ET2_{moyen}$	6,281	5,016	6,281
T_{moyen}	102,75	582,96	109,50
TT_{moyen}	103,22	641,59	146,48
N_{best}	0	40	0
$N_{improve}$	13	40	13

Tableau 5.9: Influence des voisinages N_2, …, N_k sur le comportement de l'algorithme AFFECTATION

Nous avons également testé l'influence de la taille des voisinages N_2, …, N_k sur les performances de l'algorithme AFFECTATION afin de voir si de meilleures solutions peuvent être obtenues en faisant varier les paramètres m et n. Des valeurs de m et n supérieures à 10 conduisent à des temps de calcul qui dépassent, en moyenne,

les 60 minutes alors que des valeurs de m et n inférieures à 10 donnent des solutions de moindre qualité que la deuxième variante du tableau 5.9. De plus, nous avons testé l'algorithme en inversant l'ordre des voisinages N_1 et N_2. Les meilleures performances ont été obtenues en ordonnant les voisinages N_1 et N_2 du plus étroit au plus large.

Pour la suite des tests, les voisinages N_1 et N_2 dans l'ordre établi seront utilisés dans l'algorithme DVV2 de l'algorithme AFFECTATION.

Nous avons finalement testé trois versions de l'algorithme AFFECTATION. Ces trois variantes sont les mêmes que celles testées pour l'algorithme RURAL. La troisième étape de la DVV est supprimée dans la première version. Dans la deuxième version, la stratégie *First Improvement* est utilisée. Cette stratégie permet de passer au voisinage suivant aussitôt que la solution courante est améliorée lors de la descente avec un voisinage donné. La dernière variante consiste à réamorcer la DVV avec le voisinage N_1 aussitôt qu'une meilleure solution est rencontrée lors de la descente avec le voisinage N_2.

	Algorithme AFFECTATION			
	DVV2	Version 1	Version 2	Version 3
$E1_{moyen}$	**0,230**	0,231	0,233	0,242
$ET1_{moyen}$	0,101	0,102	0,103	0,105
$E2_{moyen}$	**3,956**	3,992	4,033	4,211
$ET2_{moyen}$	5,016	5,095	5,162	5,628
T_{moyen}	582,96	600,95	369,40	372,34
TT_{moyen}	641,59	634,47	422,07	410,89
N_{best}	28	28	14	11
$N_{improve}$	40	40	40	40

Tableau 5.10: Résultats obtenus avec trois versions de l'algorithme AFFECTATION

Le tableau 5.10 montre qu'aucune des trois versions ne fournit de meilleurs résultats que ceux obtenus à l'aide de DVV2 du point de vue de la qualité des solutions. En utilisant la seconde version, les temps totaux moyens d'exécution sont

cependant réduits d'environ 34%. Pour les comparaisons à la prochaine section, seule la version 2 sera utilisée.

5.3.3 Algorithme MIXTE

Nous avons fait varier la valeur du paramètre k de l'algorithme MIXTE afin de tester différentes stratégies pour construire les circuits à l'étape 2 de l'algorithme. Pour chaque valeur de k, nous avons également testé les deux stratégies $S1$ et $S2$ présentées à la section 3 du chapitre précédent afin de choisir le sommet de départ du deuxième circuit à l'étape 2 (d) de l'algorithme MIXTE. Les 40 nouvelles instances générées pour tester différentes variantes de l'algorithme AFFECTATION ont été traitées.

	Algorithme MIXTE									
	$k=1$		$k=2$		$k=3$		$k=4$		$k=5$	
	$S1$	$S2$	$S1$	$S2$	$S1$	$S2$	$S1$	$S2$	$S1$	$S2$
$E1_{moyen}$	0,271	**0,270**	0,273	0,272	0,273	0,273	0,274	0,273	0,274	0,274
$ET1_{moyen}$	0,085	0,086	0,085	0,085	0,086	0,086	0,085	0,086	0,085	0,085
$E2_{moyen}$	3,387	**3,366**	3,419	3,408	3,415	3,405	3,423	3,404	3,428	3,434
$ET2_{moyen}$	3,067	3,031	3,124	3,102	3,096	3,088	3,084	3,060	3,125	3,139
TT_{moyen}	0,45	0,47	0,45	0,55	0,46	0,53	0,46	0,55	0,46	0,57

Tableau 5.11: Résultats obtenus en fonction du paramètre k

Le tableau 5.11 montre que des valeurs de k supérieures à 1 ne permettent pas d'atteindre de meilleures solutions. Pour une valeur de k fixée, la stratégie $S2$ donne, en moyenne, de meilleurs résultats que la stratégie $S1$. D'autres valeurs de k ont été testées sans toutefois fournir de meilleures performances. Pour les comparaisons de la section suivante, on pose $k = 1$ avec $S2$.

5.4 Comparaisons

Dans cette section, les algorithmes MIXTE, RURAL et AFFECTATION sont comparés à deux méthodes existantes pour la résolution du problème du postier rural orienté avec pénalités pour les virages et virages interdits. Il s'agit des deux méthodes heuristiques, PATCHING et H2 proposées par Corberán et al. [Cor00b] et décrites à la section 2.2.3 du chapitre 2. Une autre série de tests est ensuite effectuée sur des instances générées aléatoirement.

Problèmes de Corberán, Martí, Martínez et Soler

Les caractéristiques des 93 instances de Corberán, Martí, Martínez et Soler pour lesquelles $R = E \cup A$ sont présentées au tableau 5.12. Nous avons numéroté chaque problème dans la colonne *No*. Le nombre de sommets des graphes traités est noté $|V|$, le nombre d'arcs $|A|$ et le nombre d'arêtes $|E|$.

Les résultats obtenus par les heuristiques PATCHING et H2 de même que par les algorithmes RURAL, AFFECTATION et MIXTE sur chacun des 93 problèmes de Corberán, Martí, Martínez et Soler pour lesquels $R = E \cup A$ sont présentés au tableau 5.13. Pour chaque méthode, la colonne *coût* donne la somme $c(E_a \cup A_a) + c(T)$ et la colonne *temps* donne le temps total d'exécution. Seuls les résultats des problèmes résolus en deçà de 3 heures sont présentés. Ce tableau contient également pour chaque problème la valeur *Opt* d'une solution optimale trouvée à l'aide de la méthode exacte proposée par Corberán et al. [Cor00b] décrite à la section 2.2.3 du chapitre 2. Les valeurs des solutions qui égalent *Opt* sont en gras. Le tableau 5.14 résume les résultats obtenus par les différentes heuristiques. Les écarts $E1$ et $E2$ sont mesurés par rapport à la valeur de la solution optimale ou par rapport à la valeur de la meilleure solution obtenue par la méthode H2 de Corberán et al. lorsque la solution optimale n'a pas été trouvée. Les heuristiques PATCHING et H2 de Corberán et al. ont été testées sur un ordinateur Pentium II-350 [Cor00b].

No	$\|V\|$	$\|A\|$	$\|E\|$	*No*	$\|V\|$	$\|A\|$	$\|E\|$	*No*	$\|V\|$	$\|A\|$	$\|E\|$
1	40	90	10	32	100	220	60	63	160	340	30
2	40	90	20	33	100	220	80	64	160	340	40
3	40	90	30	34	100	220	100	65	160	370	10
4	40	90	40	35	100	260	10	66	160	370	20
5	40	110	10	36	100	260	20	67	160	370	30
6	40	110	20	37	100	260	30	68	160	370	40
7	40	110	30	38	100	260	40	69	160	370	60
8	40	110	40	39	120	260	10	70	160	370	80
9	60	140	10	40	120	260	20	71	160	370	100
10	60	140	20	41	120	260	30	72	180	380	10
11	60	140	30	42	120	260	40	73	180	380	20
12	60	140	40	43	120	260	60	74	180	380	30
13	60	160	10	44	120	260	80	75	180	380	40
14	60	160	20	45	120	260	100	76	180	400	10
15	60	160	30	46	120	290	10	77	180	400	20
16	60	160	40	47	120	290	20	78	180	400	30
17	80	170	10	48	120	290	30	79	180	400	40
18	80	170	20	49	120	290	40	80	180	400	60
19	80	170	30	50	140	300	10	81	180	400	80
20	80	170	40	51	140	300	20	82	180	400	100
21	80	200	10	52	140	300	30	83	200	420	10
22	80	200	20	53	140	300	40	84	200	420	20
23	80	200	30	54	140	330	10	85	200	420	30
24	80	200	40	55	140	330	20	86	200	420	40
25	80	200	60	56	140	330	30	87	200	420	60
26	80	200	80	57	140	330	40	88	200	420	80
27	80	200	100	58	140	330	60	89	200	420	100
28	100	220	10	59	140	330	80	90	200	440	10
29	100	220	20	60	140	330	100	91	200	440	20
30	100	220	30	61	160	340	10	92	200	440	30
31	100	220	40	62	160	340	20	93	200	440	40

Tableau 5.12: Caractéristiques des instances de Corberán et al.

Les résultats du tableau 5.14 permettent de diviser les algorithmes en trois catégories. Dans la première catégorie, on retrouve les méthodes constructives qui sont basées sur des concepts assez simples: MIXTE et PATCHING. Les écarts moyens $E1_{moyen}$ et $E2_{moyen}$ par rapport à la solution optimale ou par rapport à la solution obtenue par la méthode H2 de Corberán et al. sont respectivement de plus de 6% et 40% pour ces deux méthodes. L'écart-type $ET2_{moyen}$ étant élevé, il est cependant difficile de généraliser les résultats sur l'écart moyen $E2_{moyen}$ à d'autres instances.

No	*Opt*	MIXTE		PATCHING		RURAL		AFFECTATION		H2	
		coût	temps	coût	temps	coût	temps	coût	temps	coût	temps
1	1408	1698	0,33	1686	0,10	1552	547,67	1682	45,37	1494	0,07
2	713	1181	0,29	1036	0,21	1218	2054,98	998	57,47	818	0,07
3	856	1135	0,28	1375	0,32	1544	3611,18	1294	117,42	890	0,13
4	-	1123	0,36	1531	0,49	1567	6468,67	1244	164,21	1024	0,19
5	1389	1944	0,39	1736	0,21	1719	1040,69	1845	54,18	**1389**	0,06
6	1270	1823	0,36	1806	0,32	1652	1889,72	1877	87,55	1293	0,18
7	964	1463	0,42	1543	0,49	1779	7153,64	1181	123,77	1035	0,13
8	-	985	0,39	1716	0,65	-	-	1401	266,92	913	0,18
9	2038	3115	0,48	2544	0,38	2291	2353,25	2141	231,84	2040	0,21
10	2011	3311	0,49	2439	0,54	2745	4117,22	2370	321,33	**2011**	0,17
11	1503	2066	0,48	2205	0,76	-	-	2356	436,34	1625	0,41
12	-	2464	0,63	4302	0,99	2328	9392,28	2239	843,88	1541	0,34
13	3716	4531	0,54	4060	0,60	3964	4738,10	3887	251,32	**3716**	0,26
14	2916	3938	0,51	3127	0,76	3481	6234,75	3234	482,74	3027	0,28
15	2255	3131	0,70	2968	0,76	-	-	2664	535,39	2379	0,39
16	1944	2612	0,66	2907	1,31	-	-	2314	1140,00	2009	0,45
17	1477	2031	0,56	1710	0,60	1901	5242,33	2248	746,07	1487	0,23
18	1737	3173	0,74	2712	0,82	2361	7570,55	2314	1958,11	1840	0,31
19	-	2655	0,63	2763	1,09	2400	10649,83	2972	1946,65	1858	0,37
20	-	2801	0,85	2891	1,37	-	-	3012	2957,24	1733	0,37
21	2574	4612	1,02	2803	5,93	2861	9782,18	2821	1016,49	**2574**	0,36
22	2401	3800	0,93	2649	1,26	-	-	3280	1650,29	2456	0,48
23	2276	3554	1,03	2962	1,59	-	-	2797	1677,08	2286	0,53
24	-	3070	1,02	3616	2,08	-	-	3406	3122,93	2381	0,60
25	-	2810	1,24	3260	3,02	-	-	2760	3456,10	2354	0,85
26	-	2571	1,43	3733	4,33	-	-	2600	6156,65	2105	1,13
27	-	2114	2,02	3484	5,93	-	-	2393	6070,28	1895	1,44
28	2679	4993	1,10	3110	1,15	-	-	3299	2366,51	2782	0,47
29	2371	4074	1,20	3177	1,53	-	-	2732	2964,17	2511	0,53
30	-	4113	1,08	3477	1,86	-	-	2996	6432,18	2592	0,66
31	-	4188	1,46	3883	2,41	-	-	4005	4215,41	2700	0,78
32	-	3134	1,26	3142	3,46	-	-	2981	5039,17	2025	0,91
33	-	2207	1,76	3660	4,88	-	-	2850	9844,37	1860	1,32
34	-	2341	2,03	3938	6,64	-	-	-	-	1879	1,51
35	4009	6524	1,52	4009	1,97	-	-	4056	2938,01	**4009**	0,69
36	4498	6977	1,30	5128	2,41	-	-	5052	3289,98	4539	0,83
37	3216	4784	1,70	4139	2,96	-	-	3732	4455,87	3272	0,89
38	-	4797	1,69	4542	3,68	-	-	3929	8378,64	3408	1,02
39	2880	4168	1,35	3321	1,86	-	-	3562	2639,39	2908	0,64
40	2628	3188	1,70	3518	2,30	-	-	3502	4154,65	2680	0,72
41	-	4089	1,76	3334	2,80	-	-	3537	8332,64	2539	0,87
42	-	3886	1,66	3501	3,35	-	-	-	-	2592	1,03
43	-	3714	2,00	5595	4,83	-	-	-	-	3219	1,48
44	-	3482	2,03	4938	6,48	-	-	-	-	3026	1,71
45	-	2579	2,60	4914	8,67	-	-	-	-	2576	2,07
46	4408	7168	2,00	4696	2,58	-	-	4613	3289,90	4516	0,94
47	4184	6022	1,91	5113	3,13	-	-	5045	4999,19	4185	1,12

Tableau 5.13: Comparaison: problèmes de Corberán et al.

No	*Opt*	MIXTE		PATCHING		RURAL		AFFECTATION		H2	
		coût	temps	coût	temps	coût	temps	coût	temps	coût	temps
48	4110	6426	2,39	5276	3,78	-	-	4890	10315,12	4163	1,25
49	-	5516	2,39	4722	4,50	-	-	-	-	3584	1,37
50	2616	5251	2,57	3277	2,63	-	-	3218	5402,60	**2616**	0,88
51	2773	4587	2,65	3787	3,18	-	-	3348	7955,15	**2773**	1,05
52	-	4645	2,91	4300	3,84	-	-	-	-	2715	1,20
53	-	4502	2,57	5044	4,66	-	-	-	-	3157	1,44
54	6106	8934	2,64	6684	3,68	-	-	6757	8091,68	6215	1,29
55	5321	8417	3,31	6387	4,33	-	-	-	-	5532	1,52
56	4535	7082	3,13	5646	5,10	-	-	-	-	4635	1,69
57	-	6903	3,50	6337	6,04	-	-	-	-	4837	1,78
58	-	5244	3,72	5660	8,01	-	-	-	-	4220	2,36
59	-	4622	3,61	7299	10,38	-	-	-	-	3934	2,61
60	-	4780	4,83	6708	13,23	-	-	-	-	3507	3,51
61	2921	5364	2,91	3691	3,73	-	-	-	-	**2921**	1,18
62	2994	5738	3,05	4600	4,50	-	-	-	-	3061	1,37
63	-	5151	3,58	4856	5,32	-	-	-	-	3211	1,63
64	-	4975	3,31	4717	6,15	-	-	-	-	3207	1,82
65	5875	9963	4,30	6396	4,88	-	-	-	-	5958	1,81
66	4991	8313	3,76	6097	5,76	-	-	-	-	5143	1,93
67	-	9188	4,33	6377	6,81	-	-	-	-	5379	2,24
68	-	9281	4,52	7343	7,79	-	-	-	-	5504	2,41
69	-	7637	4,64	6583	10,27	-	-	-	-	4957	2,99
70	-	6880	5,63	7424	13,01	-	-	-	-	4767	3,42
71	-	5940	5,51	7287	16,31	-	-	-	-	4471	4,10
72	3540	6237	4,11	4329	5,05	-	-	-	-	3601	1,61
73	3381	5954	4,42	4540	5,98	-	-	-	-	**3381**	1,84
74	-	6508	4,67	4958	6,92	-	-	-	-	3831	2,20
75	-	5458	4,26	5332	8,01	-	-	-	-	3568	2,28
76	4864	7798	4,23	5184	5,93	-	-	-	-	4870	1,87
77	5153	8076	5,12	6714	6,97	-	-	-	-	5174	2,25
78	-	7189	4,38	5922	8,01	-	-	-	-	4777	2,39
79	-	6418	4,97	6263	9,22	-	-	-	-	4341	2,67
80	-	6818	5,49	6254	11,86	-	-	-	-	4553	3,28
81	-	6146	5,96	7469	14,99	-	-	-	-	4203	4,13
82	-	4665	7,77	6766	18,45	-	-	-	-	3580	4,67
83	3772	6913	5,38	4469	6,64	-	-	-	-	3856	2,09
84	3369	5403	5,74	5051	7,79	-	-	-	-	**3369**	2,30
85	-	7398	6,66	5554	8,78	-	-	-	-	4055	2,56
86	-	6823	6,30	5316	10,16	-	-	-	-	3877	2,90
87	-	5914	7,26	7163	13,01	-	-	-	-	4388	3,73
88	-	5388	7,36	7292	16,31	-	-	-	-	4277	4,50
89	-	6038	8,45	8034	20,15	-	-	-	-	4522	5,02
90	5190	10 304	5,59	5708	7,90	-	-	-	-	5227	2,46
91	5594	9565	5,80	7347	9,00	-	-	-	-	5698	2,80
92	-	8658	6,32	7260	10,27	-	-	-	-	5725	3,23
93	-	9227	5,88	8066	11,64	-	-	-	-	5494	3,49

Tableau 5.13 (suite): Comparaison: problèmes de Corberán et al.

	MIXTE	PATCHING	RURAL	AFFECTATION	H2
$E1_{moyen}$	0,082	0,062	0,052	0,042	**0,002**
$ET1_{moyen}$	0,039	0,033	0,025	0,022	0,003
$E2_{moyen}$	0,503	0,425	0,366	0,287	**0,011**
$ET2_{moyen}$	0,215	0,279	0,248	0,173	0,023
TT_{moyen}	2,84	5,23	7097,81	3065,70	1,52

Tableau 5.14: Résultats sur les problèmes de Corberán et al.

Le deuxième groupe comprend les méthodes RURAL et AFFECTATION caractérisées par une phase de descente à voisinage variable. L'écart moyen $E1_{moyen}$ par rapport à la solution optimale ou par rapport à la solution obtenue par la méthode H2 de Corberán et al. est inférieur à 6%. Bien que les écarts moyens $E2_{moyen}$ soient réduits d'environ 30%, ils restent cependant élevés. Ces méthodes demeurent aussi peu robustes que celles de la première catégorie, l'écart-type $ET2_{moyen}$ étant élevé. On remarque que les temps moyens d'exécution sont non négligeables. Ils demeurent toutefois acceptables pour le niveau de planification associé à ce type de problème.

Finalement, la méthode H2 de Corberán et al. complète le dernier groupe. Cette méthode permet d'obtenir rapidement les meilleures solutions. Beaucoup plus élaborée que les méthodes précédentes, cette méthode se caractérise par un écart moyen $E1_{moyen}$ inférieur à 0,5%, un écart moyen $E2_{moyen}$ près de 1% et au moins 10 solutions optimales sont atteintes sur les 93 problèmes traités. Cette méthode est également très robuste, les écarts-types $ET1_{moyen}$ et $ET2_{moyen}$ étant de 0,3% et de 2,3% respectivement.

Problèmes générés aléatoirement

Nous avons testé les algorithmes MIXTE, RURAL et AFFECTATION sur des instances générées aléatoirement (tableau 5.15) afin d'examiner le comportement de ces algorithmes sur des graphes rectilinéaires dont le nombre d'arêtes est important.

p	q	π_1	π_2	No	$\|V\|$	$\|E\|$	$\|A\|$	p	q	π_1	π_2	No	$\|V\|$	$\|E\|$	$\|A\|$
		0,55	0,55	1	11	11	1			0,55	0,55	41	55	59	11
		0,60	0,60	2	15	15	5			0,60	0,60	42	54	59	7
		0,65	0,65	3	15	15	1			0,65	0,65	43	53	61	8
		0,70	0,70	4	12	11	4			0,70	0,70	44	59	61	14
4	5	0,75	0,75	5	15	14	3	5	15	0,75	0,75	45	49	51	11
		0,80	0,80	6	14	14	3			0,80	0,80	46	57	65	19
		0,85	0,85	7	17	17	5			0,85	0,85	47	57	61	24
		0,90	0,90	8	14	13	4			0,90	0,90	48	48	49	19
		0,95	0,95	9	13	12	5			0,95	0,95	49	46	46	20
		1,00	1,00	10	15	14	7			1,00	1,00	50	53	52	30
		0,55	0,55	11	22	24	4			0,55	0,55	51	77	91	15
		0,60	0,60	12	15	15	1			0,60	0,60	52	77	86	25
		0,65	0,65	13	17	18	3			0,65	0,65	53	68	75	17
		0,70	0,70	14	21	20	6			0,70	0,70	54	66	74	17
5	6	0,75	0,75	15	24	27	6	10	10	0,75	0,75	55	73	81	27
		0,80	0,80	16	21	22	6			0,80	0,80	56	69	74	23
		0,85	0,85	17	18	17	8			0,85	0,85	57	71	79	26
		0,90	0,90	18	26	27	12			0,90	0,90	58	70	71	31
		0,95	0,95	19	25	25	10			0,95	0,95	59	79	78	50
		1,00	1,00	20	25	24	14			1,00	1,00	60	69	68	36
		0,55	0,55	21	32	35	4			0,55	0,55	61	106	124	14
		0,60	0,60	22	28	30	5			0,60	0,60	62	108	122	24
		0,65	0,65	23	30	34	7			0,65	0,65	63	111	126	31
		0,70	0,70	24	30	31	6			0,70	0,70	64	109	126	35
5	8	0,75	0,75	25	23	25	4	10	15	0,75	0,75	65	104	112	42
		0,80	0,80	26	25	27	6			0,80	0,80	66	98	111	30
		0,85	0,85	27	27	27	10			0,85	0,85	67	107	113	47
		0,90	0,90	28	26	26	10			0,90	0,90	68	97	101	43
		0,95	0,95	29	27	26	14			0,95	0,95	69	105	105	64
		1,00	1,00	30	28	27	14			1,00	1,00	70	109	108	67
		0,55	0,55	31	40	42	7			0,55	0,55	71	137	157	20
		0,60	0,60	32	39	42	7			0,60	0,60	72	120	137	24
		0,65	0,65	33	37	42	7			0,65	0,65	73	136	153	37
		0,70	0,70	34	38	44	7			0,70	0,70	74	150	165	54
5	10	0,75	0,75	35	42	47	13	10	20	0,75	0,75	75	135	145	46
		0,80	0,80	36	38	43	15			0,80	0,80	76	149	169	59
		0,85	0,85	37	32	32	9			0,85	0,85	77	147	158	58
		0,90	0,90	38	35	36	12			0,90	0,90	78	140	148	61
		0,95	0,95	39	36	38	17			0,95	0,95	79	147	149	81
		1,00	1,00	40	39	38	20			1,00	1,00	80	142	141	94

Tableau 5.15: Caractéristiques des instances générées aléatoirement

Les résultats obtenus par les algorithmes MIXTE, RURAL et AFFECTATION sur les 80 instances générées aléatoirement sont présentés au tableau 5.16.

No	*BI2*	MIXTE		RURAL		AFFECTATION	
		coût	temps	coût	temps	coût	temps
1	14	**37**	0,22	350	5,11	421	0,05
2	85	**259**	0,21	350	14,61	311	0,23
3	32	**569**	0,22	649	10,08	657	0,18
4	32	**220**	0,21	325	3,32	274	0,10
5	24	**298**	0,14	394	14,30	394	0,16
6	22	374	0,14	**347**	8,72	**347**	0,16
7	38	**294**	0,15	446	24,03	490	0,45
8	163	**451**	0,14	458	8,91	458	0,15
9	43	**258**	0,14	263	9,72	269	0,11
10	234	620	0,14	647	17,94	**383**	0,31
11	30	620	0,17	**545**	88,64	793	0,91
12	45	**600**	0,15	915	4,42	724	0,18
13	52	**271**	0,15	292	24,89	**271**	0,34
14	42	870	0,21	**397**	102,46	**397**	0,74
15	135	**660**	0,17	811	139,86	710	2,49
16	491	921	0,15	1126	104,39	**889**	0,84
17	215	726	0,15	616	30,02	**532**	0,38
18	1016	1835	0,17	1650	505,65	**1472**	3,27
19	180	692	0,16	540	280,68	**448**	2,29
20	790	1516	0,15	1270	251,04	**982**	2,39
21	50	**816**	0,18	1017	593,93	846	5,99
22	401	1553	0,16	1408	231,48	**1272**	4,09
23	513	1026	0,18	1181	556,16	**791**	4,68
24	248	1003	0,18	916	316,50	**745**	5,29
25	178	590	0,15	828	130,27	**574**	1,22
26	218	926	0,16	806	148,39	**752**	2,46
27	122	619	0,16	604	386,95	**586**	4,11
28	708	1947	0,17	1369	267,40	**1141**	2,51
29	1914	3133	0,18	2285	504,39	**2023**	2,97
30	667	1481	0,18	1538	229,21	**994**	4,81
31	610	2042	0,20	1854	1455,17	**1741**	26,53
32	246	**961**	0,19	1429	790,32	1364	20,93
33	486	1250	0,19	1554	1575,19	**1209**	16,66
34	163	**1213**	0,21	1381	1163,59	1374	20,89
35	79	**892**	0,22	1891	2438,94	1515	47,46
36	333	**1069**	0,21	1923	1779,76	1146	34,55
37	283	1179	0,18	1066	352,52	**873**	5,78
38	573	**1089**	0,19	1566	713,08	1159	12,31
39	200	1232	0,19	1841	690,88	**1156**	24,02
40	1752	2344	0,21	2173	2278,13	**2120**	20,64

Tableau 5.16: Résultats sur les 80 instances générées aléatoirement

No	*BI2*	MIXTE		RURAL		AFFECTATION	
		coût	temps	coût	temps	coût	temps
41	253	**1532**	0,30	1866	6648,91	1563	112,64
42	108	**1498**	0,29	2423	6140,03	1629	82,57
43	247	1399	0,27	1802	6684,11	**1323**	91,63
44	411	2161	0,34	2290	7597,01	**1710**	141,23
45	1744	2324	0,26	2316	4944,71	**2169**	39,19
46	202	**1290**	0,32	1711	6461,46	1484	159,91
47	1064	2640	0,34	2257	8522,91	**1688**	139,70
48	1228	2441	0,26	2269	2495,37	**2007**	69,16
49	835	2313	0,25	1973	3610,41	**1548**	45,69
50	1240	3394	0,30	2525	9432,79	**1944**	94,03
51	309	**1828**	0,52	-	-	2716	872,53
52	719	**2205**	0,52	-	-	2380	896,07
53	437	2156	0,40	-	-	**2079**	285,71
54	1360	3297	0,40	-	-	**2054**	233,25
55	1548	3727	0,55	-	-	**2682**	575,62
56	2435	4571	0,49	-	-	**3113**	202,06
57	1752	4051	0,52	-	-	**3209**	431,53
58	1629	3763	0,48	-	-	**2628**	329,25
59	3086	4634	0,65	-	-	**3796**	539,12
60	1405	3297	0,45	-	-	**2350**	263,72
61	324	**2543**	1,10	-	-	3981	3308,39
62	640	**2701**	1,15	-	-	2827	4149,15
63	435	**2201**	1,30	-	-	2763	4757,61
64	1294	4572	1,25	-	-	**3554**	2984,88
65	1006	3620	1,08	-	-	**3063**	2937,42
66	791	**2800**	0,96	-	-	3020	2433,61
67	3762	8064	1,28	-	-	**5111**	3055,10
68	2170	3955	0,88	-	-	**3257**	1547,22
69	2608	5832	1,08	-	-	**3629**	2509,84
70	2999	5106	1,32	-	-	**4276**	3383,82
71	587	**3703**	2,13	-	-	3786	9694,99
72	1006	3590	1,70	-	-	**3418**	5574,73
73	1813	5130	2,02	-	-	**4002**	10522,53
74	4347	9101	3,28	-	-	**6370**	-
75	2304	6031	2,28	-	-	**4437**	7170,40
76	1441	**4217**	2,94	-	-	4290	-
77	1953	4881	2,68	-	-	**4616**	-
78	2228	5566	2,69	-	-	**5347**	-
79	4185	8876	3,12	-	-	**6378**	-
80	4122	7496	2,53	-	-	**5326**	-

Tableau 5.16 (suite): Résultats sur les 80 instances générées aléatoirement

Le numéro de chaque problème figure dans la colonne *No*. Le tableau 5.16 contient également pour chaque problème la valeur de la borne inférieure *BI2* obtenue

en soustrayant de $BI1$ le coût total $c(E \cup A)$ des liens de $E \cup A$. Pour chaque problème, le coût $c(E_a \cup A_a) + c(T)$ de la meilleure solution trouvée est indiqué en caractère gras. Les temps totaux d'exécution sont également rapportés pour chaque méthode.

Dans le tableau 5.17, les valeurs $E1_{moyen}$ et $ET1_{moyen}$ représentent respectivement l'écart moyen par rapport à la borne inférieure $BI1$ et l'écart-type de $E1_{moyen}$. $E2_{moyen}$ et $ET2_{moyen}$ correspondent respectivement à l'écart moyen par rapport à la borne inférieure $BI2$ et à l'écart-type de $E2_{moyen}$. Le nombre de fois qu'une méthode produit une solution de coût inférieur ou égal (excluant les coûts fixes) à celui des solutions produites par les autres méthodes est noté N_{best}. TT_{moyen} correspond au temps total moyen d'exécution.

	MIXTE	RURAL	AFFECTATION
$E1_{moyen}$	0,299	0,350	**0,258**
$ET1_{moyen}$	0,096	0,143	0,138
$E2_{moyen}$	**3,873**	6,255	4,551
$ET2_{moyen}$	4,403	6,776	6,091
N_{best}	28	3	**52**
TT_{moyen}	0,65	1615,78	944,84

Tableau 5.17: Résumé des résultats sur les instances générées aléatoirement

Le tableau 5.17 montre que l'algorithme AFFECTATION donne pour la plupart des instances de meilleures solutions que les algorithmes MIXTE et RURAL. En effet, 50 fois sur 80, l'algorithme AFFECTATION a trouvé une solution de coût strictement inférieur au coût des solutions produites par les deux autres méthodes. Pour les deux autres instances, il y a égalité avec l'algorithme RURAL. On remarque toutefois que l'algorithme AFFECTATION est moins performant que l'algorithme MIXTE sur des graphes de petite taille. Cela peut dépendre des principes qui régissent chacun des algorithmes. L'algorithme MIXTE est une méthode qui résout le PPCM-PV en deux phases successives. Une première phase consiste à résoudre le

sous-problème de la minimisation des passages à vide. Le sous-problème de la minimisation des pénalités pour les virages est ensuite résolu en construisant un circuit eulérien à l'aide d'une méthode gloutonne. Séparer ces deux sous-problèmes simplifie la méthode de résolution mais la qualité des solutions qui en découlent demeure toutefois inférieure lorsque la taille des graphes augmente. Il existe une interdépendance entre le sous-problème de la minimisation des passages à vide et celui de la minimisation des pénalités pour les virages. En effet, le nombre et l'emplacement des passages à vide peuvent dépendre de la structure des pénalités. En traitant simultanément les deux sous-problèmes, l'algorithme AFFECTATION parvient ainsi à atteindre, en général, de meilleures solutions pour des graphes de grande taille.

L'écart moyen $E1_{moyen}$ par rapport à la borne inférieure $BI1$ obtenu avec l'algorithme AFFECTATION demeure cependant élevé. Cela peut dépendre de la mauvaise performance de la borne inférieure $BI1$ sur ce type de graphes.

En moyenne, la qualité des solutions obtenues, en excluant les coûts fixes, par l'algorithme AFFECTATION est moins bonne que celle fournie par les solutions de l'algorithme MIXTE. En effet, l'écart moyen $E2_{moyen}$ par rapport à la borne inférieure $BI2$ obtenu avec l'algorithme AFFECTATION est supérieur à celui de l'algorithme MIXTE. Ce comportement peut s'expliquer par le fait que l'algorithme AFFECTATION est moins robuste que l'algorithme MIXTE, l'écart-type $ET2_{moyen}$ est beaucoup plus grand que celui de MIXTE.

Malgré que la transformation du PPCM-PV en un PPR orienté soit formellement attrayante, les résultats du tableau 5.17 montrent que l'algorithme RURAL présente le pire comportement algorithmique. Les tailles de graphes qui peuvent être résolus en deçà de trois heures demeurent petites. Il faut dire aussi que les méthodes existantes pour résoudre le problème du postier rural dans un graphe orienté sont peu nombreuses. Il reste beaucoup de recherche à faire dans ce domaine.

Conclusion

Nous avons proposé et comparé trois nouvelles heuristiques pour résoudre le problème du postier chinois mixte avec pénalités pour les virages (PPCM-PV). Les trois heuristiques sont basées sur des méthodes existantes pour la résolution de PTA avec pénalités pour les virages et deux d'entre elles utilisent le principe de la descente à voisinage variable. Une des trois méthodes est basée sur la transformation d'un PPC en un PPR orienté et s'avère très flexible puisqu'elle permet de résoudre les versions non orienté, orienté et mixte des problèmes du postier chinois et rural.

Nous proposons également une adaptation d'une borne inférieure développée à l'origine pour le problème du postier rural dans un graphe orienté avec pénalités pour les virages (PPR-PV) afin d'évaluer la qualité des solutions produites par nos algorithmes sur des graphes rectilinéaires mixtes. Les résultats montrent qu'une méthode s'avère plus efficace que les autres lorsque la taille des graphes augmente.

D'après les comparaisons avec deux méthodes existantes pour la résolution d'un problème plus général sur la base des instances tirées de la littérature, l'heuristique de Corberán et al. [Cor00b] pour le PPR-PV demeure la meilleure méthode disponible actuellement pour résoudre rapidement ce genre de problème. Cette heuristique est cependant beaucoup plus élaborée dans sa conception que les méthodes que nous avons proposées pour le PPCM-PV.

Les comparaisons indiquent également que des progrès peuvent encore être réalisés en développant de meilleures bornes inférieures et en améliorant les heuristiques que nous avons présentées. Entre autres, la structure particulière du PPR orienté obtenu après la transformation du PPC mérite d'être étudiée.

Bibliographie

[All90] W. G. Allen Jr. Simulating Traffic Control Devices in a Sub-area Network. *Transportation Research Record 1283*, 63-69, 1990.

[Ani96] S. Anily, M. Gendreau et G. Laporte. Two Solvable Cases of the Mixed Rural Postman Problem. Rapport technique 96-35, Centre de recherche sur les transports, 1996.

[Ani97] S. Anily, M. Gendreau et G. Laporte. A Solvable Case of the Mixed Rural Postman Problem. Rapport technique 97-34, Centre de recherche sur les transports, 1997.

[Ani00] S. Anily, M. Gendreau et G. Laporte. Optimal Sequencing of Tasks on a Tree Shaped Structure. *Ricerca Operativa*, 29, 327-335, 2000.

[Ass95] A. A. Assad et B. L. Golden. Arc Routing Methods and Applications. Chapitre 5 de "Network Routing ". *Handbooks in Operations Research and Mangement Science*, M. O. Ball, T. L. Magnanti, C. L. Monma et G. L. Nemhauser, eds., North-Holland, Amsterdam, 375-483, 1995.

[Bal88] M. O. Ball et M. J. Magazine. Sequencing of Insertions in Printed Circuit Board Assembly. *Operations Research*, 36, 192-201, 1988.

[Bel74] E. J. Beltrami et L. D. Bodin. Networks and Vehicle Routing for Municipal Waste Collection. *Networks*, 4, 65-94, 1974.

[Ben99] E. Benavent et D. Soler. The Directed Rural Postman Problem with Turn Penalties. *Transportation Science*, 33, 408-418, 1999.

[Ben72] B. Bennett et D. Gazis. School Bus Routing by Computer. *Transportation Research*, 6, 317-325, 1972

[Ber73] C. Berge. *Graphes et hypergraphes*, Dunod, Paris, 1973.

[Bod78] L. D. Bodin et S. J. Kursh. A Computer-Assisted System for the Routing and Scheduling of Street Sweepers. *Operations Research*, 26, 525-537, 1978.

[Bod79] L. D. Bodin et S. J. Kursh. A Detailed Description of a Computer System for the Routing and Scheduling of Street Sweepers. *Computers & Operations Research*, 6, 181-198, 1979.

[Bod83] L. D. Bodin, B. L. Golden, A. A. Assad et M. O. Ball. Routing and Scheduling of Vehicles ans Crews. The State of the Art. *Computers & Operations Research*, 10, 63-211, 1983.

[Bod89] L. D. Bodin, G. Fagin, R. Welebny et J. Greenberg. The Design of a Computerized Sanitation Vehicle Routing and Scheduling System for the Town of Oyster Bay, New York. *Computers & Operations Research*, 16, 45-54, 1989.

[Bor98a] A. Boroujerdi et J. Uhlmann. An efficient Algorithm for Computing Least Cost Paths with Turn Constraints. *Information Processing Letters*, 67, 317-321, 1998.

[Bor98b] A. Boroujerdi et J. Uhlmann. Empirical Analysis of New Methods for Computing Minimum Cost Paths with Turn Constraints. *Proceedings of the International Society for Optical Engineering*, 3366, 262-272, 1998.

[Bou00] T. Bousonville et H. Kopfer. A Hybrid Evolutionary Algorithm for the Mixed Rural Postman Problem with Turn Penalties. Soumis à *Transportation Science*, 2000.

[Bro70] M. Bronzini. The Effect of Left Turn Penalties on Traffic Assignment Accuracy. *Highway Research Record*, 221-231, 1970.

[Bru81] P. Brucker. The Chinese Postman Problem for Mixed Graphs. Dans *Graph Theoretic Concepts in Computer Science*, H. Noltemeier, ed., Springer, Berlin, 354-366, 1981.

[Car80] G. Carpaneto et P. Toth. Some New Branching and Bounding Criteria for the Asymmetric Traveling Salesman Problem. *Management Science*. 26, 736-743, 1980.

[Chr73] N. Christofides. The Optimum Traversal of a Graph. *Omega*, 6, 719-732, 1973.

[Chr75] N. Christofides. *Graph Theory. An Algorithmic Approach.* Academic Press, London. 1975.

[Chr76] N. Christofides. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. Rapport N° 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, 1976.

[Chr81] N. Christofides, V. Campos, A. Corberán et E. Mota. An Algorithm for the Rural Postman Problem. Imperial College Report IC.O.R.81.5, 1981.

[Chr84] N. Christofides, E. Benavent, V. Campos, A. Corberán et E. Mota. An Optimal Method for the Mixed Postman Problem. Dans *System Modelling and Optimization*, Lecture Notes in Control and Information Sciences, 59, P. Thoft-Christensen, éd., Springer, Berlin, 1984.

[Chr86] N. Christofides, V. Campos, A. Corberán et E. Mota. An Algorithm for the Rural Postman Problem on a Directed Graph. *Mathematical Programming Study*, 26, 155-166, 1986.

[Clo98] J. Clossey. Problèmes de tournées sur les arêtes avec pénalités pour virages. Mémoire de maîtrise, École des Hautes Études Commerciales, Montréal, Canada, 1998.

[Cor94] A. Corberán et J. M. Sanchis. A Polyhedral Appropach to the Rural Postman Problem. *European Journal of Operational Research*, 79, 95-242, 1994.

[Cor00a] A. Corberán, R. Martí et A. Romero. Heuristics for the Mixed Rural Postman Problem. *Computers & Operations Research*, 27, 183-203, 2000.

[Cor00b] A. Corberán, R. Martí, E. Martínez et D. Soler. The Rural Postman on Mixed Graphs with Turn Penalties. À paraître dans *Computers & Operations Research*, 2000.

[Cro58] G. A. Croes. A Method for Solving Travelling-Salesman Problems. *Operations Research*, 6, 791-812, 1958.

[Dav69] T. B. Davinroy, M. S. Bronzini et J. H. Herendeen. Transportation Network Analysis Utilizing Selective Left Turn Penalties. Pennsylvania Transportation and Traffic Safety Center, Pennsylvania State University, University Park, 1969.

[deC98] P. F. de Córdoba, L. M. García Raffi et J. M. Sanchis. A Heuristic Algorithm Based on Monte Carlo Methods for the Rural Postman Problem. *Computers & Operations Research*, 25, 1097-1106, 1998.

[Edm73] J. Edmonds et E. L. Johnson. Matching, Euler Tours and the Chinese Postman. *Mathematical Programming*, 5, 88-124, 1973.

[Eis95I] H. A. Eiselt, M. Gendreau et G. Laporte. Arc Routing Problems, Part I : The Chinese Postman Problem. *Operations Research*, 43, 231-242, 1995.

[Eis95II] H. A. Eiselt, M. Gendreau, G. Laporte. Arc Routing Problems, Part II : The Rural Postman Problem. *Operations Research*, 43, 399-414, 1995.

[Eul1736] L. Euler. Solutio Problematis ad Geometrian Situs Pertinentis. *Commentarii academiae scientarum Petropolitanae*, 8, 128-140, 1736.

[Fis92] M. Fischetti et P. Toth. An Additive Bounding Procedure for the Asymetric Travelling Salesman Problem. *Mathematical Programming*, 53, 173-197, 1992.

[Fle90] H. Fleischner. Eulerian Graphs and Related Topics, Part 1, Volume 1. *Annals of Discrete Mathematics*, 45, North-Holland, Amsterdam, 1990.

[Fle91] H. Fleischner. Eulerian Graphs and Related Topics, Part 1, Volume 2. *Annals of Discrete Mathematics*, 50, North-Holland, Amsterdam, 1991.

[Fog66] L. J. Fogel, A. J. Owens et M. J. Walsh. *Artificial Intelligence through Simulated Evolution*, Wiley, New York, 1966.

[For62] L. R. Ford et D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[Fre79] G. N. Frederickson. Approximation Algorithms for some Postman Problems. *Journal of the Association for Computing Machinery*, 26, 538-554, 1979.

[Gar99] R. S. Garfinkel et I. R. Webb. On Crossings, the Crossing Postman Problem, and the Rural Postman Problem. *Networks* 34, 173-180, 1999.

[Gen92] M. Gendreau, A. Hertz et G. Laporte. New Insertion and Post-optimization Procedures for the Travelling Salesman Problem. *Operations Research*, 40, 1086-1094, 1992.

[Ghi00] G. Ghiani et G. Laporte. A Branch-and-cut Algorithm for the Undirected Rural Postman Problem. *Mathematical Programming*, 87, 467-481, 2000.

[Gol81] B. L. Golden et R. T. Wong, Capacitated Arc Routing Problems. *Networks*, 11, 305-315, 1981.

[Gua62] M. Guan. Graphic Programming Using Odd and Even Points. *Chinese Mathematics*, 1, 273-277, 1962.

[Gün93] H. Gün. Polyhedral Structure and Efficient Algorithms for Certain Classes of the Directed Rural Postman Problem. Thèse de doctorat, University of Maryland at College Park, 1993.

[Han97] P. Hansen et N. Mladenović. Variable Neighborhood Search. *Computers & Operations Research*, 24, 1097-1100, 1997.

[Han99] P. Hansen et N. Mladenović. First Improvement may be Better than Best Improvement: An Empirical Study. Les Cahiers du GERAD, G-99-40, 1999.

[Han00] P. Hansen et N. Mladenović. Recherche à voisinage variable. Les Cahiers du GERAD, G-2000-08, 2000.

[Her99] A. Hertz, G. Laporte et P. Nanchen-Hugo. Improvement Procedures for the Undirected Rural Postman Problem. *INFORMS Journal on Computing*, 11, 53-62, 1999.

[Kap79] C. H. Kappauf et G. J. Koehler. The Mixed Postman Problem. *Discrete Applied Mathematics*, 1, 89-103, 1979.

[Kar79] R. M. Karp. A Patching Algorithm for the Nonsymmetric Traveling Salesman Problem. *SIAM Journal on Computing*, 8, 561-573, 1979.

[Kau67] A. Kaufmann. *Graphs, Dynamic Programming and Finite Games*. Academic Press, New York, 1967.

[Kir69] R. F. Kirby et R. B. Potts. The Minimum Route Problem for Networks with Turn Penalties and Prohibitons. *Transportation Research*, 3, 397-408, 1969.

[Lap89] G. Laporte, S. Chapleau, P.-É. Landry et H. Mercure. An Algorithm for the Design of Mailbox Collection Routes in Urban Areas. *Transportation Research*, B 23B, 271-280, 1989.

[Lap97] G. Laporte. Modeling and Solving Several Classes of Arc Routing Problems as Traveling Salesman Problems. *Computers & Operations Research*, 24, 1057-1061, 1997.

[Lap99] G. Laporte et F. Semet. Computational Evaluation of a Transformation Procedure for the Symmetric Generalized Traveling Salesman Problem. *INFOR*, 37, 114-120, 1999.

[Lee97] J. H. Lee, C.-Y. Chung, B. H. Lee et M. H. Choi. Turn Penalty Problem Model for Minimum Time AGV Routing. *Proceedings of the IECON'97 23rd International Conference on Industrial Electronics, Control and Instrumentation*, Catalogue 97CH6066, p.4, vol. xxvii+1555, 1323-1326, 1997.

[Len76] J. K. Lenstra et A. H. G. Rinnooy Kan. On General Routing Problems. *Networks*, 6, 273-280, 1976.

[Lin88] Y. Lin et Y. Zhao. A New Algorithm for the Directed Chinese Postman Problem. *Computers & Operations Research*, 15, 577-584, 1988.

[Mar98] E. Martínez et D. Soler. The One Vehicle Postman Problem with Turn Penalties. Working paper 31, Université de Valence, Espagne, 1998.

[McB82] R. McBride. Controlling Left and U-turns in the Routing of Refuse Collection Vehicles. *Computers & Operations Research*, 9, 145-152, 1982.

[Min78] E. Minieka. *Optimisation Algorithms for Networks and Graphs*. Marcel Dekker, New York, 1978.

[Min79] E. Minieka. The Chinese Postman Problem for Mixed Networks. *Mangement Science*, 25, 643-648, 1979.

[Mit99] M. Mittaz. Problèmes de Cheminements Optimaux dans les Réseaux avec Contraintes Associées aux Arcs. Thèse de doctorat, École Polytechnique Fédérale de Lausanne, Suisse, 1999.

[Nam98] S. Namkoong, J.-H. Rho et J.-U. Choi. Development of the Tree-Based Link Labeling Algorithm for Optimal Path-Finding in Urban Transportation Networks. *Mathematical & Computer Modelling*, 27, 51-65, 1998.

[Nob96] Y. Nobert et J.-C. Picard. An Optimal Algorithm for the Mixed Chinese Postman Problem. *Networks*, 27, 95-108, 1996.

[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 Association for Computing Machinery*, 23, 544-554, 1976.

[Par98] C.-K. Park, S. Park et H.C. Jin. A Fast Algorithm for the Shortest Path Problem for Network with Turn Penalties and Prohibitions. *Journal of Korean Operations Research and Management Science Society*, 23, 17-26, 1998.

[Pea86] W. L. Pearn, A. A. Assad et B. L. Golden. Transforming Arc Routing into Node Routing Problems. *Computers & Operations Research*, 14, 285-288, 1987.

[Pea95a] W. L. Pearn et T. C. Wu. Algorithms for the Rural Postman Problem. *Computers & Operations Research*, 22, 819-828, 1995.

[Pea95b] W. L. Pearn et C. M. Liu. Algorithms for the Chinese Postman Problem on Mixed Networks. *Computers & Operations Research*, 22, 479-489, 1995.

[Pea99] W. L. Pearn et J. B. Chou. Improved solutions for the Chinese Postman Problem on Mixed Networks. *Computers & Operations Research*, 26, 819-827, 1999.

[Rag99] B. Raghavachari et J. Veerasamy. A 3/2-Approximation Algorithm for the Mixed Postman Problem. *SIAM Journal on Discrete Mathematics*, 12, 425-433, 1999.

[Ral93] T. K. Ralphs. On the Mixed Chinese Postman Problem. *Operations Research Letters*, 14, 123-127, 1993.

[Rob90] J. D. Robinson, L. S. Ogawa et S. G. Frickenstein. The Snow-plow Routing Problem. *The UMAP Journal*, 11, 251-259, 1990.

[Roy89] S. Roy et J. - M. Rousseau. The Capacitated Canadian Postman Problem. *INFOR*, 27, 58-73, 1989.

[Shu74] K. A. Shuster et D. A. Schur. Heuristics Routing for Solid Waste Collection Vehicles. U. S. Environmental Protection Agency Publication SW-113, U. S. Government Printing Office, Washington, D. C. , 1974.

[Sin74] E. H. E. Singer. A Method of Fonding Cost-Minimising Routes Incorporating Turn Costs. *Traffic Engineering & Control*, 15, 771-772, 1974

[Sol95] D. Soler. Problemas de Rutas por Arcos con Giros Prohibidos. Thèse de doctorat, Université de Valence, Espagne, 1995.

[Su92] S. I. Su. The General Postman Problem - Models and Algorithms. Thèse de doctorat, College of Business & Management, University of Maryland, 1992.

[Tuc79] W. B. Tucker et G. M. Clohan. Computer Simulation of Urban Snow Removal. *Transportation Research Board Special Research*, Rapport 185, 293-302, 1979.

[Tuc83] A. C. Tucker et L. D. Bodin. A Model for Municipal Street Sweeping Operations. Chapitre 6 de "Modules in Applied Mathematics". *Discrete and System Models*, W. F. Lucas, F. S. Roberts et R. M. Thrall, éd., 3, 76-111, 1983.

[van51] T. van Aardenne-Ehrenfest et N.G. de Bruijn. Circuits and Trees in Oriented Linear Graphs. *Simon Stevin*, 28, 203-217, 1951.

[Vee99] J. Veerasamy. Approximation Algorithms for Postman Problems. Thèse de doctorat, University of Texas, Dallas, 1999.

[Wan95] H.-F. Wang. Insights into a Mixed Chinese Postman Problem. *Journal of the Chinese Institute of Engineers*, 18 (5), 683-689, 1995.

[Wat63] J. A. Wattleworth et P. W. Shuldiner. Analytical Methods in Transportation: Left Turn Penalties in Traffic Assignment Models. *Journal of the Engeneering Mechanics Division, Proceedings of the American Society of Civil Engineers*, vol. 89, Part 1, No. EM6, 97-126, 1963.

[Woo69] D. L. Woods, J. T. Brudeseth et V. G. Stover. Turn Penalty Effects on Minimum Time Paths. *ASCE Journal of Transportation Engineering*, 659-666, 1969.

[Yan98] H. Yan et G. L. Thompson. Finding Postal Carrier Walk Paths in Mixed Graphs. *Computational Optimization and Applications*, 9, 229-247, 1998.