Thèse
de doctorat
de l'UTT

Julien MICHALLET

Problèmes de tournées de véhicules périodiques avec contraintes de sécurité ou de qualité de service

Spécialité :
Optimisation et Sûreté des Systèmes

2013TROY0023 **Année 2013**

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Julien MICHALLET

le 15 novembre 2013

Problèmes de tournées de véhicules périodiques avec contraintes de sécurité ou de qualité de service

JURY

M. B. GRABOT	PROFESSEUR DES UNIVERSITES	Président
M. L. AMODEO	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. D. FEILLET	PROFESSEUR ENSM SAINT ETIENNE	Rapporteur
Mme S. U. NGUEVEU	MAITRE DE CONFERENCES	Examinateur
M. C. PRINS	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. M. SEVAUX	PROFESSEUR DES UNIVERSITES	Rapporteur
M. F. YALAOUI	PROFESSEUR DES UNIVERSITES	Examinateur

Personnalité invitée

M. G. VITRY	INGENIEUR

À Manon, ma fiancée

Remerciements

Je tiens en premier lieu à remercier les personnes qui m'ont accompagnées tout au long de cette thèse à l'UTT : mes directeurs de thèse Lionel Amodeo et Christian Prins mais également Farouk Yalaoui qui s'est joint au projet. Merci à eux pour tout le temps qu'ils ont pu me consacrer et tous les conseils qu'ils m'ont donnés. J'ai appris bien plus que je ne l'avais espéré au cours de ces années.

Je remercie également Grégoire Vitry de m'avoir fait confiance pour entreprendre cette thèse CIFRE chez Nexxtep Technologies. Le versant industriel de ce projet, au contact des clients et de leurs contraintes, fut une incitation au pragmatisme et à l'humilité.

Je voudrais ensuite remercier chaleureusement les membres de mon jury. Merci donc à Dominique Feillet et à Marc Sevaux pour l'honneur qu'ils me font d'être les rapporteurs de cette thèse, pour la relecture avisée qu'ils ont faite du manuscrit et leur venue à Troyes. Merci aussi à Bernard Grabot, président du jury, et à Sandra Ulrich Ngueveu, une ancienne de la maison, d'avoir accepté d'examiner ces travaux.

Une thèse, c'est également une vie de bureau. C'est pourquoi je voudrais remercier tout particulièrement Atefeh, Slim et Thibaut. Nos échanges pas toujours très scientifiques, nos petits déjeuners, notre plantation de tomates, le kashk bademjan et autres koukous ont contribué à rendre ces années en G202 particulièrement agréables. Je n'oublie pas non plus les autres doctorants : nos voisins Julie & Yassine, Matthieu, David, Andres, William, Julien, Valéria, Sona, Elyn, Juan Carlos, Guillermo, Birome et Mustapha.

Une pensée également pour l'équipe franco-brésilienne avec laquelle Thibaut et moi avons participé au Challenge organisé par la Roadef et Google : Renaud et Hugues à Nantes, et, de l'autre coté de l'Atlantique, Puca, Anand et Vinicius. J'ai beaucoup appris au cours de cette année d'aventure qui a été véritablement passionnante.

Un grand merci à Andréa grâce à qui j'ai découvert les joies de l'enseignement. Ce fut une expérience très enrichissante à laquelle je crois bien avoir pris goût...

Merci aussi à tous les personnes que j'ai rencontré au LOSI : Caroline, Nacima, Alice, Murat,

Christophe, Matthieu, Faicel, Hicham, Fred et Haoxun.

J'adresse bien sur de vifs remerciements à Véro, Marijo et maintenant Bernadette qui, chaque jour, répondent à un nombre hallucinant de requêtes et résolvent des problèmes combinatoires impossibles pour nous assurer des conditions de travail optimales.

Chez Nexxtep Technologies, je tiens à remercier tout particulièrement Matthieu sans qui le logiciel d'optimisation de tournées ne serait pas ce qu'il est. Merci aussi à tout le reste de l'équipe : Olivier, Gauthier, Julien, les deux Thomas, Kévin, Jonathan et Florian, Axel, Denis, Michaël, les Marjorie's, Sophie ainsi que tous les stagiaires que j'ai pu côtoyer.

Je voudrais également remercier la famille Revert, pour son soutien et pour m'avoir si souvent accueilli à Troyes lorsque je me rendais au laboratoire.

Je remercie mes proches qui m'ont toujours encouragé : mes parents qui m'ont fait l'immense joie d'être présents le jour de la soutenance, mes quatre frères et sœurs et tous les membres de ma famille.

Enfin, je ne peux terminer sans exprimer toute ma reconnaissance à ma future femme, Manon. Merci ma chérie pour ta patience, ton indéfectible soutien et l'amour que tu me témoignes chaque jour. Ta présence à mes cotés m'est inestimable, ce travail t'est dédié.

Résumé

Cette thèse aborde le problème de tournées de véhicules périodiques (PVRP) lorsqu'il est appliqué au transport de marchandises convoitables. Des contraintes spécifiques relatives à la sécurité du convoi doivent être définies.

Le problème de tournées de véhicules périodiques avec dispersion des instants de service (PVRPTS) est alors décrit puis modélisé mathématiquement. Le but est de servir un ensemble de clients sur plusieurs jours en respectant un degré de variation défini dans les heures de service. Le modèle obtenu est discuté et deux heuristiques constructives sont proposées et évaluées pour sa résolution.

Une recherche locale itérée avec redémarrages (MS-ILS) est proposée pour ce problème. Les résultats obtenus montrent que cette méthode surpasse les deux précédentes sur toutes les instances de test. Elle est ensuite évaluée sur un problème plus classique de la littérature : le problème de tournées de véhicules avec fenêtres horaires souples (VRPSTW) et s'avère très compétitive, produisant de nouvelles meilleures solutions.

La MS-ILS est ensuite transposée au problème de tournées de véhicules régulières (ConVRP). Contrairement au PVRPTS, il s'agit dans le ConVRP de servir régulièrement des clients aux demandes intermittentes. La méthode montre une flexibilité remarquable et produit de bons résultats.

Pour finir, les développements effectués chez Nexxtep Technologies sont présentés. Ils comprennent la conception d'un logiciel commercial pour l'optimisation de tournées de véhicules et l'implémentation des méthodes développées.

Abstract

This thesis is dedicated to the periodic vehicle routing problem when applied to the transportation of valuable goods. Specifics constraints have to be defined to ensure the security of the convoy.

The periodic vehicle routing problem with time spread constraints on services (PVRPTS) is defined and a mathematical model is given. The goal is to serve a set of customers over several days such that their visit times differ by a minimum amount. The depicted model is discussed and two constructive heuristics are designed and assessed.

A Multi-start iterated local search (MS-ILS) is proposed to solve this problem. The results shows that the method outperform the two previous heuristics. For the sake of comparison with previous approach, the MS-ILS is evaluated on a more classical problem : the vehicle routing problem with soft time windows (VRPSTW). The method proves very competitive, producing new best known solutions.

The MS-ILS is then adapted to solve the consistent vehicle routing problem (ConVRP). Unlike the PVRPTS, the goal of the ConVRP is to deliver customers with intermittent demand with regularity in terms of service times and drivers. The method demonstrate his flexibility and produce good results.

Finally, the developments performed at Nexxtep Technologies are depicted. They encompass commercial-software design and implementation of the the proposed methods.

Table des matières

Remerciements **5**

Résumé **7**

Abstract **9**

Introduction générale **21**

1 Présentation du problème industriel **25**

1.1 Introduction 25

1.2 Problème industriel 25

1.3 Données du problème 26

1.3.1 L'entreprise de convoyage 27

1.3.2 Nature des fonds transportés 27

1.3.3 Les clients 28

1.3.4 Les véhicules 28

1.3.5 Le personnel 29

1.3.6 Le réseau routier 29

1.4 Contraintes 29

1.4.1 Contraintes de ressources 29

1.4.2 Contraintes liées au service client 30

1.4.3 Contraintes liées au réseau 31

1.4.4 Contraintes de sécurité 32

1.5 Les objectifs d'optimisation 33

1.6 Synthèse 33

2 État de l'art des problèmes de tournées de véhicules sur nœuds 35

2.1 Introduction 35

2.2 Modèles mathématiques 36

2.2.1 Le problème du voyageur de commerce 36

2.2.2 Le problème de tournées de véhicules 37

2.2.3 Le problème de tournées de véhicules avec flotte hétérogène 39

2.2.4 Le problème de tournées de véhicules avec retours 39

2.2.5 Le problème de tournées de véhicules avec retours mixés 41

2.2.6 Le problème de tournées de véhicules avec prises et déposes simultanées 41

2.2.7 Le problème de collectes et déposes 42

2.2.8 Le problème de tournées de véhicules avec fenêtres horaires 43

2.2.9 Le problème de tournées de véhicules avec fenêtres horaires et prises et déposes simultanées et problèmes riches 44

2.2.10 Le problème de tournées de véhicules avec gestion de stock 44

2.2.11 Les problèmes de tournées de véhicules multi-périodes 46

2.2.12 Les problèmes m-péripatétiques 46

2.3 Méthodes de résolution exactes 51

2.3.1 Énumération complète 51

2.3.2 Programmation dynamique 52

2.3.3 Algorithme par séparation et évaluation 53

2.3.4 Algorithme de branchement et coupes 53

2.3.5 Génération de colonnes 54

2.3.6 Relaxation lagrangienne 55

2.4 Méthodes de résolution heuristiques 56

2.4.1 Heuristiques constructives 57

2.4.2 Les procédures de recherche locale 61

2.4.3 Métaheuristiques 63

2.5 Synthèse 67

3 Problème de tournées de véhicules périodique avec variation des temps de service 69

3.1 Introduction 69

3.2 Description 69

3.3 Modélisation 70

3.4 Problème de détermination des dates de départ de chaque route 73

3.4.1 Description 73

3.4.2 Complexité 74

3.5 Heuristiques constructives 76

3.5.1 Motivations 76

3.5.2 Heuristique de construction séquentielle 76

3.5.3 Heuristique de construction parallèle 79

3.5.4 Expérimentation et résultats 80

3.5.5 Résultats sur les petites instances 81

3.5.6 Résultats sur les grandes instances 83

3.6 Synthèse 88

4 Métaheuristique MS-ILS efficace pour le *PVRPTS* 89

4.1 Introduction 89

4.2 MS-ILS 90

4.2.1 Principes des méthodes ILS et MS-ILS 90

4.2.2 Heuristique randomisée pour le *PVRPTS* 91

4.2.3 Recherche locale pour les problèmes avec fenêtres horaires 92

4.2.4 Perturbation 104

4.3 Évaluations numériques 104

4.3.1 Implémentation et instances 104

4.3.2 Résultats sur les instances de Solomon 105

4.3.3 Problèmes réels 109

4.3.4 Tests sur les instances de VRPSTW 111
4.4 Conclusion 114

5 Transposition au problème de tournées régulières 117

5.1 Introduction 117
5.2 Présentation du problème 117
5.3 Modélisation 119
5.4 Approche proposée 121
5.4.1 Espace de recherche considéré 122
5.4.2 Fonctions de pénalité horaires 124
5.4.3 Pré-calculs de données 125
5.4.4 Heuristique constructive aléatoire 126
5.4.5 Perturbation 127
5.4.6 Recherche Locale 128
5.4.7 Réparation 128
5.4.8 Mise à jour des coefficients de pénalités 129
5.5 Problème de tournées régulières avec décalage des dates de départ des routes 129
5.5.1 Détermination des instants départ pour le *Consistent VRP* 129
5.5.2 Adaptation de la MS-ILS 131
5.6 Expérimentations numériques 132
5.6.1 Jeux de tests utilisés 132
5.6.2 Calibrage des paramètres 132
5.6.3 Résultats sur les instances classiques 134
5.6.4 Résultats sur les instances étendues 135
5.7 Conclusion 136

6 Mise en place des solutions proposées chez Nexxtep Technologies 137

6.1 Introduction 137
6.2 Un premier simulateur 137
6.3 Le logiciel 140

6.3.1 L'architecture 141
6.3.2 Base de données 143
6.3.3 L'interface utilisateur 146
6.4 Synthèse 149

Conclusion générale **151**

Bibliographie **155**

Table des figures

1.1 Tournées des convoyeurs de fonds 27

2.1 Première solution 58
2.2 Ajout de l'arc (1, 3) 58
2.3 Exploration des positions d'insertion pour le client 1 59
2.4 Déroulement de l'heuristique de Gillet et Miller 60
2.5 Application d'un mouvement 2-*opt** 62

3.1 Exemple avec six clients et deux périodes 74
3.2 Représentation du graphe $G = (V', E)$ du problème de coloration et de l'instance construite par la réduction au STP. 76
3.3 Construction de la nouvelle fenêtre horaire du client i pour la période $p + 1$. 77
3.4 Génération des sous-fenêtres de service pour le client i sur 3 périodes 80
3.5 Temps CPU pour le $PVRPTS$ avec 3 périodes et $\epsilon = \lfloor \epsilon_{max} \rfloor$ sur R101 85

4.1 Procédure de découpage des fenêtres horaires pour le client i avec $p = 3$ . . . 93
4.2 Fonction de pénalité de fenêtre horaire au client i 96
4.3 Pénalité d'écart générée par une arrivée à l'instant t_i^r chez le client i dans la route r 96
4.4 Fonctions de pénalité d'écart chez un client i aux périodes 1 et 2. 97
4.5 Fonction de pénalité totale chez le client i dans la route 1. 98
4.6 Parcours avant/arrière et données sur les nœuds le long de la séquence σ . . 98
4.7 Séquences impliquées dans un mouvement de déplacement inter-routes 101
4.8 Concaténations évaluées pour le mouvement de déplacement inter-routes . . 102

4.9 Déplacement dans une même route “en avant” ou “en arrière” 103
4.10 Comportement de *MS-ILS* sur l'instance RD_1 111

5.1 Fonction de pénalité du dépôt 124
5.2 Pénalité induite par l'arrivée chez le client i à la période p 125
5.3 Fonctions de pénalité $U_i^p(t)$ pour le client i aux périodes 1 et 2 126
5.4 Le STPConVRP modélisé avec la méthode MPM 131
5.5 Adaptation de la fonction de pénalisation du dépôt 132

6.1 Vue générale du simulateur 138
6.2 Exemple d'édition d'un client ; ici le client 2. 139
6.4 Instance C101 de Solomon 139
6.3 Résultat obtenu avec l'heuristique de Clarke et Wright 140
6.5 Première version du logiciel utilisant Microsoft MapPoint 141
6.6 Architecture du système 142
6.7 Import d'une liste de clients à partir d'un fichier 143
6.8 Géocodage des clients 144
6.9 Fiche client 145
6.10 Vue principale du logiciel 146
6.11 Création d'une instance 147
6.12 Instance à réaliser 148
6.13 Vérifications préalables 149
6.14 Tournée optimisée 150
6.15 Jeu de tournées irrégulières 150

Liste des tableaux

3.1 Résultats de H1 et H2 avec $n = 5$, $p = 3$ et $\epsilon = \epsilon_{max}$ 82

3.2 Comparaison de H1 et H2 avec CPLEX pour $n = 5$, $p = 3$ et $\epsilon = 0.5 \times \epsilon_{max}$ 84

3.3 Résultats de H1 et H2 avec $n = 50$, $p = 3$ et $\epsilon = 0.5 \times \epsilon_{max}$ 86

3.4 Résultats de H1 et H2 avec $n = 100$, $p = 3$ et $\epsilon = 0.5 \times \epsilon_{max}$ 87

4.1 Résultats pour le jeu R1 avec $n = 5$, $p = 3$ et $\epsilon = \epsilon_{max}$ 106

4.2 MS-ILS, H1, H2 et CPLEX pour $n = 5$, $p = 3$ et $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$ 107

4.3 Ensemble R (23 problèmes), $n = 100$, $p = 3$ et $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$ 108

4.4 Ensemble C (17 problèmes), $n = 100$, $p = 3$ et $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$ 109

4.5 Ensemble RC (16 problèmes), $n = 100$, $p = 3$ et $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$ 110

4.6 Moyenne des indicateurs pour chaque jeu d'instances 110

4.7 Résultats pour les instances réelles 110

4.8 Comparaison sur 21 instances de VRPSTW de type 2 113

4.9 Temps CPU pour les instances de VRPSTW 114

5.1 Résultats de calibration des paramètres N_e et π 133

5.2 Comparaison de MS-ILS avec les approches existantes sur les instances de Christofides modifiées : 5 jours avec une fréquence de livraison de 0.7 134

5.3 Comparaison de MS-ILS avec TALNS sur les instances étendues avec une fréquence de livraison de 0.5, $L = L_{0.4}$ et dates de départ ajustables 136

Introduction générale

C'est au VIe siècle av. J.-C que Sun Tzu, général chinois auteur du plus ancien ouvrage de stratégie militaire connu [118], met en avant la nécessité de disposer de chariots d'approvisionnement de denrées lors des grandes campagnes. Ses écrits mettent clairement l'accent sur le fait que la stratégie de commandement doit permettre de vaincre son adversaire en minimisant le coût induit. Il avait décrit un problème d'optimisation et donné naissance à l'activité de logistique militaire. Très longtemps, la logistique restera une considération militaire et ce n'est qu'après la seconde guerre mondiale, profitant du savoir accumulé et de l'avènement de la Recherche Opérationnelle, que des entreprises de transport "civiles" s'y intéresseront.

En ce début de XXIe siècle, la logistique a pris un essor important, on la retrouve dans les cinq principaux secteurs économiques que sont l'automobile, les biens de consommation, la pharmacie, la grande distribution et l'industrie. Les besoins dans ces secteurs seront de plus en plus importants du fait de la croissance démographique mondiale. Les prix toujours plus hauts des carburants ont incité les entreprises du secteur logistique à optimiser leurs tournées de véhicules. La récente prise de conscience écologique renforce ce besoin, l'optimisation des déplacements entraîne la réduction des émissions pour un plan de transport donné. La raréfaction des ressources en carburant conduit à adopter d'autres technologies de propulsion moins autonomes (électrique), une optimisation de leur utilisation permet alors d'en tirer le meilleur parti. La quête de méthodes de résolution efficaces pour les problèmes d'optimisation du transport a eu un impact majeur sur le développement de la Recherche Opérationnelle. Le problème du voyageur de commerce est en effet l'un des problèmes les plus emblématiques de ceux concernés par cette discipline scientifique à part entière.

Pour traiter ces problèmes, il faut les modéliser sous forme mathématique. La méthode de résolution employée dépend ensuite de la nature du modèle obtenu. Certains de ces problèmes sont "faciles" à résoudre dans la mesure où l'on connait des algorithmes capables d'en déterminer la meilleure solution "rapidement". Les autres, majoritaires en pratique, sont difficiles car on ne connait pas d'algorithme capable d'obtenir la meilleure solution dans

un temps satisfaisant. La taille des problèmes qui peuvent être résolus en pratique par les méthodes dites "exactes" est alors limitée. Des méthodes dites "heuristiques", permettant de trouver rapidement des résultats de très bonne qualité, sont le plus souvent utilisées pour les traitement de cas réels. La garantie d'optimalité est en quelque sorte sacrifiée pour obtenir une solution dans un temps raisonnable. La caractérisation entre problèmes "faciles" et "difficiles" est faite, dans le cas des problèmes d'optimisation, à partir du problème de décision associé. Un problème de décision est une question à laquelle on cherche à répondre par "oui" ou par "non". Un problème d'optimisation vise à trouver la meilleure solution, typiquement le problème de décision associé consiste à savoir si étant donnée une solution il en existe une meilleure. Les problèmes de décisions pour lesquels il est facile de vérifier la validité d'une solution appartiennent à une classe nommée NP (Non-deterministic Polynomial). Ceux dont la solution est facile à trouver appartiennent à la classe P (Polynomial). On sait que P est inclus dans NP : il est facile de vérifier ce qu'il est facile de trouver. Par contre, on ne sait toujours pas si NP est inclus ou non dans P : ce qui est facile à vérifier est-il également facile à trouver ? Il existe en effet des problèmes de décision de NP pour lesquels on ne connait pas d'algorithme de résolution "rapide" (ils ne semblent pas être dans P). On dit de ces problèmes de décision qu'ils sont NP-complets.

La résolution du problème "P=NP" est mise à prix un million de dollars par le Clay Mathematics Institute qui le considère comme un des sept problèmes du millénaire (il n'en reste en fait plus que six, la conjecture de Poincaré a été démontrée en 2003 par Grigori Perelman). La réponse à cette question est d'une importance capitale pour l'informatique théorique et pour l'humanité. En effet, s'il s'avérait que $P = NP$, tous les problèmes connus aujourd'hui comme NP pourraient être résolus "rapidement". La sécurité des transactions bancaires serait alors remise en question. La méthode de chiffrement RSA est en effet basée sur le fait que la factorisation d'un entier en nombre premiers est dans NP mais ne serait pas dans P. Concrètement, si un pirate veut décoder un message, il doit être capable de factoriser un entier en un produit de nombres premiers. Aujourd'hui, personne ne sait le faire pour des entiers de plus de 200 chiffres. Les enjeux liés à la résolution de ce problème inspirent jusqu'au cinéma. En effet, un récent thriller américain de Timothy Lanzone, *Traveling Salesman* (2012), explore les conséquences tragiques que pourrait avoir $P = NP$ sur l'humanité. Toutefois, une grande majorité de la communauté scientifique conjecture que $P \neq NP$, rendant peu probable un tel scénario, mais ceci reste toujours à démontrer.

Loin d'essayer de répondre à cette épineuse question, cette thèse s'intéresse en premier lieu au convoyage de fonds. Activité logistique relative au transport d'argent, elle est concernée comme toute autre activité logistique par un besoin d'optimisation. Les contraintes de sécurité inhérentes à la profession ne sont à ce jour ni définies, ni prises en compte par les logiciels

d'optimisation présents sur le marché. La société Nexxtep Technologies est spécialisée dans la conception de systèmes de télémesure et d'aide à la décision. Elle souhaite développer et commercialiser une solution logicielle d'optimisation de tournées qui puisse être utilisée par ses clients convoyeurs de fonds. Cette thèse en convention CIFRE a donc pour but premier de fournir un modèle du problème d'optimisation de tournées pour le convoyage de fonds et d'en proposer la résolution par différentes approches. Les développements algorithmiques doivent par la suite pouvoir être intégrés au logiciel proposé par Nexxtep Technologies.

Le premier chapitre de cette thèse est consacré à la définition de la problématique de tournées des convoyeurs de fonds. Cette première définition est faite manière strictement littéraire et ne comporte aucun terme mathématique. le but est de décrire la réalité du terrain le plus fidèlement possible afin de servir de base de modélisation.

Un état de l'art des modélisations et des méthodes de résolution relatives à ce problème de tournée de véhicule est ensuite donné. La description du premier chapitre et l'état de l'art permettent de proposer un modèle mathématique pour le problème de tournées des convoyeurs de fonds dans le troisième chapitre. Ce modèle est alors discuté et la complexité du problème est étudiée. Deux procédures de résolutions approchées sont proposées et comparée entre elles et avec les solutions optimales d'un solveur commercial sur de petits problèmes. Le quatrième chapitre propose une méthode approchée de type métaheuristique très efficace pour le problème étudié. Pour permettre une comparaison avec la littérature existante, la méthode est ensuite testée sur le problème de tournées de véhicules avec fenêtres horaires souples. Le cinquième chapitre propose d'appliquer la méthode précédente au problème ou l'on cherche cette fois à obtenir des tournées les plus régulières et prévisibles possibles. L'évaluation est faite sur les jeux de tests classiques de la littérature ou la bonne performance de la méthode est démontrée. Pour finir, un sixième et dernier chapitre est consacré à la description des travaux d'implémentation effectués chez Nexxtep Technologies, avant une conclusion générale et des perspectives de travaux futurs.

Chapitre 1

Présentation du problème industriel

1.1 Introduction

Ce chapitre est consacré à la description du problème industriel que rencontre Nexxtep Technologies et qui est à l'origine de cette thèse CIFRE. Le but recherché est ici de relater le plus fidèlement possible la réalité du terrain. Il n'est fait usage d'aucune notation ou équation pour le moment, il s'agit d'identifier les différentes composantes de la problématique. La présentation du problème considéré et ses motivations sont données dans la première partie. Une seconde partie est consacré à l'identification des données du problèmes. Les différentes contraintes sont ensuite inventoriées et détaillées. Pour finir, les objectifs d'optimisation sont présentés et une synthèse est faite. Une définition plus formelle, au moyen d'un modèle mathématique, est donnée dans le chapitre 3 et s'appuie sur cette description.

1.2 Problème industriel

Malgré la dématérialisation des transactions financières rendue possible par les cartes de crédit et plus récemment par les systèmes de paiement en ligne, les espèces restent encore largement utilisées. Retirer de l'argent sur un automate bancaire est devenu un geste banal tant ces DAB/GAB (Distributeur Automatique de Billets/Guichet Automatique Bancaire) sont présents autour de nous. La France compte aujourd'hui un automate bancaire pour 1000 personnes [1] et leur nombre est en constante augmentation : ils étaient 47 800 en France en 2006 contre 39 000 en 2002 selon le CECEI (Comité des Établissements de Crédit et des Entreprises d'Investissement) [15] soit une augmentation de 22,5% en 4 ans. Outre les particuliers, les commerçants ont également pris l'habitude de venir déposer leur recette du

jour en sécurité dans les GAB qui assurent en plus la fonction d'automates de dépôt. L'approvisionnement/délestage de ces automates est confié par les banques à des entreprises dites de convoyage de fonds. Ces entreprises sont spécialisées dans le transport d'argent. Elles ont pour clients les banques mais également les supermarchés de la grande distribution et certains commerçants. Leur mission est d'assurer le réapprovisionnement/délestage de leurs clients en devises. Le convoyage de fonds est donc une activité de transport logistique où le bien transporté est l'argent. Comme tout transporteur, les entreprises de convoyage de fonds sont soumises au problème de l'organisation de leurs tournées de service. Outre les contraintes rencontrées par les transporteurs classiques, la nature même des biens transportés fait émerger ici une problématique de sécurité jusqu'alors peu abordée. Nexxtep Technologies est une entreprise spécialisée dans le développement de solutions pour les systèmes embarqués, la géolocalisation et l'optimisation et possède une forte expertise du monde de la sécurité. Elle souhaite développer un logiciel qui puisse permettre à ses clients convoyeurs d'optimiser leurs activités de transport de marchandises de valeurs et notamment de fonds. Nous nous attachons dans les sections qui suivent à recenser les données du problème, les différentes contraintes auxquelles sont soumises les tournées de convoyeurs de fonds et le/les critère(s) selon le(s)quel(s) de telles tournées peuvent être évaluées.

1.3 Données du problème

Cette section a pour but d'identifier les données du problème et de faire apparaitre les différents liens qui peuvent exister entre elles. L'activité de l'entreprise de convoyage de fonds est celle de délester et/ou de réapprovisionner ses clients en devises au moyen de personnel et de véhicules à travers un réseau routier. L'activité fait donc intervenir les entités suivantes :

- L'entreprise de convoyage,
- Les clients,
- Les fonds,
- Le personnel,
- Les véhicules,
- Le réseau routier.

La suite de cette section détaille plus précisément chacune des données qui sont listées ici.

1.3.1 L'entreprise de convoyage

L'entreprise de convoyage de fond possède habituellement un (ou plusieurs) centre fort à partir duquel (desquels) elle organise son activité de transport. Le personnel et les véhicules utilisés pour les opérations sont basés au centre fort. C'est également dans le centre fort que les fonds collectés sont comptés et conditionnés avant d'être ré-acheminés. Un véhicule effectue son service comme suit : il quitte le centre fort, effectue un certain nombre d' "arrêts" puis retourne à son centre fort d'origine. Nous avons donc ici affaire à un problème de tournées sur nœuds.

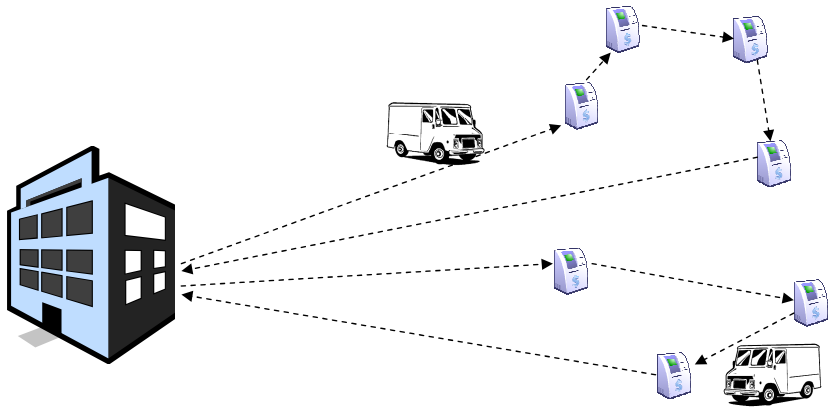


FIGURE *1.1: Tournées des convoyeurs de fonds*

1.3.2 Nature des fonds transportés

La marchandise, qui se trouve être de l'argent dans le problème étudié, peut se présenter sous différentes formes :

- Des sacs de billets
- Des sacs de pièces
- Des enveloppes de dépôt.
- Des cassettes de transport sécurisées.

Les deux premiers types de marchandises listés ici sont les plus évidents et correspondent bien à l'idée que l'on peut se faire du transport de fond. Les enveloppes de dépôt contenant des billets sont celles déposées dans les automates de dépôt par les commerçants. Ces automates de dépôt peuvent se trouver à l'extérieur (galerie marchande, sur rue...) mais également

chez le commerçant. Une fois déposé dans l'automate, l'argent est sous la responsabilité des convoyeurs de fonds qui doivent le récupérer. Les cassettes de transport sécurisées, sont apparues relativement récemment pour répondre en partie à la problématique de sécurité de la profession. Il s'agit de cassettes dans lesquelles sont transportés les billets déclenchant leur destruction par explosion ou maculation à l'encre indélébile en cas de tentative d'ouverture frauduleuse. Ces cassettes sont transportées dans des racks spéciaux à l'intérieur du véhicule et la destruction de leur contenu est également prévue si la procédure d'extraction n'est pas respectée. Il existe plusieurs fournisseurs de cassettes qui ont chacun leurs racks spécifiques.

1.3.3 Les clients

L'entreprise de convoyage de fonds exerce son activité pour des clients en nombre défini dispersés géographiquement dans la zone de service qu'elle couvre. Comme il a été précisé en section 1.2, les clients des convoyeurs de fonds ne sont pas uniquement des banques mais peuvent également être des commerces de plus ou moins grande taille. Chacun de ces clients utilise un ou plusieurs types de marchandises. En effet, un client peut utiliser des billets, des pièces de monnaie ou encore des cassettes sécurisées. Un client dispose à chaque demande d'une certaine quantité de marchandise à collecter ou à recevoir. Ce service doit être effectué en respectant les horaires d'ouverture et de fermeture qui ont été convenues avec lui. Chaque client possède ses propres modalités d'accès (difficulté d'accès, temps d'attente éventuel...) qui influent sur le temps opératoire.

1.3.4 Les véhicules

Chaque entreprise de convois de fonds possède un nombre défini de véhicules qui sont en général de deux types :

- Fourgon blindé,
- Véhicule blindé léger banalisé.

Le deuxième type de véhicule est une alternative plus discrète au fourgon blindé et se trouve être de plus en plus demandé par les clients. Le service est effectué incognito au moyen d'une voiture blindée banalisée qui se fond parfaitement dans le paysage urbain. Ce mode de service a l'avantage de ne pas créer de situation anxiogène aux abords du point de desserte et s'avère beaucoup plus difficile à identifier par d'éventuels assaillants.

1.3.5 Le personnel

Une entreprise de convois de fonds emploie un personnel en nombre défini qui doit respecter une durée légale de travail. Chacun des membres de ce personnel possède une qualification et un niveau de compétence qui lui est propre.

1.3.6 Le réseau routier

Pour livrer ses clients, l'entreprise de convoyage de fonds devra emprunter avec ses véhicules le réseau routier. Ce réseau définit les temps et distances de parcours entre chaque client. Ce réseau est vivant, soumis aux aléas de la circulation.

1.4 Contraintes

Cette section est destinée à identifier les différentes contraintes auxquelles est soumise l'activité de transport de fonds. Il est possible de rassembler ces contraintes dans les catégories suivantes :

- Les contraintes de ressources,
- Les contraintes liées au service client,
- Les contraintes liées au réseau,
- Les contraintes de sécurité.

Nous explicitons ces ensembles de contraintes dans la suite de cette section.

1.4.1 Contraintes de ressources

Les contraintes de ressources sont les contraintes dont l'origine est imputable aux ressources exploitées par l'entreprise pour mener à bien son activité. Dans le cas qui nous intéresse, l'ensemble de ces ressources est constitué du personnel et des véhicules. S'agissant du personnel, nous rencontrons les contraintes suivantes :

- Le personnel est en nombre limité,
- Le temps de travail des membres du personnel est limité.

Les véhicules utilisés impliquent les contraintes suivantes :

- La flotte de véhicules est en nombre limité,
- Chaque véhicule a une capacité monétaire limitée,

- Chaque véhicule est rattaché à un centre fort auquel il doit retourner une fois son service terminé.

Chaque véhicule comporte un chauffeur et une équipe de convoyeurs en nombre défini. Nous admettons dans ce manuscrit, pour nous limiter au problème de transport, qu'il est toujours possible de trouver une affectation du personnel aux véhicules disponibles. La limite considérée est alors la taille de la flotte de véhicules. Le temps de travail du personnel est limité et cette limite est matérialisée par une fenêtre horaire de service pour le centre fort. Les repas ne sont pas pris au cours de la tournée, on imagine en effet mal une équipe de convoyeurs garer un fourgon rempli de devises pour se rendre tranquillement dans un restaurant. Les tournées sont donc organisées en "tournées du matin" et "tournées de l'après midi" qui sont organisées séparément. La contrainte de capacité monétaire est tout particulièrement spécifique à l'activité concernée. Il s'agit plus précisément du respect des montants assurables à l'intérieur des véhicules. Ce montant dépend du type de véhicule pris en compte. Il apparait de plus d'après les échanges que nous avons pu avoir avec des entreprises opérant dans le transport de fonds, que la capacité monétaire est prégnante sur la capacité "physique". La marchandise transportée n'est pas volumineuse au regard de sa valeur fiduciaire. Pour finir, le fait que chaque véhicule soit attaché à un dépôt matérialise ici le fait qu'une tournée débute à un centre fort et se termine à celui-ci. Dans le cas qui nous intéresse, chaque centre fort couvre un secteur bien défini et connu de son personnel. Un client est donc affecté à un centre fort en fonction du secteur géographique dans lequel il se trouve. Chaque centre fort effectue ainsi l'organisation de ses tournées indépendamment des autres.

1.4.2 Contraintes liées au service client

Un certain nombre de contraintes sont inhérentes au service chez le client. En voici la liste :

- Le respect des plages horaires de service,
- La disponibilité des automates,
- Synchronisation avec les tournées de dabistes (terme expliqué plus loin),
- Respect du mode de service demandé,
- Adéquation entre les moyens (humain, matériel) et les spécificités du lieu :
  - ⇒ le personnel affecté à une mission doit connaitre un minimum les sites à visiter (serrure qui coince, interrupteur lumière défectueux, contraintes d'accès, marche à suivre en cas d'absence).
  - ⇒ le matériel doit être adapté au site à visiter (largeur/hauteur d'accès, contraintes de charge, ...)

Le client fixe une plage horaire dans laquelle il souhaite que le service soit effectué, ceci lui permet de s'organiser afin que les locaux soient disponibles pour la livraison. Le respect des fenêtres horaires de livraison joue également un rôle important dans le cadre de la sécurité comme indiqué en 1.4.4. Le client souhaite éviter toute rupture de stock dans ses distributeurs ou "trop plein" dans ses automates de dépôt. Il est donc impératif que chaque client prévu pour la journée soit visité. Quand il s'agit d'alimenter des DAB/GAB, la plupart du temps, les convoyeurs déposent l'argent dans un coffre dit de "transfert". Le remplissage effectif de l'automate est effectué par une personne qui passera plus tard que l'on nomme en référence au DAB/GAB un "dabiste". Les dabistes sont soit des employés de l'entreprise de convoyage de fonds, soit des employés de la banque. La bonne synchronisation de la tournée des convoyeurs avec celles des dabistes doit donc être assurée. La nécessité de respecter les plages horaires de livraison s'en trouve donc renforcée. Comme précisé en 1.3.2, le client peut choisir une livraison plus discrète que celle effectuée traditionnellement au moyen d'un fourgon blindé. Un fourgon blindé ne peut effectuer le service requérant un véhicule léger et inversement. Pour que le mode de service soit respecté, les tournées de véhicules légers et de fourgons blindés doivent donc être traités séparément. L'adéquation des ressources employées avec les spécificités du site client est primordial. Afin de centrer notre étude sur l'aspect "routage", il est admis ici que le personnel affecté a connaissance des particularités de chaque site et que les véhicules disposent du matériel adéquat pour effectuer n'importe quel service.

1.4.3 Contraintes liées au réseau

Le client est servi au moyen des ressources via le réseau routier. Ce réseau routier comporte toutefois ses spécificités :

- Les routes du réseau possèdent des vitesses de circulation limitées qui dépendent des véhicules,
- Les routes du réseau ont des longueurs définies et donc des coûts de parcours définis,
- Les routes comportent des sens de circulation.

La connaissance des temps de parcours est une donnée indispensable au respect des fenêtres horaires. Du fait des limitations de vitesse sur les routes, les temps de parcours ne dépendent pas nécessairement de la longueur de celles-ci. Il est donc important d'avoir connaissance à la fois de la longueur des routes mais également de leurs temps de parcours. Les limites de vitesse de circulation spécifiques aux véhicules impliquent par ailleurs qu'un fourgon blindé parcourt un tronçon de voie rapide moins vite qu'un véhicule léger. Les sens de circulation des routes impliquent que la distance et le temps de parcours ne sont pas nécessairement identiques selon que l'on chemine d'un client A vers un client B ou l'inverse (les sens interdits

peuvent amener à faire des détours que l'on ne fait pas à l'aller).

Ajoutons que la prise en compte des données de circulation est primordiale pour pouvoir générer des tournées faisables. Le réseau est en effet sujet à une variabilité que l'on peut décomposer en trois niveaux :

Niveau 1 : La variabilité "saisonnière" qui est redondante et donc facilement prévisible (ex : la rue de Rivoli à Paris dans la journée).

Niveau 2 : La variabilité "événementielle" qui est elle constituée d'événements plus ponctuels (un match de foot ou un marathon...) mais qui sont prévisibles.

Niveau 3 : La variabilité due aux "aléas" qui elle est difficilement prévisible et qui a comme seule parade l'information en temps réel la plus précise possible. Il est donc important de pouvoir être "dynamique" de manière à faire face à ces différents aléas.

Dans la pratique, la prise en compte de ces données est rendue compliquée par la difficulté rencontrée pour en faire la collecte. Dans le cas qui nous intéresse et de manière à simplifier l'utilisation par le décideur, nous proposerons deux réseaux : un pour les tournées effectuées le matin et l'autre pour celles de l'après midi.

1.4.4 Contraintes de sécurité

L'exigence relative à la sécurité des biens transportés engendre des contraintes que nous avons pu identifier au contact d'acteurs du secteur. Celles qui sont apparues comme prégnantes sont listées ici :

- Les fenêtres horaires de livraisons chez le client doivent être respectées,
- Les temps d'attente sont proscrits,
- Les convoyeurs ne doivent pas se trouver en même temps que les dabistes dans les locaux techniques,
- Les services ne doivent pas être prévisibles.

Si le respect des fenêtre horaires de livraison est une contrainte de qualité de service imposée par le client, elle se trouve aussi être une contrainte de sécurité. La livraison d'une marchandise telle que l'argent ne peut avoir lieu en n'importe quelle circonstance. Une agence bancaire peut par exemple préférer que le service ne soit pas effectué durant la période d'ouverture aux clients afin d'éviter de dramatiques dommages collatéraux en cas d'attaque. De même, toute attente dans le processus de livraison n'est pas souhaitable. Un convoi de fonds n'est jamais autant en insécurité que lorsqu'il est arrêté sur la voie publique, il lui est alors plus difficile de s'échapper en cas attaque. Pour cette raison, nous considérons les fenêtres

horaires de livraison client au sens le plus strict ("hard time windows" en anglais). Cette considération permet de ne pas créer de situations où le convoi doit attendre parce qu'il est arrivé trop tôt. L'arrivée inattendue des convoyeurs armés alors que le dabiste se trouve dans les locaux est génératrice de stress et peut potentiellement déclencher une réaction de défense de part et d'autre avec des conséquences regrettables. Il est donc important, pour cette raison également, de respecter scrupuleusement les fenêtres horaires de livraison. Enfin, afin d'éviter de rendre les passages chez le client trop prévisibles, il est important d'éviter toute régularité apparente dans le plan de transport. Une embuscade est en effet plus facile à organiser si l'on connait avec certitude l'heure et le lieu de passage du convoi. Nous devons donc considérer ce problème de livraison sur plusieurs jours. Un niveau de "similarité" entre deux périodes de livraison doit pouvoir être défini et ne pas être dépassé. La définition de cette "similarité" ne doit pas se limiter uniquement à la route empruntée mais prendre également en compte des caractéristiques temporelles afin que le convoi ne se trouve pas toujours au même endroit au même moment.

1.5 Les objectifs d'optimisation

L'objectif premier est la minimisation des coûts engendrés par l'activité de desserte des clients. Ce coût peut être établi en fonction du nombre total de kilomètres parcourus ou encore du temps total opératoire. Un autre critère d'optimisation qui a été défini comme une contrainte dans ce chapitre peut être vu comme un objectif à maximiser : la sécurité. Il ne s'agit plus alors de respecter un niveau de sécurité donné mais de le maximiser. Une optimisation bi-critère est donc envisageable en considérant conjointement le critère de coûts et celui de sécurité. On cherchera alors à minimiser le coût et à maximiser la sécurité (ou minimiser l'insécurité) afin d' obtenir les solutions offrant le meilleur arbitrage possible de ces deux critères.

1.6 Synthèse

Le cas réel présenté dans ce chapitre porte sur le service de clients en valeur monétaires sur plusieurs jours en respectant un ensemble de contraintes induites par les ressources utilisées, le service client, le réseau emprunté et finalement le niveau de sécurité désiré. Les critères d'évaluation d'une solution peuvent être le coût de l'activité, le niveau de sécurité produit ou les deux simultanément. Pour optimiser l'activité ainsi décrite, il faut pouvoir la modéliser mathématiquement. Le chapitre suivant propose un état de l'art des principales

problématiques de tournées de véhicules sur nœuds, leurs modélisations mathématiques ainsi que les méthodes de résolutions existantes. L'analyse conduite ici, associée à cet état de l'art, permet d'établir le modèle mathématique adéquat pour traiter le problème réel dans le chapitre 3.

Chapitre 2

État de l'art des problèmes de tournées de véhicules sur nœuds

2.1 Introduction

Le problème de tournées de convoyeurs de fonds se présente de la manière suivante : chaque jour, plusieurs véhicules quittent le centre fort pour aller alimenter ou vider les automates de retrait/dépôt de leurs clients et retournent au dépôt. Ces opérations se font en respectant un certain nombre de contraintes et notamment celles de sécurité. Le problème à traiter est donc une version enrichie du célèbre problème de tournées de véhicules (VRP, pour Vehicle Routing Problem en anglais) proposé par Dantzig et Ramser [21]. A l'époque, les auteurs décrivent un problème bien réel : les tournées d'approvisionnement en essence des stations-service. Ils proposent alors la première formulation mathématique de ce problème maintenant largement étudié. Le VRP est un problème NP-difficile, il généralise le problème du voyageur de commerce (TSP, pour Travelling Salesman Problem en anglais) prouvé NP-difficile (preuve dans le livre de Garey et Johnson [44]). Ce chapitre est divisé en trois parties. La première est consacrée à la description des modèles mathématiques du TSP et du VRP. Les variantes les plus courantes sont décrites puis l'accent est mis sur celles relatives au problème de tournées de convoyeurs de fonds. Les principales méthodes de résolution exactes, permettant de résoudre à l'optimalité de tels problèmes, sont décrites dans une deuxième partie. La troisième partie de ce chapitre est consacrée aux méthodes "heuristiques", permettant d'obtenir une solution plus rapidement mais sans garantie d'optimalité. Une synthèse des modèles et méthodes est finalement proposée et donne l'orientation du chapitre suivant, dédié à la modélisation du problème de tournées de convoyeurs de fonds.

2.2 Modèles mathématiques

2.2.1 Le problème du voyageur de commerce

Le problème du voyageur de commerce (TSP) a été décrit pour la première fois par Dantzig et Ramser [21] en 1959. Dans ce problème, on imagine un représentant de commerce qui doit rendre visite à plusieurs de ses clients, chacun situé dans une ville différente. Le but est alors de trouver le tour visitant toutes les villes et qui minimise la distance parcourue par le représentant. Ce problème revient à trouver un cycle hamiltonien de coût minimal dans un graphe complet. Voici la modélisation de ce problème proposée par Dantzig et Ramser [21] : Soit $G = (V, A)$ un graphe orienté complet sans boucles où $V = \{1, 2, i, j \ldots, n\}$ l'ensemble des sommets représente les clients à visiter et A l'ensemble des arcs représente les routes entre ces clients. On associe alors à chaque arc (i, j) de A un coût c_{ij} positif et on utilise les $(n-1)^2$ variables binaires x_{ij} telles que :

$$x_{ij} = \begin{cases} 1 \text{ si l'arc } (i,j) \text{ est utilisé.} \\ 0 \text{ sinon.} \end{cases}$$

Il est alors possible d'écrire ce problème sous la forme d'un programme linéaire en nombres entiers :

$$\min z = \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{2.1}$$

Sous les contraintes :

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in V, \tag{2.2}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall i \in V, \tag{2.3}$$

$$\sum_{i,j\in S} x_{i,j} \leq |S| - 1 \qquad \forall S \subset V; 2 \leq |S| \leq n-1, \tag{2.4}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall (i,j) \in A; i \neq j. \tag{2.5}$$

Il s'agit donc comme l'exprime la fonction objectif (2.1) de minimiser la somme des coûts engendrés par l'utilisation des arcs. Les contraintes (2.2) et (2.3) sont des contraintes de flots qui permettent d'établir qu'un et un seul arc arrive sur chaque sommet (2.2) et qu'un et un seul arc en reparte (2.3). La contrainte (2.4) est quant à elle utilisée pour interdire la formation de sous-tours dans la solution. En effet, sans cette contrainte, une solution peut

prendre la forme de "bulles" qui ne sont reliées ni entre elles, ni au dépôt. On impose alors ici que le nombre d'arcs utilisés à l'intérieur de n'importe quel sous-ensemble d'au moins deux sommets soit au plus égal au nombre de sommets composant cet ensemble moins un. Les sous-tours sont ainsi brisés et la connectivité de la solution est assurée. En pratique, cet ensemble de contraintes est difficile à traiter du fait de sa cardinalité. Il y a en effet $2^{n-1}-2$ sous-ensembles S à vérifier pour un problème de taille $|V| = n$ et donc autant de contraintes.

Il existe un jeu de contraintes dont la cardinalité est en $n^2 - 3n + 2$ qui a été proposé par Miller et al. [82] :

$$u_i - u_j + (n-1)x_{ij} \leq n-2 \qquad \forall i, j \in V \setminus \{1\}\,; i \neq j, \tag{2.6}$$

$$1 \leq u_i \leq n-1 \qquad \forall i \in V \setminus \{1\}\,. \tag{2.7}$$

Les variables u_i représentent ici le rang (l'ordre) du client i dans le parcours. La contrainte (2.6) est violée si, dans une solution, un sommet est relié à un autre de rang inférieur. La formation de sous-tours devient donc impossible. Les variables additionnelles u_i rendent toutefois la relaxation continue du programme linéaire plus faible que la formulation avec (2.4), ces contraintes sont donc moins efficaces en pratique. En général, les contraintes de sous-tours sont dans un premier temps relâchées et le problème résolu. Des algorithmes de détection de sous-tours permettent ensuite d'ajouter seulement les coupes nécessaires et de relancer la résolution. Cette phase d'ajout de coupes/résolution est répétée tant que la solution proposée présente des sous-tours.

2.2.2 Le problème de tournées de véhicules

Le problème de tournées de véhicules (VRP pour Vehicle Routing Problem) est une extension du TSP dans laquelle K véhicules de capacité W doivent à partir d'un dépôt commun servir un ensemble de n clients. Plusieurs modèles sont possibles pour décrire ce problème et dépendent de la méthode de résolution qui va être employée. Un état de l'art de ces modélisations est dressé dans le livre de Toth et Vigo [117]. La formulation à trois indices présentée ici a été adaptée de celle du TSP par Fisher et Jaikumar [39] et présente l'avantage d'être particulièrement flexible. Ce problème peut être représenté dans un graphe orienté complet $G = (V, A)$ où $V = \{0, 1, \ldots, i, \ldots, n\}$ est l'ensemble des sommets et A l'ensemble des arcs (i, j) de ce graphe. Le sommet 0 du graphe représente le dépôt et chaque sommet i est un client à qui il faut livrer une quantité de marchandise q_i. Chacun des arcs (i, j) possède un poids c_{ij} qui représente le coût de déplacement du client i au client j. Cette modélisation nécessite alors n^2K variables de décision binaires x_{ij}^k telles que :

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

Puis nK variables de décision y_i^k telles que :

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

$$\min z = \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \tag{2.8}$$

Sous les contraintes :

$$\sum_{i \in V} q_i y_i^k \leq W \qquad \forall k \in K \tag{2.9}$$

$$\sum_{k \in K} y_i^k = 1 \qquad \forall i \in V \backslash \{0\} \tag{2.10}$$

$$\sum_{k \in K} y_0^k = K \tag{2.11}$$

$$\sum_{i \in V} x_{ij}^k = y_j^k \qquad \forall j \in V \backslash \{1\}, \forall k \in K \tag{2.12}$$

$$\sum_{j \in V} x_{ij}^k = y_i^k \qquad \forall i \in V \backslash \{1\}, \forall k \in K \tag{2.13}$$

$$\sum_{i,j \in S} x_{ij}^k \leq |S| - 1 \qquad \forall S \subset V; 2 \leq |S| \leq n-1, \forall k \in K \tag{2.14}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i,j \in V \mid i \neq j; \forall k \in K \tag{2.15}$$

$$y_i^k \in \{0,1\} \qquad \forall i \in V; \forall k \in K \tag{2.16}$$

Il s'agit là de minimiser le coût lié à l'utilisation des arcs (2.8). La présence des contraintes (2.9) assurent que chaque camion ne dépasse pas la capacité maximum dans sa tournée, on parle alors de problème de tournées de véhicules sous contraintes de capacité (CVRP pour Capacited Vehicle Routing Problem). Le jeu de contraintes (2.10) impose qu'un et un seul véhicule desserve le client i, (2.11) permettent quant à elles de s'assurer que K et seulement K véhicules sont utilisés. Les contraintes (2.12) et (2.13) garantissent que le véhicule qui sert un client est bien celui qui en repart, ce sont les contraintes dites de "flots" . Enfin, (2.14) permettent l'élimination des sous- tours. De la même manière que pour le TSP vu

précédemment, il existe un jeu de contraintes de cardinalité moindre qui est la généralisation de (2.6) et (2.7) de Miller et al. [82] pour le VRP :

$$u_i^k - u_j^k + W x_{ij}^k \leq W - q_i \qquad i, j \in V \setminus \{0\}\,; i \neq j; q_i + q_j \leq W; k \in K \tag{2.17}$$

$$q_i \leq u_i^k \leq W \qquad i \in V \setminus \{0\}\,; k \in K. \tag{2.18}$$

L'ajout de ce jeu de contraintes rend d'ailleurs obsolète celles de capacité (2.9). Malheureusement, ces contraintes de sous-tours, même si elles semblent plus simples et offrent une cardinalité moindre, fournissent une relaxation continue de mauvaise qualité. Le jeu de contraintes (2.14) est donc plus largement utilisé.

2.2.3 Le problème de tournées de véhicules avec flotte hétérogène

Il n'est pas rare, en pratique, que les véhicules utilisés pour le service de clients soient de types différents. Une prise en compte de la spécificité de chaque véhicule est nécessaire en termes de capacité de chargement, temps de parcours, coûts de transport...etc. Toutefois, la version la plus étudiée de ce problème dans la littérature considère seulement le cas des capacités variables. Le problème est appelé problème de tournées de véhicules avec flotte hétérogène (HVRP pour Heterogeneous Vehicle Routing Problem) et fait l'objet d'un vif intérêt comme récemment par Prins [101].

2.2.4 Le problème de tournées de véhicules avec retours

Dans beaucoup de problèmes de tournées réels, les véhicules doivent alimenter les clients mais également les délester. Cette problématique de logistique inverse a été traitée dès 1984 par Deif et Bodin [26] qui adaptent l'heuristique des "savings" décrite par Clarke et Wright [18] pour le VRP classique. Cependant, dans la version du problème traité, les auteurs considèrent qu'un client peut être soit livré en marchandise, soit délesté mais pas les deux simultanément. De plus, dans chaque tournée, les clients sont servis dans un ordre bien défini : ceux à livrer d'abord puis, une fois que toutes les livraisons sont terminées, ceux à collecter. La résolution proposée consiste en l'introduction d'un terme de pénalité qui force les clients chez qui l'on doit récupérer de la marchandise à la fin de la route de chaque véhicule. Ce problème prend le nom de problème de tournées de véhicules avec retours (VRPB pour Vehicle Routing Problem with Backhauls), terme qui nous vient de Goetschalckx et Jacob-Blecha [54]. Une version dans laquelle les clients servis (linehaul customers) et les clients délestés (backhaul

customers) peuvent être visités dans un ordre quelconque est proposée par Golden et al. [55] et une heuristique d'insertion proposée. Il s'agit alors d'un VRPMB (Vehicle Routing Problem with Mixed Backhauls), le VRPB étant en fait par extension un VRPCB (Vehicle Routing Problem with Clustered Backhauls). L'approche "delivery-first, pickup-second" du VRPCB (VRPB) initialement proposée par Deif et Bodin a longtemps été majoritaire dans les articles traitant le problème de la logistique de retour. Cette préférence des auteurs vient de la difficulté rencontrée en pratique pour ré-agencer le chargement d'un véhicule au cours de sa tournée. Ainsi, il est plus aisé de considérer que les camions partent pleins du dépôt, se vident chez les clients demandant une livraison puis, une fois vide, visitent les clients chez qui ils doivent ramasser. Ceci conduit les auteurs à séparer l'ensemble des clients $i \in V \setminus \{1\}$ en deux sous-ensembles P et D où P est l'ensemble des clients chez lesquels il doit y avoir enlèvement (Pick-up) de marchandise et D celui des clients qui doivent être livrés. Comme les clients de D sont visités avant les clients de P, la contrainte de capacité du CVRP (2.9) doit être séparée en deux parties :

$$\sum_{i \in P} q_i y_i^k \leq W \qquad \forall k \in K \tag{2.19}$$

$$\sum_{i \in D} |q_i| \, y_i^k \leq W \qquad \forall k \in K \tag{2.20}$$

On notera ici que la quantité q_i désigne une quantité "à prendre" pour les clients de l'ensemble P alors qu'elle désigne une quantité "à livrer" pour les clients de l'ensemble D. Une quantité "à livrer" a pour effet de faire diminuer la charge W du véhicule, elle est par conséquent considérée comme négative. Une nouvelle définition de q_i peut être donnée qui facilitera l'écriture des modèles qui suivront :

$$q_i = q_i^+ - q_i^- \tag{2.21}$$

Où $q_i^+ \geq 0$ est la quantité de marchandise "chargée" chez le client i tandis que $q_i^- \geq 0$ représente la quantité déchargée. Pour le cas du VRPB, chaque élément i de P respectivement de D aura $q_i^- = 0$ respectivement $q_i^+ = 0$. Pour forcer les collectes à la fin de chaque route, il faut introduire la contrainte suivante qui interdit qu'une livraison soit positionnée après une récupération :

$$x_{ij}^k = 0 \quad \forall i \in P, \forall j \in D, \forall k \in K \tag{2.22}$$

Très souvent, une contrainte supplémentaire est ajoutée qui proscrit la création de routes exclusivement composées de collectes :

$$x_{0i}^k = 0 \quad \forall i \in P, \forall k \in K \tag{2.23}$$

2.2.5 Le problème de tournées de véhicules avec retours mixés

Le problème de tournées de véhicules avec retours mixés (VRPMB pour Vehicle Routing Problem with Mixed Backhauls) est une extension du VRPB qui autorise que les collectes et les livraisons soient effectuées dans un ordre quelconque au cours de chaque route. Elles est notamment décrite dans l'étude de Parragh et al. [94, 95] sur les problèmes de transport associant opérations de collectes et de déposes. Les contraintes (2.22) et (2.23) doivent alors être retirées du précédent modèle. Les contraintes (2.19) et (2.20) sont remplacées par :

$$x_{ij}^k = 1 \Rightarrow Q_j^k = Q_i^k + q_j \qquad \forall i, j \in V, \forall k \in K \tag{2.24}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{W, W + q_i\} \qquad \forall i \in V, \forall k \in K \tag{2.25}$$

$$Q_0^k = -\sum_{j \in D} q_j \sum_{i \in V} x_{ij}^k \qquad \forall k \in K \tag{2.26}$$

Où Q_i^k représente la quantité de marchandise dans le véhicule k à son départ du nœud i. La contrainte (2.24) assure la conservation et le respect des quantités de marchandise quant à la (2.25), elle permet de s'assurer que la capacité maximale W par camion n'est pas dépassée. La contrainte (2.26) permet de s'assurer que chaque véhicule parte du dépôt avec la quantité nécessaire de marchandise pour effectuer ses livraisons. Cette quantité ne pouvant dépasser la capacité du véhicule sans violer la contrainte (2.25).

2.2.6 Le problème de tournées de véhicules avec prises et déposes simultanées

L'apparition de véhicules qui permettent un chargement par l'arrière ou les cotés et les avancées de la recherche dans les algorithmes de chargement permettent à présent d'envisager plus sérieusement des approches où les opérations de collecte et de ramassage peuvent être faites de manière quelconque. S'affranchir de la considération "delivery-first, pickup-second" permet une amélioration notoire de la qualité des solutions dans les applications réelles comme le suggère Nagy [87]. Dans notre problème, ces opérations peuvent être faites

simultanément. Nous avons alors affaire à un problème de tournées de véhicules avec prises et déposes simultanées(VRPSPD pour Vehicle Routing Problem with Simultaneous Pick-up and Deliveries) introduit pour la première fois par Min [83]. Dans ce problème, chaque client peut demander à la fois à être servi mais également délesté en marchandises. Les opérations de dépose et de collecte chez un client doivent être réalisées par un seul véhicule lors d'un seul et unique passage. Le VRPMB peut être vu comme un cas particulier du VRPSPD dans lequel chaque élément i de $V \setminus \{0\}$ a sa quantité $q_j^- = 0$ ou $q_i^+ = 0$. Min [83] présente une résolution du VRPSPD avec un dépôt et 22 clients. Des groupes de clients sont d'abord formés puis le problème du voyageur de commerce (TSP) est résolu une première fois dans chacun d'eux. Les vérifications de charge sont ensuite faites et les arcs qui ne sont pas faisables sont pénalisés puis la résolution du TSP recommencée. Dethloff [28] propose un modèle et une heuristique de résolution par insertion. Montané et Galvao [86] présentent également un modèle pour ce problème et une résolution par recherche Taboue. Nagy [87] décrit un ensemble d'heuristiques constructives et améliorantes pour le VRPSPD avec un ou plusieurs dépôts. Bianchessi et Righini [10] comparent les performances de différentes heuristiques et métaheuristiques. La modélisation de ce problème se fait en remplaçant les contraintes (2.26) du VRPMB par les contraintes suivantes :

$$Q_0^k = \sum_{j \in V \setminus \{0\}} q_j^- \sum_{i \in V} x_{ij}^k \qquad \forall k \in K \tag{2.27}$$

Ces contraintes permettent de s'assurer comme le faisaient (2.26) pour le VRPMP que chaque véhicule quitte le dépôt avec la quantité de marchandise suffisante pour subvenir à toutes les livraisons qu'il doit effectuer.

2.2.7 Le problème de collectes et déposes

De manière encore plus vaste, le problème de collectes et déposes (PDP pour Pick-up and Deliveries Problem) est la généralisation du VRPSPD. Dans le PDP, les marchandises sont échangées entre les clients et plus nécessairement à destination/en provenance du dépôt. Dans ce problème, si tous les clients adressent leurs marchandises à un seul d'entre eux et si ce client fournit également toutes les marchandises qui doivent être livrées à tous les autres clients alors ce client peut être vu comme un dépôt et le problème comme un VRPSPD. Le PDP devient le Dial-a-Ride Problem (DARP) lorsqu'il s'agit de transport de passagers avec prise en compte de critères de qualité de service. Partant d'une version enrichie du PDP avec fenêtres horaires (RPDPTW), Pisinger et Ropke [96] proposent de

résoudre plusieurs problèmes de tournées de véhicules qui en découlent. Ils construisent une méthode de recherche locale adaptative à voisinage large (ALNS pour Adaptative Large Neighborhood Search) combinant des heuristiques de destruction et de réparation.

2.2.8 Le problème de tournées de véhicules avec fenêtres horaires

Si les convoyeurs doivent livrer et/où récupérer de l'argent en un même point, ils doivent également le faire dans certaines plages horaires et notamment, pour des raisons évidentes de sécurité, hors de celles consacrées à l'ouverture au public. Le respect des plages horaires est modélisé dans le problème de tournées de véhicules avec fenêtres horaires (VRPTW pour Vehicle Routing Problem with Time Windows). Les premiers cas traités dans la littérature sont des cas pratiques tels le transport de courrier en vrac dans la région de Londres abordé par Pullen et Webb [103] en 1967 ou encore la planification de la flotte d'une entreprise de transport par Knight et Hofer [66]. Solomon [113] propose une heuristique de résolution extrêmement efficace. Il s'agit d'une méthode constructive par insertion basée sur l'heuristique de Clarke et Wright [18] et sur celle de Mole et Jameson [85] mais qui intègre la notion de décalage temporel induit par l'insertion d'un client dans une route. Il construit par la même occasion un jeu d'instances de 100 clients qui fait encore référence aujourd'hui pour l'évaluation d'algorithmes de résolution du VRPTW. Desrochers et al [27] développent une approche de résolution exacte basée sur la génération de colonnes. Enfin Cordeau et al. [19] introduisent un modèle. Une modélisation de ce problème peut être obtenue en ajoutant au modèle à trois indices pour le VRP de Fisher et Jaikumar les jeux de contraintes suivantes :

$$B_i^k + d_i + t_{ij}^k - M(1 - x_{ij}^k) \leq B_j^k \qquad \forall i \in V, \forall j \in V \setminus \{0\}, \forall k \in K \tag{2.28}$$

$$a_i \leq B_i^k \leq b_i \qquad \forall i \in V, \forall k \in K \tag{2.29}$$

La variable B_i^k représente la date de début de service chez le client i par le véhicule k, d_i est la durée opératoire chez le client i, t_{ij}^k est le temps de trajet du véhicule k entre le client i et le client j et a_i et b_i sont respectivement les dates de début au plus tôt et au plus tard de l'activité chez le client i. M est une constante de valeur très grande. La contrainte (2.28) permet l'élimination des sous-tours par l'utilisation des variables temporelles ce qui conduit à supprimer la contrainte (2.14) proposée pour le VRP. La contrainte (2.29) force le début d'opération chez le client i dans sa plage horaire.

2.2.9 Le problème de tournées de véhicules avec fenêtres horaires et prises et déposes simultanées et problèmes riches

Le problème de tournées de véhicules avec fenêtres horaires et prises et déposes simultanées (VRP-SPDTW pour Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows) est la combinaison de deux problèmes présentés précédemment : le VRPSPD et le VRPTW. Il s'agit donc d'un problème de tournées de véhicules avec dépose et collecte simultanée et prise en compte des fenêtres horaires. Il fait parti de la famille de problème que l'on qualifie de "riches" car composés de plusieurs attributs. Ce problème n'a pas fait l'objet d'un grand nombre d'études en dépit de son utilité pratique. On peut toutefois citer les travaux de Rieck et Zimmermann [108] qui proposent une heuristique de type "multi start" pour résoudre un problème de tournées de véhicules avec flotte hétérogène, dépose/collecte et fenêtres horaires. Les auteurs s'intéressent également aux contraintes inhérentes aux plateformes de chargement. Belmecheri et al. [8] proposent un algorithme de colonie de fourmis dans lequel les fourmis construisent les solutions par insertions, guidé par une heuristique inspirée de celle de Solomon [113]. Parragh et al. [94, 95] proposent quant à eux une étude en deux volumes des problèmes de collecte/dépose sur un modèle générique auquel plusieurs contraintes peuvent être rajoutées de manière à tenir compte, entre autres, des fenêtres horaires. Plus récemment, Mingyong et Erbao [84] proposent un modèle et une résolution du VRP-SPDTW par un algorithme d'évolution différentielle. De manière plus générale, les problèmes combinant plusieurs variantes du VRP sont appelés "problèmes riches". Il existe ainsi une grande quantité de problèmes riches dans les applications pratiques, le développement de méthodes générales pour leur résolution est devenu un enjeu majeur.

2.2.10 Le problème de tournées de véhicules avec gestion de stock

De nouveaux paradigmes d'organisation logistique tel le réapprovisionnement continu ou VMI (Vendor Managed Inventory en anglais) font apparaître de nouvelles variantes. Il s'agit de problèmes dans lesquels le transporteur doit assurer un niveau de disponibilité-produit chez chacun de ses clients. Il connait à priori les lois de consommation de chacun d'eux. Il est libre d'organiser ses livraisons comme bon lui semble et n'a de seul impératif que de maintenir la disponibilité du produit chez ses clients. L'objectif du transporteur va donc être d'organiser ses tournées de livraison (ordre et date de passage) afin de minimiser le coût de cette activité tout en respectant ses engagements. L'IRP (Inventory Routing Problem en anglais) permet de répondre à ce besoin de modélisation. Il a été introduit pour traiter des problèmes exigeants à la fois la prise en compte des dimensions stockage et transport,

Campbell et al. [14] en font une bonne description. L'IRP inclut classiquement les coûts de possession des marchandises pour l'évaluation de la fonction-objectif. Le problème des convoyeurs de fonds nécessite la prise en compte de la contrainte de disponibilité des fonds chez le client et de l'organisation de tournées. Un modèle simple d'IRP réduit à la prise en compte des coûts de transport et avec comme contrainte d'assurer la disponibilité en stock chez le client tout en respectant les plafonds, semble être une bonne base de modélisation. L'IRP nécessite un découpage de l'horizon temporel T en instants $t \in \{1, 2, \ldots, H\}$. Soit r une tournée appartenant à R l'ensemble des tournées extractibles du graphe G. Soit $serv_{ir}$ un indicateur qui vaut 1 si le client i est servi sur la tournée r et 0 sinon. On définit alors c_r le coût de cette route, x_{ir}^t la quantité de marchandise livrée au client i par la tournée r durant la période t et y_r^t une variable binaire telle que :

$$y_r^t = \begin{cases} 1, \text{ si la route } r \text{ est utilisée pendant la période } t, \\ 0, \text{ sinon.} \end{cases}$$

I_i^t est la quantité de marchandise en stock chez le client i à la période t, q_i la quantité de marchandise que le client consomme par période de temps t et C_i la capacité de stockage maximale du client i. On peut alors écrire le programme linéaire en variables mixtes suivant :

$$\min z = \frac{1}{H} \sum_{t \in T} \sum_{r \in R} c_r y_r^t \tag{2.30}$$

$$\sum_{i \in I} x_{ir}^t \leq Q y_r^t \qquad \forall t \in T, \forall r \in R \tag{2.31}$$

$$x_{ir}^t \leq Q serv_{ir} \qquad \forall t \in T, \forall i \in I, \forall r \in R \tag{2.32}$$

$$I_i^t = I_i^0 + \sum_{s=1}^{t} \sum_{r \in R} x_{ir}^s - t q_i \qquad \forall i \in I, \forall t \in T \tag{2.33}$$

$$I_i^t + q_i \leq C_i \qquad \forall i \in I, \forall t \in T \tag{2.34}$$

$$I_i^t \geq 0 \qquad \forall i \in I, \forall t \in T \tag{2.35}$$

$$x_{ir}^t \geq 0 \qquad \forall t \in T, \forall i \in I, \forall r \in R \tag{2.36}$$

$$y_r^t \in \{0, 1\} \qquad \forall t \in T, \forall r \in R \tag{2.37}$$

La fonction-objectif vise à minimiser le coût moyen d'approvisionnement sur une période. Les contraintes (2.31) assurent que la capacité des véhicules ne soit pas dépassée, (2.32) que le service du client i sur la route r est effectué s'il est effectivement visité par cette route. Les contraintes (2.33) définissent la quantité de marchandise en stock chez chaque client i et à

chaque instant t. Les contraintes (2.34) imposent le respect de la quantité maximum que le client i peut stocker et (2.35) assurent qu'il ne puisse pas y avoir de stock négatif (rupture de stock). Enfin les contraintes (2.36) et (2.37) donnent le domaine de définition des variables de décision.

2.2.11 Les problèmes de tournées de véhicules multi-périodes

Le problème de tournées de convoyeur de fonds implique la planification des livraisons sur plusieurs jours. Il s'agit alors d'un problème de tournées de véhicules multi-périodes que l'on note PVRP (Periodic Vehicle Routing Problem en anglais). La première description est faite par Beltrami et Bodin [9] qui s'intéressent au problème de collecte des déchets ménagers. Les lieux de collecte diffèrent par le nombre de visites qu'ils requièrent dans la semaine. Le but est alors de définir le plan des tournées pour chaque jour de la semaine, en minimisant la distance parcourue tout en respectant les passages requis pour chaque site. Une bonne revue de littérature du PVRP est faite par Francis et al. [41] dans le livre de Golden et al. [56]. Des méthodes heuristiques très performantes ont été développées notamment le Unified Tabu Search de Cordeau et al. [20], la recherche à voisinage variable de Hemmelmayr et al. [61], plus récemment l'algorithme génétique de Vidal et al. [120] et l'heuristique de Cacchiani et al. [13]. Des méthodes exactes ont également été proposées, citons Francis et al. [40] qui décrivent une approche par relaxation lagrangienne et Baldacci et al. [5] dont la méthode basée sur une formulation en problème de recouvrement permet de résoudre des instances comportant jusqu'à 100 clients et 6 périodes.

2.2.12 Les problèmes m-péripatétiques

Le problème de tournées de véhicules auquel nous avons affaire dans le cadre des tournées de convoyeurs de fonds doit pouvoir introduire des contraintes dites de "sécurité". La définition de ces contraintes n'est pas aisée et implique de prendre un parti, de se fixer une notion de ce qu'est la sécurité. Les récents travaux de thèse de S.U. Ngueveu [88] se rapportant entre autres aux problèmes de tournées des convoyeurs de fonds, proposent pour répondre à ces impératifs de sécurité d'introduire des contraintes de "péripatéticité" dans le modèle du VRP classique. Les contraintes de péripatéticité consistent à imposer dans le calcul de m solutions d'une instance de VRP que celles-ci n'aient aucun arc en commun. Ces contraintes de péripatéticité étaient jusqu'alors uniquement définies pour le TSP qu'elles font devenir le problème du vendeur m-péripatétique.

2.2.12.1 Le problème du vendeur m-péripatétique

Comme précédemment mentionné, une possibilité pour introduire un degré de variation dans les routes est d'éviter de réutiliser une route d'une période sur l'autre. Cet attribut qui impose l'utilisation unique des routes définit la famille des problème de tournées de véhicules péripatétiques. Le premier problème de ce type a été défini par Krarup [69], il s'agit du problème du vendeur m-péripatétique (m-PSP pour m-peripatetic salesman problem). Ce problème ne contenant pas de contraintes de capacité est une extension du TSP sur m périodes. Tous les clients doivent être visités une fois par période mais chaque arrête $[i, j]$ ne peut être utilisée au plus qu'une seule fois sur l'horizon de planification. En d'autre termes, le m-PSP consiste à trouver m cycles hamiltoniens disjoints avec un poids total minimum. Krarup propose une heuristique de résolution simple : partant du graphe original, résoudre un problème de TSP pour obtenir un cycle, retirer du graphe les arêtes utilisées par ce cycle et recommencer jusqu'à avoir obtenu le nombre de cycles voulu ou qu'il ne soit plus possible d'en produire (le graphe n'est plus hamiltonien). De Kort [22] a quant à lui donné des bornes inférieures pour le m-PSP ; elles sont basées respectivement sur le calcul de chemins à arêtes disjointes de coût minimum et sur le coût de la solution optimale du problème de TSP associé au graphe original. Il prouve aussi [23] que le m-PSP est NP-difficile et décrit une borne inférieure pour le 2-PSP basée sur le calcul de deux arbres couvrants disjoints de coût minimum. Les arbres obtenus sont ensuite transformés en cycles hamiltoniens par des heuristiques dans le but d'obtenir des solutions faisables pour le 2-PSP. Pour finir, il décrit un branch and bound [24] capable de résoudre à l'optimalité des instances de 2-PSP contenant jusqu'à 130 nœuds. Une approche polyédrale pour le m-PSP est présentée par Duchenne et al. [32, 33] et permet la résolution d'instances jusqu'à 280 nœuds pour $m = 2$ et 26 nœuds pour $m = 12$. Une borne inférieure et un ensemble d'heuristiques inspirées de celle de Krarup [69] ont été développées plus récemment par les mêmes auteurs [34]. La borne inférieure est obtenue en agrégeant les contraintes des m périodes sur une seule. Le problème ainsi relaxé est résolu en ajoutant différentes inégalités valides et en utilisant l'algorithme de branch and cut développé dans [32]. Cinq des sept heuristiques proposées ensuite effectuent des mouvements de type 2-opt sur le premier cycle de la solution obtenue par l'heuristique de Krarup [69] puis en complétant la solution normalement ensuite. Cette procédure est alors répétée jusqu'à ce qu'il ne soit plus possible de trouver de mouvements 2-opt améliorant dans le premier cycle. Cette méthode présente le désavantage d'être potentiellement coûteuse en temps dans la mesure ou plusieurs problèmes de TSP sont résolus dans la phase de complétion. Les auteurs proposent alors d'accélérer le calcul en travaillant sur un graphe réduit. Ce graphe est construit à partir du graphe original par suppression

des arrêtes ne pouvant pas appartenir à une solution meilleure que celle de l'heuristique de Krarup. La borne inférieure précédemment décrite est utilisée pour arrêter le processus de complétion lorsqu'il ne pourra pas améliorer la meilleure solution courante. La sixième heuristique propose de favoriser les meilleures arrêtes du dernier cycle de la solution obtenue par l'heuristique de Krarup. La dernière heuristique décrite utilise un graphe réduit et essaye d'améliorer la solution courante en retirant les arcs utilisés les plus couteux puis en relançant l'heuristique de Krarup. Toutes ces heuristiques sont évaluées sur des instances contenant jusqu'à 318 nœuds et 9 périodes générées aléatoirement ou provenant de la TSPLIB [107]. Ces heuristiques se révèlent performantes puisqu'elles se trouvent en moyenne à 0.45% des meilleures bornes inférieures connues.

Quelques résultats d'approximation sont disponibles pour le 2-PSP. Ageev et al. [2] proposent une 3/4-approximation pour le problème consistant à trouver deux cycles hamiltoniens mais avec un poids maximum. Pour le 2-PSP euclidien (le poids des arcs satisfait l'inégalité triangulaire) et en minimisation, Ageev et al. [3] présentent une 2-approximation. Ils se basent sur l'algorithme de Roskind et Tarjan [109] de manière à obtenir deux arbres couvrant de poids minimum desquels deux cycles hamiltoniens disjoints sont déduits. Le facteur d'approximation peut être encore amélioré lorsque le poids des arcs est égal à 1 ou 2. Gimadi et al. [49] ont récemment proposé quatre algorithmes polynomiaux avec des facteurs de performance $4/3$, $5/4$, $26/21$ et $6/5 = 1.2$. Tous ces algorithmes consistent à transformer des tours partiels sans arrêtes communes en cycles hamiltoniens.

Quelques applications pratiques ont été décrites pour le m-PSP. Lindner-Dutton et al. [75] suggèrent de l'utiliser pour optimiser le transport de matières dangereuses. Les différentes périodes générées n'utilisant pas les mêmes routes permettent une répartition équitable du risque sur les zone visitées tout en minimisant le coût de transport. De Kort [24] propose de l'employer dans la conception de réseaux informatiques où un ensemble de cycles disjoints reliant tous les nœuds doit pouvoir être trouvé de manière à minimiser l'impact de la rupture d'une connexion. Une autre application ayant trait à la sécurité est décrite par Wolfler Calvo and Cordone [123] et concerne une compagnie de surveillance italienne. Ils définissent le problème du gardien de nuit (Watchman Tour Problem) dans lequel un veilleur de nuit doit assurer une série de visites sur certains sites en évitant d'être prévisible. Les contraintes péripatétiques sont, entre autres, utilisées pour diversifier les routes employées et une heuristique de décomposition est utilisée pour résoudre des instances pratiques.

Le problème du vendeur m-péripatétique (m-PSP pour m-Peripatetic Salesman Problem) peut se formuler de la manière suivante : Soit $G = (V, A)$ un graphe orienté complet où $V = \{1, 2, i, j \ldots, n\}$ l'ensemble des sommets représente les clients à visiter et A l'ensemble

des arcs (i, j) représente les routes entre ces clients. Soit P l'horizon temporel découpé en m périodes $p \in \{1, 2, \ldots, m\}$. On associe alors à chaque arc (i, j) de A un coût c_{ij} positif et on utilise les mn^2 variables de décision x_{ij}^p telles que :

$$x_{ij}^p = \begin{cases} 1, \text{ si l'arc } (i,j) \text{ est utilisé pendant la période } p, \\ 0, \text{ sinon.} \end{cases}$$

La solution est obtenue en résolvant le programme linéaire en 0-1 suivant :

$$\min z = \sum_{p \in P} \sum_{(i,j) \in A} c_{ij} x_{ij}^p \tag{2.38}$$

Sous les contraintes :

$$\sum_{i \in V} x_{ij}^p = 1 \qquad \forall j \in V, \forall p \in P \tag{2.39}$$

$$\sum_{j \in V} x_{ij}^p = 1 \qquad \forall i \in V, \forall p \in P \tag{2.40}$$

$$\sum_{i,j \in S} x_{ij}^p \leq |S| - 1 \qquad \forall S \subset V, S \neq V, S \neq \emptyset, \forall p \in P \tag{2.41}$$

$$\sum_{p \in P} x_{ij}^p \leq 1 \qquad \forall i, j \in V \tag{2.42}$$

Le modèle suivant n'est ni plus ni moins que celui du TSP pour chaque période p à qui l'on a rajouté la contrainte (2.42). C'est cette contrainte qui impose pour chaque arc donné du graphe G une utilisation unique sur l'ensemble des périodes $p \in P$.

2.2.12.2 Le problème de tournées de véhicules m-péripatétique

Les contraintes de péripatéticité ont été adaptées au CVRP par Ngueveu et al. [90]. Ils définissent le problème de tournées de véhicules m-péripatétique (m-PVRP pour m-Peripatetic Vehicle Routing Problem) et proposent trois bornes inférieures "simples". La première propose de considérer le coût de la solution optimale du CVRP correspondant sur une période et de le multiplier par le nombre de périodes considérées. La seconde est basée sur un ensemble d'arbres couvrants disjoints de poids minimum obtenu à l'aide de l'algorithme de Roskind et Tarjan [109] sur l'ensemble des clients et complété par la paire d'arêtes de plus faible poids avec le dépôt. Les arbres couvrants ne prenant pas en compte les contraintes de capacité et

les contraintes de degrés sur les nœuds, la solution produite est donc bien une borne inférieure. La troisième borne proposée se base sur une transformation du problème initial en un problème de b-couplage parfait relaxé. Un problème polynomial est donc obtenu et peut être résolu avec un solveur classique. Cette dernière borne est la meilleure, elle se trouve en effet très proche ($< 2\%$) des meilleures bornes supérieures qui seront trouvées par la suite. De plus, son efficacité augmente avec le nombre de périodes considérées. Une quatrième borne inférieure plus élaborée est proposée en résolvant par génération de colonnes le modèle de partitionnement agrégé de l'ensemble des périodes. Les auteurs proposent en particulier d'appliquer ce problème à l'activité de transport de fonds et décrivent une recherche taboue guidée par la solution du b-couplage parfait relaxé afin de pouvoir traiter des instances de tailles réelles [89]. Les tests sont effectués sur les 7 instances VRPNC proposées par Christofides et al. [17] contenant 7 à 200 nœuds et sur les ensembles d'instances A, B et P pour le VRP de Augerat contenant respectivement 27, 23 et 23 instances de 19 à 101 sommets et sur 20 instances euclidiennes de la TSPLIB [107] utilisées auparavant par Duchenne et al.[33] avec un nombre de périodes compris entre 2 et 7.

Le m-PVRP peut être modélisé par un programme linéaire en nombres entiers. On considère x_e^p la variable de décision qui vaut 1 si l'arrête e est utilisée au cours de la période $p \in \{1, 2, \ldots, m\}$ et 0 sinon. On notera également $\delta(S) \subset E$ l'ensemble des arêtes qui ont une extrémité dans $S \subseteq V \setminus \{0\}$ et l'autre en dehors. Pour une meilleure lisibilité, on notera $\sigma(i)$ à la place de $\sigma(\{i\})$. On notera également $r(S)$ la valeur qui définie le nombre minimal de tournées nécessaires au service d'un sous-ensemble $S \subseteq V \setminus \{0\}$ de clients. Cette valeur est définie comme suit :

$$r(S) = \left\lceil \sum_{i \in S} \frac{q_i}{W} \right\rceil \tag{2.43}$$

On écrira $\lambda = r(V \setminus \{0\})$ le nombre minimal de tournées nécessaires pour approvisionner l'ensemble des clients. Soit $G = (V, E)$ un graphe non orienté, il est alors possible d'écrire le programme linéaire en nombres entiers suivant :

$$\min z = \sum_{p \in P} \sum_{e \in E} c_e x_e^p \tag{2.44}$$

Sous les contraintes :

$$\sum_{e \in \sigma(i)} x_e^p = 2 \qquad \forall p \in P, \forall i \in V \setminus \{0\} \tag{2.45}$$

$$\sum_{e \in \sigma(0)} x_e^p \geq 2\lambda \qquad \forall p \in P \tag{2.46}$$

$$\sum_{e \in \sigma(S)} x_e^p \geq 2r(S) \qquad \forall p \in P, S \subseteq V \setminus \{0\}, S \neq \varnothing \tag{2.47}$$

$$\sum_{p \in P} x_e^p \leq 1 \qquad \forall e \in E \tag{2.48}$$

Les contraintes (2.45) Les contraintes (2.46) assurent que le nombre d'arêtes incidentes au dépôt correspondent au moins au nombre minimal de tournées requises pour la desserte des clients. Comme une tournée valide possède deux arrêtes incidentes au dépôt, au moins 2λ arêtes sont requises.

2.3 Méthodes de résolution exactes

Dans cette section, nous nous intéressons aux méthodes utilisées pour d'obtenir la ou les solution(s) optimale(s) d'un problème d'optimisation combinatoire NP-difficile comme le problème de tournées de véhicules. On oppose traditionnellement les méthodes exactes avec les méthodes dites "approchées", qui ne fournissent pas nécessairement la ou les solution(s) optimale(s).

2.3.1 Énumération complète

C'est la première solution qui vient à l'esprit lorsque l'on se trouve face à un problème combinatoire. Énumérer et tester toutes les solutions puis conserver la meilleure d'entre elles. Considérons par exemple le problème du voyageur de commerce (TSP) décrit en 2.2.1, qui consiste à trouver l'ordre dans lequel un représentant de commerce doit visiter ses clients afin de minimiser la distance totale parcourue. Ce problème est d'une grande simplicité apparente. Si l'on considère le problème avec deux clients n_1 et n_2, on peut soit visiter n_1 puis n_2 soit n_2 puis n_1. On explore en fait les différentes permutations de clients, pour $n = 2$ clients il y a $n! = 2$ permutations possibles. Prenons maintenant le problème avec 10 clients ($n = 10$), $10! = 3628800$ permutations possibles et avec 100 il y en a $9,33 * 10^{157}$. Si l'on considère qu'un ordinateur actuel est capable d'explorer un milliard de ces combinaisons en une seconde, cela prendrait $9,33 * 10^{148}$ secondes soit environ 10^{133} milliards d'années. A la

date à laquelle cette thèse est rédigée, l'age de l'univers est estimé à 14 milliards d'années. Un petit transporteur peut facilement servir 500 clients par jours, il est donc impensable d'utiliser cette méthode.

2.3.2 Programmation dynamique

La programmation dynamique est une méthode basée sur la décomposition du problème original en sous problèmes imbriqués. Le paradigme est qu'une décision optimale peut être obtenue par une suite de sous-décisions optimales. Elle est développée par Bellman [7] pour résoudre des problèmes de chemins optimaux (de coûts minimums ou maximums). Le principe d'optimalité de la programmation dynamique repose sur la propriété d'un problème de faire apparaitre des sous structures optimales. La solution optimale du problème considéré peut alors être obtenue en combinant des solutions optimales de sous-problèmes. On dit que la résolution du problème se fait de bas en haut (Bottom up), on commence par obtenir les solutions optimales pour des problèmes de petites tailles qui sont ensuite utilisées pour obtenir la solution optimale du problème complet. La méthode permet de ne pas réexaminer inutilement des ensembles de solutions qui l'ont déjà été. Le meilleure exemple de son efficacité est son utilisation dans la résolution du problème de sac à dos en $0-1$. Ce problème se pose de manière simple en considérant un sac à dos de capacité W dans lequel on souhaite emporter des objets tel que chaque objet i a une taille w_i et un profit p_i. Il s'agit alors de choisir parmi n objets donnés un sous-ensemble dont la taille ne dépasse pas W et maximise le profit réalisé.

$$\max z = \sum_{i=1}^{n} p_i x_i \tag{2.49}$$

Sous les contraintes :

$$\sum_{i=1}^{n} w_i x_i \leq W \tag{2.50}$$

$$x_i \in \{0,1\} \qquad \forall i \in \{1,\dots n\} \tag{2.51}$$

La programmation dynamique appliquée à ce problème consiste à décomposer le problème initial en sous-problèmes. Soit $z^*(k,w)$ le profit maximal réalisable en considérant les k premiers objets, on peut alors écrire $z^*(k,w) = \max\{z^*(k-1,w), z^*(k-1,w-w_k)+p_k\}$. La résolution du problème se fait donc en évaluant les Wn sous-problèmes. L'obstacle majeur auquel se heurte la programmation dynamique est le nombre de sous-problèmes à résoudre et

la mémoire nécessaire au stockage de tous les résultats intermédiaires. La taille des problèmes pouvant être résolue est ainsi limitée en pratique.

2.3.3 Algorithme par séparation et évaluation

Les méthodes par séparation et évaluation (Branch and Bound selon le terme anglo-saxon) introduites par Land et Doig [70] fonctionnent selon le principe "diviser pour régner". Comme nous avons pu nous en rendre compte précédemment, l'ensemble des solutions d'un problème combinatoire est beaucoup trop grand pour pouvoir être exploré de manière exhaustive. Un algorithme par séparation et évaluation va donc diviser le problème initial en sous-problèmes de sorte que l'union de leurs ensembles de solutions respectifs forment un recouvrement (ou mieux, une partition) de celui du problème initial. Chaque sous-problème obtenu peut être séparé de la même façon et ceci récursivement jusqu'à ce que l'ensemble de solutions associé soit un singleton. L'ensemble de solutions de chacun des sous-problèmes obtenus est alors évalué afin de savoir s'il est pertinent d'en continuer l'exploration. L'évaluation d'un sous-ensemble de solutions est fait implicitement, généralement, elle fait appel au calcul d'une borne inférieure (en minimisation). La borne inférieure est généralement calculée en considérant une relaxation (linéaire, lagrangienne, ...) du sous-problème associé, elle donne une estimation du coût de la meilleure solution du sous-ensemble considéré. Si la solution associée à cette borne inférieure est réalisable pour le problème non relaxé, elle remplace la meilleure solution connue si elle est de meilleur coût et on ne sépare plus le sous-problème. Si par contre le coût de la borne inférieure est supérieure (toujours en minimisation) au coût de la meilleure solution valide connue, il est inutile de poursuivre l'exploration du sous ensemble considéré. La performance d'un algorithme par séparation et évaluation est en grande partie liée à la qualité des bornes et au schéma de séparation qui est adopté. En effet, dans le cas d'un problème de minimisation, si la borne inférieure est trop basse (i.e. la relaxation est trop forte), aucun sous ensemble ne sera éliminé et l'algorithme risque de se réduire à une énumération complète.

2.3.4 Algorithme de branchement et coupes

Une solution optimale d'un programme linéaire se trouve sur un sommet du polyèdre convexe défini par les contraintes. Les algorithmes de branchement et coupes (Branch and cut) initialement introduits par Gomory [57] proposent de commencer la résolution en considérant une relaxation du problème initial (généralement il s'agit de la relaxation continue). Si la solution optimale du problème relaxé est entière, alors elle est aussi solution optimale du

problème initial. Si cette solution est fractionnaire, on ajoute alors un jeu de contraintes (ou coupes) "valides" qui ne sont pas vérifiées par cette solution mais qui reste valides pour toute solution réalisable du problème initial. La résolution est alors relancée jusqu'à obtention d'une solution entière qui est alors la solution optimale ou jusqu'à ce que l'algorithme ne parvienne plus à trouver de coupe valide. On ne dispose alors que d'une borne inférieure pour le problème. Des plans de coupe génériques ont été proposés notamment par Gomory [57] mais aussi par Balas et al. [4] suivant la méthode "lift and project". Padberg et Rao [92] montrent qu'un problème peut être résolu en temps polynomial si et seulement si il est possible de séparer les contraintes définissant le polytope en temps polynomial. Malheureusement, pour les problèmes de programmation linéaire en nombres entiers (PLNE) qui sont NP-difficiles, on ne connait pas de description complète du polytope associé. Il n'existe pas non plus d'algorithme de séparation exacte pour toutes les familles de facettes. Souvent donc la méthode de plans sécants s'arrête avant d'avoir trouvé la solution optimale. C'est pour cette raison que la combinaison avec un algorithme par séparation et évaluation a été proposé par Grötschel et al. [59]. La méthode de coupe est alors utilisée pour le calcul des bornes inférieures de la procédure d'évaluation. C'est Padberg et Rinaldi [93] qui emploient pour la première fois le terme " branch and cut" et utilisent la méthode pour un problème de voyageur de commerce. La combinaison permet de profiter à la fois de la capacité d'exploration de l'algorithme par séparation et évaluation, et de l'efficacité des plans de coupe.

2.3.5 Génération de colonnes

Les méthodes de résolution exactes de programmes linéaires en nombre entier telles que décrites précédemment (2.3.4, 2.3.3) se basent souvent sur la résolution de la relaxation linéaire du problème initial. Il peut arriver que cette relaxation contienne un nombre exponentiel de variables, la rendant impossible à résoudre en pratique avec la méthode du simplexe. La génération de colonne est basée sur la constatation qu'à l'optimum, la plupart des variables se trouvent hors-base et nulles. Seul un petit sous-ensemble de variables est donc nécessaire pour caractériser l'optimum. La génération de colonne débute donc la résolution du programme linéaire en ne considérant qu'un sous-ensemble restreint de colonnes (variables) qui forment le "problème maître". Au terme de cette résolution, deux cas de figures sont possibles : Si aucune des colonnes non choisies n'a un coût réduit négatif, la solution optimale est atteinte car même si ces colonnes étaient considérées, elles ne pourraient pas améliorer la valeur de la fonction objectif. S'il existe des colonnes non choisies de coût réduit négatif, elles doivent donc être insérées dans le problème maître et celui-ci est résolu à nouveau. La détection de colonnes à coût réduit négatif est appelé problème de "pricing", l'efficacité

de la méthode dépend grandement de la qualité de cette procédure. Pour les problèmes de tournées de véhicules, une formulation du problème maître sous forme d'un problème de partitionnement ou de couverture est la plupart du temps utilisée. Les variables considérées sont toutes les routes réalisables possibles. Le sous-problème de pricing consiste à trouver des tournées réalisables de coûts réduits négatifs afin de les faire entrer dans le problème maitre. Ce sous problème peut souvent se ramener à un problème de recherche de plus court chemin élémentaire avec contraintes de ressources (Elementary Shortest Path Problem with Resource Constraints, ESPPRC). Le problème de plus court chemin élémentaire est malheureusement NP-difficile. Toutefois, Feillet et al. [35] proposent une méthode de résolution efficace basée sur la programmation dynamique où la multiplication des labels est contenue par des règles de dominance. Cette méthode est une extension de celle proposée précédemment par Desrochers et al. [27] pour le calcul du plus court chemin avec contraintes de ressources. Ils l'utilisent comme sous-problème dans une résolution par génération de colonne du problème de tournée de véhicule avec fenêtre horaire. Le problème de plus court chemin avec contraintes de ressources mais sans contrainte d'élémentarité (SPPRC) est plus facile à résoudre par programmation dynamique mais n'en demeure pas moins NP-difficile.

2.3.6 Relaxation lagrangienne

La relaxation lagrangienne est basée sur le constat que la plupart des problèmes d'optimisation combinatoire peuvent être vus comme des problèmes simples, compliqués par quelques contraintes "difficiles". L'idée est de s'affranchir de ces contraintes compliquées en pénalisant leur violation dans la fonction objectif. Un problème relaxé, plus facile à résoudre, est alors obtenu et sa résolution fournit une borne inférieure du problème initial. Geoffrion [47] a montré que cette borne inférieure est au moins aussi bonne que celle fournie par la relaxation linéaire du problème, elle peut donc être utilisée au sein d'une méthode par séparation et évaluation. Si l'on considère par exemple le programme linéaire suivant :

$$\min z = c^T x \tag{2.52}$$

Sous les contraintes :

$$Ax \geq b \tag{2.53}$$

$$Dx \geq e \tag{2.54}$$

$$x \in \mathbb{R}^{n^+} \tag{2.55}$$

Les matrices A et B définissent les contraintes et sont telles que $A \in \mathbb{R}^{m_1,n}$ et $B \in \mathbb{R}^{m_2,n}$. Supposons maintenant que les contraintes (2.53) rendent la résolution de ce programme

linéaire difficile. Elles ne sont alors plus considérées comme contraintes mais leur violation est pénalisée dans la fonction objectif :

$$\min L(x, \lambda) = c^T x + \lambda^T (b - Ax) \tag{2.56}$$

Sous les contraintes :

$$Dx \geq e \tag{2.57}$$

$$x \in \mathbb{R}^{n^+} \tag{2.58}$$

Dans ce nouveau programme linéaire, $\lambda = (\lambda_1, \ldots, \lambda_{m_1})$ est le vecteur des multiplicateurs lagrangiens tel que $\lambda \in \mathbb{R}^{{m_1}^+}$. La relaxation de ces contraintes "gênantes" doit permettre de minimiser facilement la nouvelle fonction objectif $L(x, \lambda)$ pour λ fixé, ce qui revient à déterminer un point de la fonction duale $L(\lambda) = \min_{x \in \mathbb{R}^{n^+}} \{L(x, \lambda)\}$. On remarque alors que $L(\lambda) \leq L(x, \lambda)$ pour tout λ et tout x. Or $\lambda \geq 0$ et $b - Ax \leq 0$ donc $c^T x + \lambda^T (b - Ax) \leq c^T x$ soit $L(x, \lambda) \leq z$ et donc $L(\lambda) \leq z$. On peut donc dire que $L(\lambda)$ est une borne inférieure pour la fonction z du problème original car on a en particulier $L(\lambda) \leq z^*$. La question qui se pose est alors de savoir quel est le vecteur λ qui assure la meilleure borne inférieure (i.e. la plus élevée). La recherche de ce vecteur se fait en résolvant le problème $\max_{\lambda \in \mathbb{R}^{m_1+}} \{L(\lambda)\}$ qui est appelé problème dual lagrangien. Le plus souvent, la méthode de sous-gradient est utilisée. Elle agit en déplaçant de manière itérative le vecteur λ dans la direction opposée au sous-gradient de la fonction objectif actuelle. La plupart du temps, lorsque le problème original est en nombres entiers, il n'est pas possible d'aboutir à une solution optimale car le saut de dualité $z^* - L(\lambda^*)$ n'est pas nul. Il n'en reste pas moins qu'a chaque itération de la méthode de sous-gradient une borne inférieure de très bonne qualité est produite et peut être utilisée dans une méthode arborescente ou pour l'évaluation de la qualité d'une heuristique.

2.4 Méthodes de résolution heuristiques

Les tailles de problèmes que peuvent traiter les méthodes exactes sont souvent limitées et les rendent ainsi inutilisables dans un contexte opérationnel. Des méthodes beaucoup plus rapides existent afin de pouvoir obtenir des solutions aux problèmes de grande taille. Le prix à payer pour cette vitesse de calcul est l'optimalité des solutions obtenues qui n'est alors plus garantie. Le paradigme sur lequel se basent ces méthodes est qu'il vaut mieux avoir une très bonne solution (dont l'optimalité n'est certes pas prouvée) tout de suite que de ne rien avoir du tout.

2.4.1 Heuristiques constructives

Les heuristiques constructives peuvent être vues comme des "recettes" pour construire une solution. Elles sont soit issues de principes empiriques, soit de tâtonnements effectués, ou alors construites spécifiquement pour garantir un meilleur rapport possible à l'optimum. Il existe un très grand nombre d'heuristiques pour les problèmes de tournées de véhicules, seules les plus rencontrées sont décrites ici.

2.4.1.1 Heuristique du plus proche voisin

C'est la plus simple des heuristiques constructives. Elle consiste, en partant du dépôt, à se rendre au client le plus proche. Le client le plus proche du dernier ajouté est ensuite choisi et ainsi de suite. Cette opération est répétée jusqu'à ce qu'il ne soit plus possible de rajouter de clients dans la tournée sans violer de contraintes. Une nouvelle tournée est ensuite démarrée à partir d'un client choisi selon un critère défini par l'utilisateur. L'algorithme s'arrête lorsque tous les clients sont affectés à une tournée. A chaque étape, l'heuristique de plus proche voisin effectue un choix qui minimise localement le critère de coût, elle est en ce sens un algorithme glouton. Sa performance varie beaucoup en fonction de l'instance sur laquelle elle est exécutée, on constate cependant qu'elle est plus efficace lorsque les clients sont regroupés en paquets plutôt que de manière homogène dans l'espace. Son implémentation est simple et son exécution rapide avec une complexité en $O(n^2)$.

2.4.1.2 Heuristique de Clarke et Wright

Cette heuristique, du nom de ses auteurs Clarke et Wright [18], est basée sur le principe de l'épargne. Il s'agit de partir de la solution la plus coûteuse et d'effectuer tour à tour des changements pour améliorer le coût global de la fonction objectif. Dans le cas du VRP classique, voici comment se déroulerait cette heuristique : Soit un VRP simple avec trois clients et un dépôt, la solution la plus coûteuse est alors construite.

Cette solution consiste à faire un aller retour entre le dépôt et chacun des clients, utilisant ainsi autant de camions que de clients comme illustré par la figure 2.1. Pour chaque couple de clients (i, j) une quantité $s_{i,j}$ qui est l'économie réalisée en reliant le client i au client j est ensuite calculée. Cette économie vaut ici : $s_{i,j} = c_{i,0} + c_{0,j} - c_{i,j}$ où $c_{i,j}$ est le coût d'utilisation de l'arc (i, j). S'il est décidé de relier le client 1 au client 3 dans l'exemple de la figure 2.1, cela économise :

- Le retour de 1 au dépôt (maintenant la route continue vers 3), c'est le $c_{i,0}$ ici $c_{1,0}$.
- L'aller du dépôt à 3 (effectivement l'arrivée se fait maintenant de 1), c'est le $c_{0,j}$ ici $c_{0,3}$.

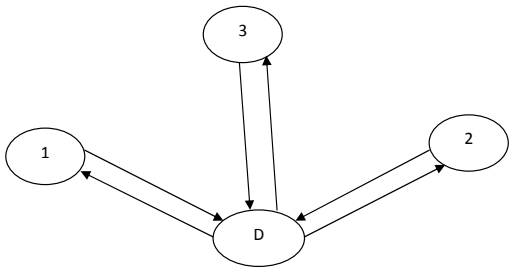


FIGURE *2.1: Première solution*

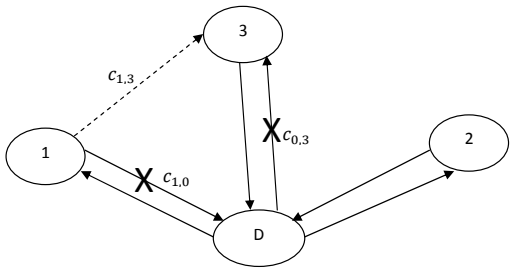


FIGURE *2.2: Ajout de l'arc (1, 3)*

Il faut retirer de cette économie le coût du trajet de 1 à 3 auparavant inexistant, c'est le $c_{i,j}$ ici $c_{1,3}$. L'économie totale effectuée par l'ajout de l'arc (1,3) vaut donc $s_{1,3} = c_{1,0} + c_{0,3} - c_{1,3}$, le résultat est représenté sur la figure 2.2. Une fois toutes les économies $s_{i,j}$ calculées (en $O(n^2)$), la plus intéressante est appliquée sous réserve qu'elle ne viole aucune contrainte du problème. L'opération est ainsi répétée jusqu'à ce qu'il n'y ait plus d'améliorations possibles en ajoutant un arc (i, j) (i.e. tous les $s_{i,j}$ sont négatifs ou nuls). Cette heuristique est donc relativement simple et rapide puisqu'elle peut être implémentée en $O(n^2 \log_2 n)$.

2.4.1.3 Heuristiques d'insertion

Les heuristiques par insertion procèdent en insérant successivement les clients dans la solution. La place à laquelle se fait l'insertion est celle qui maximise un critère choisi, faisant de ces heuristiques des algorithmes gloutons. Comme toutes les positions d'insertion sont testées pour tous les clients, la complexité est en $O(n^3)$. L'heuristique de Mole et Jameson est une de ces procédures conçues pour le VRP. Elle est basée sur le calcul de deux critères α et β. Le premier est un critère d'insertion d'un sommet $v_k \in V \setminus \{v_0\}$ entre deux clients consécutifs v_i et $v_j \in V$ sur une tournée : $\alpha(v_i, v_k, v_j) = c_{ik} + c_{kj} - \lambda c_{ij}$ où c_{ij} est le coût du déplacement entre le sommet i et le sommet j et λ un paramètre fixé par l'utilisateur. Si celui-ci est pris égal à 1, $\alpha(v_i, v_k, v_j)$ peut être vu comme le surcoût induit par l'insertion du sommet v_k entre v_i et v_j. Le second paramètre β est déterminé de la manière suivante : $\beta(v_i, v_k, v_j) = \mu c_{0k} - \alpha(v_i, v_k, v_j)$. Ce critère dépend également d'un paramètre μ. Si celui-ci est pris égal à 2, $\beta(v_i, v_k, v_j)$ peut être vu comme le gain engendré par l'insertion de v_k dans la route de i à j par rapport à la desserte de v_k directement depuis le dépôt. Les tournées sont construites séquentiellement. Une première tournée constituée du dépôt et du sommet qui en est le plus éloigné est considérée. La meilleure insertion dans cette tournée est calculée pour chaque sommet grâce au premier critère α. Une fois que la meilleure insertion dans la tournée de tous les sommets candidats est connue, le sommet qui maximise le second critère β est inséré si cela ne viole pas la contrainte de capacité du véhicule. Les insertions se ré-

pètent ainsi jusqu'à ce que l'on ne puisse plus insérer de clients dans la tournée auquel cas une nouvelle tournée est démarrée entre le point non desservi le plus éloigné du dépôt et le dépôt lui même. L'algorithme s'arrête lorsque tous les clients ont été affectés à une tournée.

Solomon [113] adapte cette heuristique pour le problème de tournée de véhicules avec fenêtres horaires (VRPTW) en 1987. Une première route est initiée avec un nœud choisi grâce à un critère d'initialisation. Solomon en propose deux, soit le client le plus éloigné du dépôt, soit celui dont la date de fermeture de la fenêtre horaire est la plus tôt. Tout comme pour l'heuristique de Mole et Jameson, deux critères, ici c_1 et c_2, sont utilisés conjointement. Pour chaque nœud non encore inséré u, sa meilleure insertion dans la route en cours est celle qui minimise le critère $c_1(i,u,j)$ avec $c_1(i,u,j) = \alpha_1(d_{iu} + d_{uj} - \mu d_{ij}) + \alpha_2(b_{j_u} - b_j)$. Dans cette expression d_{ij} désigne la distance entre deux nœuds i et j, b_j est la date arrivée au client j avant l'insertion de u, b_{j_u} la date d'arrivée au client j après insertion de u et α_1, α_2 et μ des paramètres tel que $\alpha_1 + \alpha_2 = 1, \alpha_1 \geq 0, \alpha_2 \geq 0, \mu \geq 0$. La meilleure insertion de u retenue est celle qui minimise le critère $c_1(i,u,j)$. La figure 2.3 donne une illustration de cet étape de l'algorithme. L'insertion effectuée est alors celle qui maximise le critère

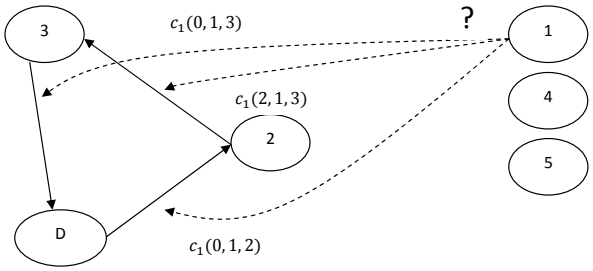


FIGURE *2.3: Exploration des positions d'insertion pour le client 1*

$c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j)$ avec λ un paramètre tel que $\lambda \geq 0$. Le processus se répète ensuite jusqu'à ce qu'il n'y ait plus d'insertion faisable dans la route en cours. Une nouvelle route au départ du dépôt est alors initiée en choisissant un nœud parmi ceux non encore positionnés et l'heuristique s'arrête lorsqu'il ne reste plus de nœuds à insérer. Cet algorithme dont le principe est simple, intégrant bien la dimension "temporelle" du problème donne de très bons résultats. Pour traiter des instances de grandes tailles, l'algorithme doit être implémenté de manière efficace afin d'obtenir une complexité acceptable. Il faut en effet s'assurer de la faisabilité de toutes les insertions qui sont testées, tant en terme de capacité que de respect des fenêtres horaires. Cette vérification peut être faite en temps constant ($O(1)$). Solomon explique comment : en maintenant à jour une variable pour chaque client qu'il nomme "Push-Forward". Elle représente la quantité de temps dont il est possible de retarder l'arrivée chez ce client sans violer les contraintes de fenêtres horaires. Certes, cela

nécessite de maintenir les "Push-Forward" à jour mais ceci n'est fait que lorsqu'un client est inséré dans une route, tous les tests d'insertion qui suivent en bénéficient.

2.4.1.4 Heuristiques Cluster First Route Second

Ces heuristiques partitionnent d'abord l'ensemble des clients en groupes puis décident de leur séquencement dans un deuxième temps. L'heuristique de Gillet et Miller [48], algorithme dit par "balayage" ("sweep algorithm" en anglais), est l'illustration la plus simple de ce principe. Les clients à servir sont classés en fonction de leurs coordonnées polaires dans le plan euclidien. La construction des tournées se fait ensuite par balayage. Les nœuds sont balayés dans le sens horaire ou inversement (trigonométrique). A chaque étape, l'affectation du client considéré à la tournée en cours est faite dans la mesure où elle n'implique pas de violation de contraintes sinon c'est le retour vers le dépôt. Cette heuristique peut produire plusieurs résultats qui vont dépendre du sens de rotation choisi pour le balayage et du client de départ. Le déroulement de la méthode est illustré par la figure 2.4.

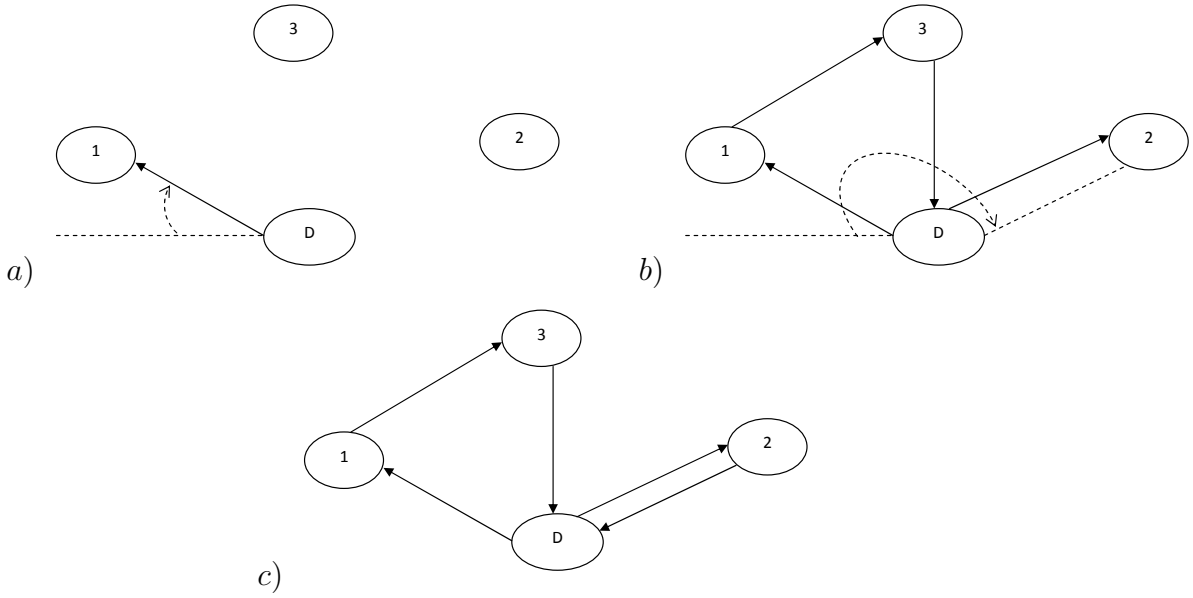


FIGURE *2.4: Déroulement de l'heuristique de Gillet et Miller*

a) Départ dans le sens horaire, le premier nœud rencontré est le nœud 1, il est donc inclus dans une première tournée partant du dépôt.

b) L'ajout du client 2 dans la tournée en cours entraîne la violation d'une contrainte, un retour au dépôt est effectué et une nouvelle tournée initiée avec le client 2.

c) L'algorithme se termine lorsque tous les nœuds ont été balayés.

L'heuristique de Fisher et Jaikumar [39] constitue un autre exemple. Elle propose de résoudre un problème d'affectation des sommets en "clusters". Une fois le partitionnement terminé, la route est construite dans chaque partition en résolvant le problème de voyageur de commerce (TSP) qui lui correspond. La création des groupes de sommets est effectuée en résolvant un problème d'affectation (GAP pour General Assignment Problem en anglais) tel qu'introduit par Ross et Soland [110]. Des zones de fortes concentrations sont définies et représentées par les clients se trouvant en leur centre. Le problème d'affectation est résolu en considérant une estimation du coût de la route comme coût d'affectation d'un client à une zone. Les poids des clients sont les quantités de marchandises qu'ils demandent et la capacité des zones est la capacité des véhicules.

2.4.1.5 Heuristiques Route First Cluster Second

A l'inverse des heuristique Cluster First Route Second, il s'agit de déterminer d'abord le séquencement des clients et de les partitionner ensuite en tournées. Ce paradigme a été proposé par Beasley [6]. Un problème de TSP est d'abord résolu avec l'ensemble des clients créant ainsi un tour géant. Ce tour géant est ensuite partitionné en tournées de manière optimale pour la séquence de clients qu'il présente. Ce partitionnement optimal est déterminé par la résolution d'un problème de plus court chemin dans un graphe auxiliaire. Beasley ne donne toutefois aucun résultat numérique, les premiers à en fournir sont Prins et al.[102].

2.4.2 Les procédures de recherche locale

Les procédures de recherche locales consistent, à partir d'une solution s existante, à explorer son voisinage $V(s)$ dans l'espoir d'y trouver une ou plusieurs solutions de meilleure qualité. Dans le cas du problème de tournées de véhicules, s'il est défini dans un graphe orienté $G(V, A)$, le voisinage peut être défini en terme de mouvements de suppression/ajout d'arcs dans la solution initiale. La méthode de recherche locale explore un ou plusieurs voisinages et remplace ensuite la solution courante par la meilleure solution voisine trouvée. Le processus se termine lorsqu'il n'est plus possible de trouver de meilleure solution dans le voisinage considéré. La méthode se trouve alors dans un optimum local, elle a en effet produit une solution qui se trouve être la solution de coût minimum dans son voisinage. La littérature foisonne de mouvements (voisinages), les plus classiques sont les mouvements *k-opt*, les mouvements *Or-opt*, les échanges de séquences et les chaines d'éjection.

Les mouvements de type *k-opt* tels que décrits par Lin et Kernighan [73] pour le TSP consistent à supprimer k arcs dans la solution courante et à les remplacer par k autres de

manière à reconstituer une solution faisable. La taille du voisinage est en $O(n^k)$. Le voisinage 2-*opt* est très populaire dans la littérature sur le VRP, Potvin et Rousseau [97] proposent de l'adapter pour qu'il opère entre deux routes distinctes. L'une des possibilités est nommée 2-*opt**, elle est illustrée par la figure 2.5.

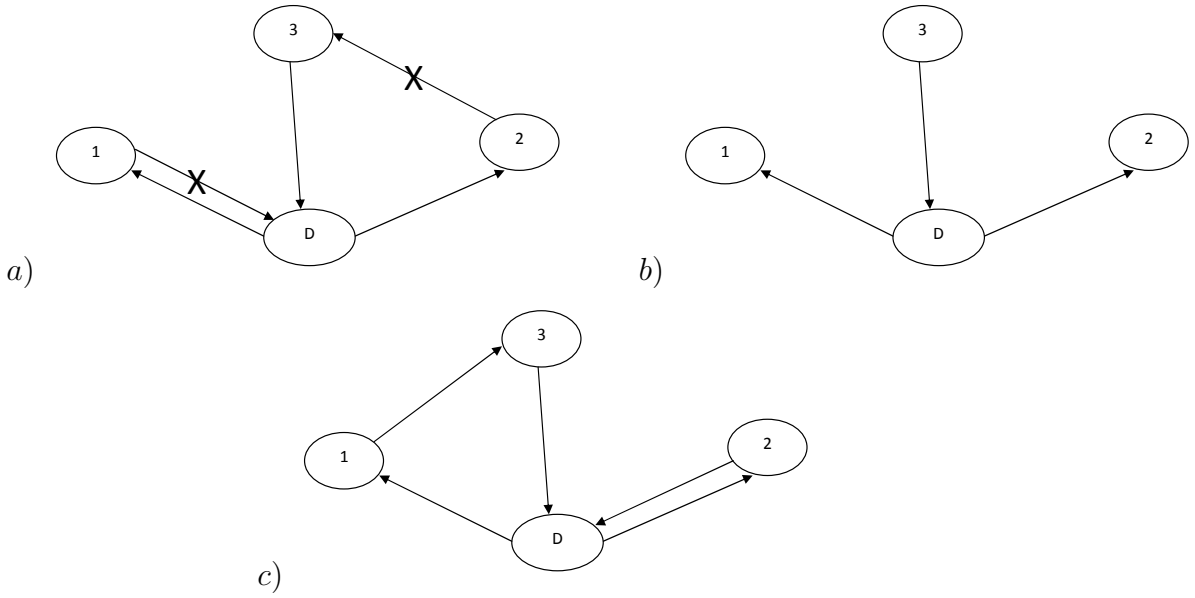


FIGURE *2.5: Application d'un mouvement* 2-*opt**

a) Suppression de deux arcs dans la solution,

b) Solution une fois les arcs supprimés,

c) Unique reconstruction possible.

Les mouvements de type *Or-opt* décrits pour la première fois dans la thèse de doctorat de Or [91] pour le TSP, consistent au déplacement d'une chaine de l clients servis consécutivement dans la solution en cours. Ce déplacement peut faire intervenir ou non le retournement de la séquence déplacée. Une fois qu'il n'est plus possible de trouver de mouvements améliorant pour une longueur de chaine de l, le processus est réitéré avec $l - 1$. Originalement, Or décrit ce mouvement pour $l = 3$. La généralisation pour le VRP dans laquelle la chaine de sommets peut également être déplacée d'une route à l'autre est le mouvement dit d'insertion (rencontré sous l'appellation "insert", "shift" ou encore "relocate" dans la littérature). Si deux séquences de clients sont échangées, on parle alors de mouvement de type "swap". Bräysy [11] propose un mouvement appelé "I-CROSS" qui généralise les précédents dans lequel les chaines de clients considérées, dont une peut être vide, peuvent être inversées. D'autres mouvements combinant les précédents et appelés "mouvements composés" peuvent être définis. Les chaines d'éjections en sont un exemple : si la route sensée recevoir la chaine

de client d'un mouvement d'insertion devient infaisable, alors un ou plusieurs des clients qui la composent sont éjectés vers une autre route. Le processus insertion-éjection est répété tant qu'une solution acceptable n'est pas trouvée ou que le nombre d'itérations effectuées atteint un maximum préalablement défini. Cette procédure, initialement proposée par Glover [52] pour le TSP, est notamment utilisée pour le VRP par Rego [104] et pour le VRPTW par Ibaraki et al. [63] qui la nomment "cyclic exchange". L'heuristique de Lin et Kerninghan [74] en est un autre exemple. Originalement proposée pour le TSP, elle fut considérée pendant longtemps comme la meilleure heuristique pour ce problème. La méthode consiste à déterminer à chaque étape de l'algorithme le nombre d'arcs qui doivent être échangés, faisant d'elle une procédure *k-opt* guidée.

2.4.3 Métaheuristiques

Les métaheuristiques sont des méthodes se voulant le plus générique possible, pouvant être utilisées pour la résolution de nombreux problèmes de nature différentes. Contrairement aux heuristiques précédemment décrites, elles ne comportent dans leurs principes généraux aucune spécificité relative au VRP. La plupart du temps, les métaheuristiques sont des algorithmes stochastiques alternant phases de diversification et d'intensification dans l'espace de recherche. Il en existe une très grande variété et certaines occupent une place importante dans la littérature relative au VRP. Elles sont décrites par la suite, regroupées en deux grandes catégories selon qu'elles sont à base de populations ou de recherche locale.

2.4.3.1 Métaheuristiques à base de population

Les métaheuristiques à base de population sont souvent inspirées de la nature et se basent sur des interactions entre agents. En font ainsi partie les algorithmes génétiques, les algorithmes de colonies de fourmis, les essaims particulaires, les colonies d'abeilles etc. Un très grand nombre de métaheuristiques à population ont été développées ces dernières années, rendant un inventaire exhaustif difficile à faire. Deux de ces métaheuristiques, très présentes dans la littérature sur le VRP, sont décrites ici : les colonies de fourmis et les algorithmes génétiques.

L'optimisation par colonie de fourmis (ACO pour Ant Colony Optimisation en anglais) a été introduite par Dorigo et al. [31], elle est inspirée de l'observation du comportement d'une colonie de fourmis dans son travail de recherche de nourriture. Les chercheurs se sont en effet aperçus que les fourmis finissent toujours par trouver le chemin le plus court entre une source de nourriture et la fourmilière. L'expérimentation montre que lorsqu'une fourmi a trouvé de la nourriture, elle dépose des phéromones sur le chemin du retour. Les autres fourmis qui

passent à proximité de cette piste de phéromones sont attirées et remontent ainsi la piste jusqu'à la nourriture. Sur le chemin du retour, elles émettent également des phéromones. Les phéromones sont volatiles et s'évaporent avec le temps, si une piste n'est pas rechargée suffisamment souvent elle finie par disparaître. L'attraction qu'a une fourmi pour une piste est directement proportionnelle à sa charge en phéromone. Le chemin de retour le plus rapide à la fourmilière est parcouru par plus de fourmis que les autres dans un même laps de temps. Par conséquent, ce chemin est à terme le plus chargé en phéromones et donc le plus attractif. Les autres chemins finissent par disparaître par manque de rechargement en phéromones et il ne reste alors que le chemin le plus court. Concrètement, l'algorithme fait l'analogie entre les chemins parcourus par les fourmis et les arcs du graphe du problème considéré. Une matrice de phéromones dans laquelle sont consignées les quantités de phéromones présentent sur chaque arc du graphe est utilisée. Cette quantité de phéromones permet de calculer la probabilité qu'une fourmi a de choisir l'arc concerné lorsqu'elle se trouve sur un sommet auquel il est connecté. La première fourmi choisi son chemin de manière probabiliste dans le graphe, à ce stade sur chaque sommet, tous les arcs ont la même probabilité d'être choisis. La fourmi poursuit les visites de clients tant que la route créée reste faisable au sens des contraintes du problème. Dès que ce n'est plus le cas, une nouvelle route est initiée. Une fois tous les clients affectés, une mise à jour des quantités de phéromones présentes sur les arcs appartenant à la solution que vient de créer la fourmi est effectuée. Cette mise à jour dite "mise à jour locale" est une première manifestation du phénomène d'évaporation des phéromones, elle permet d'empêcher que trop de fourmis ne prennent la même route et constitue donc une procédure de diversification. Une nouvelle fourmi est ensuite lancée et trace également sa solution qui sera mise à jour, cette opération est répétée autant de fois qu'il y a de fourmis dans la population. Une fois que l'ensemble d'une population de fourmis a parcouru le graphe, une mise à jour globale des quantités de phéromones est appliquée en marquant plus fortement les arcs qui appartiennent à la meilleure solution trouvée. Cette nouvelle mise à jour permet d'initialiser le réseau pour la prochaine population de manière à ce qu'elle explore le voisinage de la meilleure solution trouvée. L'algorithme se poursuit ainsi jusqu'à ce que le nombre de populations devant explorer le réseau soit atteint. Adaptée au TSP par Dorigo et Gambardella [30] puis au CVRP par Bullnheimer et al [12], l'optimisation par colonie de fourmis est ensuite appliquée à de nombreuses reprises avec succès. Citons Reimann et al. [106] et Yu et al. [124] pour le CVRP, Gajpal et Abad [43] et Zhang et al. [125] pour le VRPSPD, Reimann et al. [105] qui proposent un ACO unifié pour plusieurs variantes du VRP, ou encore Belmecheri et al. [8] qui l'appliquent à un VRP multi-attributs.

Les algorithmes génétiques introduits par Holland [62] sont basés sur l'évolution naturelle et la génétique. Lorsque les individus d'une espèce se reproduisent, ils engendrent des enfants

possédant une combinaison du patrimoine génétique de leurs deux parents. Ce patrimoine génétique peut également être affecté par des mutations qui apportent de nouvelles caractéristiques à la descendance. La sélection naturelle intervient en éliminant prématurément les individus qui ne sont pas adaptés à leur milieu. Les algorithmes génétiques établissent le parallèle entre un individu d'une population et une solution d'un problème d'optimisation. La solution est codée par un chromosome qui la représente. Une phase de sélection de parents est initiée pour la reproduction, un enfant pouvant ici avoir plus de deux parents. Une procédure de croisement est alors appliquée de manière à générer le chromosome de l'enfant à partir de ceux des parents. Une mutation est ensuite appliquée avec une certaine probabilité sur le génome de l'enfant avant que celui ci ne soit réintroduit dans la population. Une étape de sélection élimine ensuite la solution la moins performante de la population, l'algorithme itère ainsi jusqu'à ce qu'un critère d'arrêt soit atteint. Ce critère peut être un nombre d'itérations ou le non dépassement d'un pourcentage d'amélioration de la meilleure solution. L'utilisation de ces algorithmes pour le VRP doit beaucoup aux travaux de Prins [99] qui propose d'utiliser un chromosome dans lequel la solution est codée sous la forme d'un tour géant. Ce tour géant n'inclut pas les délimiteurs entre tournées dont la gestion s'avère problématique, principalement lors du croisement. L'évaluation de la solution est faite en plaçant les délimiteurs de tournée de manière optimale pour le séquencement des clients dans le tour géant. Ce placement optimal est déterminé par la résolution d'un problème de plus court chemin dans un graphe auxiliaire comme proposé par Beasley [6].

Pour être compétitives, les métaheuristiques à populations doivent comporter une procédure de recherche locale de manière à renforcer l'intensification. Les algorithmes génétiques qui en possèdent une sont appelés algorithmes "mémétique". C'est le cas de l'algorithme de Prins [99] ou plus récemment de celui de Vidal et al. [120] qui s'avère très compétitif sur le CVRP, le PVRP et le VRP multi-dépôts (MDVRP pour Multi Depot Vehicle Routing Problem).

2.4.3.2 Les métaheuristiques à base de recherche locale

Si les procédures de recherche locale permettent d'évoluer vers des optima locaux, la définition d'une stratégie d'exploration est nécessaire pour permettre une exploration plus exhaustive de l'espace de recherche. Le recuit simulé, proposé par Kirkpatrick et al. [65] explore le voisinage de la solution courante mais accepte potentiellement des solutions dégradantes selon une probabilité qui dépend d'un critère de "température". Plus la température est élevée et plus la probabilité d'accepter une solution non-améliorante est grande. L'algorithme débute avec une valeur de température élevée, celle-ci évolue ensuite (linéairement ou non) selon une règle de refroidissement à définir. La baisse de température conduit progressive-

ment à n'accepter plus que des solutions améliorantes. La recherche taboue introduite par Glover [50, 51, 53] explore le voisinage de la solution courante et progresse vers la solution voisine qui minimise la fonction objectif considérée. Tout mouvement susceptible de ramener l'algorithme vers une solution déjà explorée est proscrit afin de ne pas s'enfermer dans un optimum local. Le cœur de l'algorithme réside alors dans la gestion d'une liste de mouvements "tabous" dont la taille est définie par l'utilisateur. Toutefois, si un mouvement se trouvant dans cette liste est susceptible de mener vers une très bonne solution, un critère dit "d'aspiration" qui permet de ne pas l'ignorer. La méthode s'arrête lorsqu'un nombre maximal d'itérations est atteint. Les algorithmes basés sur la recherche taboue ont longtemps été les plus performants pour les problèmes de tournées de véhicules, citons notamment le TABUROUTE de Gendreau et at. [46] et le UTS (Unified Tabu Search) de Cordeau et al. [20]. Proposée par Feo et Resende [37, 38] la méthode GRASP (Greedy Randomized Adaptive Search Procedure) est une métaheuristique fonctionnant en deux étapes. La première étape consiste à générer une solution réalisable au moyen d'une heuristique gloutonne randomisée. La deuxième étape consiste en une recherche locale sur la solution trouvée pour atteindre un minimum local. Les deux étapes sont répétées en conservant la meilleure solution obtenue jusqu'à l'atteinte d'un critère d'arrêt. Ce critère peut être un nombre maximum d'itérations, un nombre d'itérations sans amélioration ou encore l'atteinte d'un seuil d'amélioration. La méthode ILS (Iterated Local Search), dont une récente revue de littérature est proposée par Lourenco et al. [76], est basée sur le principe de proximité des optima locaux "*proximate optimality principle*" établi par Glover [53]. Selon ce principe empirique, les optimum locaux seraient regroupés "*en grappes*" dans l'espace de recherche. Partant d'une solution générée par une heuristique gloutonne, une série de perturbations et de recherche locale sont appliquées sur la meilleure solution connue. Le dimensionnement des perturbations a un impact important sur la qualité de la méthode. Si la perturbation est trop faible, elle ne permet pas de sortir du bassin d'attraction de la meilleure solution connue qui sera alors retrouvée par la recherche locale. Inversement, si la perturbation est trop forte, la recherche quitte la "grappe" qu'elle explore et la méthode s'apparente davantage à un GRASP. Cette métaheuristique, bien que d'une grande simplicité, se révèle d'une efficacité redoutable. Toutefois, le nombre réduit de paramètres qu'elle comporte est sensible et tout particulièrement celui régissant la taille de la perturbation. Pour le VRP, Prins [100] propose une adaptation de cette méthode obtenant des résultats remarquables. Subramanian et al. [114] proposent une implémentation parallèle efficace de ce principe pour le VRPSPD.

2.5 Synthèse

Dans ce chapitre, sont présentées les principales variantes du problème de tournée de véhicules. L'accent est mis ensuite sur les modèles se rapprochant le plus du problème qui nous intéresse. Certaines caractéristiques du problème de tournées de convoyeurs de fonds sont présentes dans la littérature existante. Toutefois, les contraintes de sécurité telles qu'elles sont rencontrées dans le cas qui nous intéresse n'ont encore jamais été proposées. Il est donc nécessaire de proposer un nouveau modèle combinant les caractéristiques existantes voulues et incluant de nouvelles contraintes de sécurité. Les méthodes de résolutions de types exacte et approchées sont également présentées. La taille des instances qui doivent être traitées en pratique oriente nos travaux sur des méthodes approchées de type heuristiques ou métaheuristiques. Le chapitre suivant a pour but de proposer un modèle mathématique qui réponde aux spécificités du problème que nous rencontrons en se basant sur cet état de l'art. Il propose également deux méthodes de résolution heuristiques inspirées de celles inventoriées dans ce chapitre. Le chapitre 4 est quand à lui consacré au développement d'une métaheuristique basée sur la méthode ILS qui est présentée dans la section 2.4.3.2 de ce chapitre.

Chapitre 3

Problème de tournées de véhicules périodique avec variation des temps de service

3.1 Introduction

Dans ce chapitre, nous proposons une modélisation mathématique du problème de convoyeurs de fonds en nous appuyant sur la description et l'état de l'art effectué précédemment. Le modèle élaboré est ensuite discuté puis le sous-problème d'ajustement des temps de départ qu'il contient est étudié et sa complexité est déterminée. Le modèle obtenu est alors résolu de manière exacte grâce au solveur commercial CPLEX sur de petites instances. Deux heuristiques constructives sont introduites, évaluées grâce aux résultats obtenus par CPLEX puis comparées entre elles sur de plus grands jeux de tests dérivés de la littérature.

3.2 Description

Lorsqu'il s'agit de tournées de véhicules et de sécurité, nous avons vu qu'il n'existait que des modèles basés sur des irrégularités d'arcs (i.e. où l'on évite d'utiliser plusieurs fois un même arc). De l'aveu même de sociétés opérants dans la sécurité, un convoi de marchandises sensibles n'est jamais autant vulnérable que lorsqu'il se trouve à l'arrêt. Il semblerait donc que ce ne soit pas nécessairement sur les routes, lorsque le véhicule est en mouvement, qu'il faille le plus protéger le convoi mais bien pendant les livraisons, lorsqu'il est à l'arrêt. Une manière de sécuriser les arrêts est de les rendre imprévisibles et ainsi de limiter la

possibilité pour d'éventuels assaillants de planifier une attaque. Pour rendre imprévisibles ces arrêts, il faut en varier les horaires d'une période de livraison sur l'autre. Un autre point important est le respect de fenêtres horaires de livraison car elles sont en pratiques exigées par la plupart des clients. Les banques, par exemple, sont servies de préférence en dehors des horaires d'ouverture aux clients pour limiter les dommages liés à une attaque. Les périodes d'immobilité étant nuisibles à la sécurité du convoi, aucun temps d'attente ne peut être toléré. Le respect strict de ces fenêtres horaires est donc impératif. Il n'existe dans la littérature aucun travail similaire. Le problème opposé, celui qui consiste à définir un plan de livraison qui soit le plus régulier possible d'une période sur l'autre, a été étudiée par Groer et al. [58] pour la société United Parcel Service company (UPS). Ce problème, appelé problème de tournées de véhicules régulière (ConVRP pour Consistent Vehicle Routing Problem en anglais) se résume à définir des routes sur plusieurs périodes de livraison de façon à ce qu'un client, lorsqu'il est livré, le soit par le même chauffeur et dans un intervalle de temps qui ne soit pas trop grand. Ils introduisent ainsi un critère de qualité de service dans le problème de tournées de véhicules périodiques. Les auteurs proposent un algorithme de type record-to-record pour résoudre ce problème et conduisent leurs expérimentations sur des instances basées sur celles de Christofides et Eilon [16] modifiées pour correspondre au ConVRP et sur des jeux de données contenant jusqu'à 505 clients fournis par UPS.

3.3 Modélisation

Nous proposons ici d'introduire un nouveau modèle que nous appellerons problème de tournées de véhicules périodiques avec dispersion des instants de services (PVRPTS pour Periodic Vehicle Routing Problem with Time Spread en anglais). Le PVRPTS peut être défini sur un graphe orienté $G = (V, A, C)$ et un horizon de planification P de p périodes ou "jours". L'ensemble des sommets $V = \{0, 1, ..., n+1\}$ comprend n clients numérotés à partir de 1 et deux sommets 0 et $n+1$ représentant le dépôt où est basé un ensemble K de véhicules identiques de capacité W. A est l'ensemble des arcs (i, j) avec $i, j \in V$. Le coût d'utilisation d'un arc (i, j) est c_{ij} et d_{ij} désigne le temps nécessaire au parcours de cet arc. Le service à chacun des nœuds i consiste à délivrer une quantité q_i de marchandises, pendant un temps s_i, sans commencer plus tôt que a_i et plus tard que b_i. Chez un même client et au cours de deux périodes distinctes, ces livraisons doivent temporellement être espacées d'au moins ϵ. Contrairement au VRPTW, il n'est pas possible d'attendre chez le client. Une route valide est alors un circuit qui quitte le dépôt, livre un sous-ensemble de clients pendant leurs disponibilités avec une quantité totale de marchandises qui ne dépasse pas la capacité du véhicule, et retourne au dépôt. Le problème est NP-difficile car il généralise le VRP qui est

NP-difficile. En effet, le cas ou l'on ne considère qu'une seule période ($p = 1$) et des fenêtres de livraison grandes $a_i = 0, b_i = +\infty, \forall i \in V$ modélise le VRP. Il peut être modélisé comme un programme linéaire mixte (MIP pour Mixed Integer Program en anglais). Les variables binaires y_i^{kp} sont égales à 1 si le client i est servi par le véhicule k pendant la période p, les variables binaires x_{ij}^{kp} sont égales à 1 si l'arc (i, j) est traversé par le véhicule k pendant la période p. Les variables réelles non-négatives t_i^{kp} représentent l'instant d'arrivée du véhicule k chez le client i pendant la période p. M désigne ici une grande constante positive.

$$\min z = \sum_{p \in P} \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^{kp} \tag{3.1}$$

$$\sum_{k \in K} y_i^{kp} = 1 \qquad \forall i \in V \setminus \{0, n+1\}, \forall p \in P, \tag{3.2}$$

$$\sum_{k \in K} y_0^{kp} \leq |K| \qquad \forall p \in P, \tag{3.3}$$

$$\sum_{i \in V} x_{ij}^{kp} = y_j^{kp} \qquad \forall j \in V \setminus \{0\}, \forall k \in K, \forall p \in P, \tag{3.4}$$

$$\sum_{j \in V} x_{ij}^{kp} = y_i^{kp} \qquad \forall i \in V \setminus \{n+1\}, \forall k \in K, \forall p \in P, \tag{3.5}$$

$$\sum_{i \in V} x_{i,n+1}^{kp} = 1 \qquad \forall k \in K, \forall p \in P, \tag{3.6}$$

$$\sum_{i \in V} q_i y_i^{kp} \leq W \qquad \forall k \in K, \forall p \in P, \tag{3.7}$$

$$t_i^{kp} + s_i + d_{ij} - M(1 - x_{ij}^{kp}) \leq t_j^{kp} \qquad \forall i, j \in V, \forall k \in K, \forall p \in P, \tag{3.8}$$

$$t_i^{kp} + s_i + d_{ij} + M(1 - x_{ij}^{kp}) \geq t_j^{kp} \qquad \forall i, j \in V, \forall k \in K, \forall p \in P, \tag{3.9}$$

$$a_i \leq t_i^{kp} \leq b_i \qquad \forall i \in V, k \in K, \forall p \in P, \tag{3.10}$$

$$t_i^{kp} - t_i^{k'p'} + M(1 - X_i^{pp'kk'}) \geq \epsilon \qquad \forall i \in V \setminus \{0, n+1\}, \forall k, k' \in K, \forall p, p' \in P, p \neq p', \tag{3.11}$$

$$t_i^{k'p'} - t_i^{kp} + M X_i^{pp'kk'} \geq \epsilon \qquad \forall i \in V \setminus \{0, n+1\}, \forall k, k' \in K, \forall p, p' \in P, p \neq p', \tag{3.12}$$

$$y_i^{kp}, x_{ij}^{kp} \in \{0, 1\} \qquad \forall i, j \in V, i \neq j, \forall k \in K, \forall p \in P, \tag{3.13}$$

$$X_i^{pp'kk'} \in \{0, 1\} \qquad \forall i \in V, \forall k, k' \in K, \forall p, p' \in P, p \neq p', \tag{3.14}$$

$$t_i^{kp} \in \mathbb{R}^+ \qquad \forall i \in V, \forall k \in K, \forall p \in P. \tag{3.15}$$

L'objectif (3.1) est de minimiser le coût total engendré par l'utilisation des arcs pendant l'horizon de planification. Les contraintes classiques du VRP (3.2) assurent que chaque client soit visité exactement une fois par un véhicule à chaque période. Les Contraintes (3.3) vérifient que le nombre de départs du dépôt à chaque période ne dépasse pas la taille de la

flotte. Les contraintes (3.4) et (3.5) sont les contraintes de flots classiques et les contraintes (3.6) assurent que toutes les routes finissent au nœud dépôt $n+1$. La capacité des véhicules est respectée grâce aux contraintes (3.7).

Le rôle des contraintes (3.8) et (3.9) est de rendre les dates d'arrivées cohérentes dans la route et d'éviter la formation de sous-tours. En effet, si $x_{ij}^{kp} = 1$ (i.e si le véhicule k visite le nœud j immédiatement après le nœud i au cours de la période p), les deux contraintes sont équivalentes à $t_i^{kp} + s_i + d_{ij} = t_j^{kp}$ (rappelons ici que les temps d'attentes ne sont pas autorisés) alors que si $x_{ij}^{kp} = 0$ elles sont satisfaites de manière triviale. Les contraintes (3.10) assurent que l'arrivée chez chaque client se fait bien pendant la fenêtre de temps. Comme expliqué dans le livre de Toth and Vigo [117], la grande constante M dans (3.8)-(3.9) peut être remplacée par $\max\{b_i + s_i + d_{ij} - a_j, 0\}$ pour chaque arc (i, j), afin d'éviter les problèmes de précision numérique à la résolution.

Les contraintes spécifiques d'espacement (3.11)-(3.12) sont la traduction linéaire des contraintes suivantes :

$$\left| t_i^{kp} - t_i^{k'p'} \right| \geq \epsilon \qquad \forall i \in V \setminus \{0, n+1\}, \forall k, k' \in K, \forall p, p' \in P, p \neq p'. \tag{3.16}$$

La linéarisation utilise un jeu de variables binaires $X_i^{pp'kk'}$ qui valent 1 si $t_i^{kp} \geq t_i^{k'p'}$. Pour $i \in V \setminus \{0, n+1\}, k, k' \in K, p, p' \in P, p \neq p'$, il y a deux possibilités :

1. Si $X_i^{pp'kk'} = 1$, la contrainte (3.11) devient $t_i^{kp} - t_i^{k'p'} \geq \epsilon$ et il est trivial que la contrainte (3.12) est satisfaite.
2. Si $X_i^{pp'kk'} = 0$, il est trivial que la contrainte (3.11) est vérifiée et (3.12) devient $t_i^{k'p'} - t_i^{kp} \geq \epsilon$.

Grâce aux contraintes (3.10) sur les t_i^{kp}, M peut être remplacé dans les équations (3.11) et (3.12) par $\max_{i \in V}\{b_i - a_i\} + \epsilon$.

Une condition nécessaire pour la faisabilité du problème est que la plus petite fenêtre horaire fournisse suffisamment d'espace pour y réaliser m visites, e.g. :

$$\epsilon \leq \frac{min_{i \in V}\{b_i - a_i\}}{m-1} \tag{3.17}$$

3.4 Problème de détermination des dates de départ de chaque route

3.4.1 Description

Dans le modèle décrit précédemment, les instants de départ des routes, les t_0^{kp}, sont des variables de décision. Pour le développement de méthodes de résolution, il est important de savoir s'il est possible d'en déterminer facilement les valeurs conduisant à une solution réalisable. Une solution du $PVRPTS$ se compose de m périodes chacune composée de routes $r \in R$ avec $R \subseteq \mathscr{R}$ où $\mathscr{R}$ est l'ensemble des routes qu'il est possible d'extraire de G. Comme le montre Savelsbergh [112], il est facile de déterminer pour une route si il existe un instant de départ tel qu'il n'y ait pas de temps d'attente, i.e. les contraintes (3.8)-(3.9) sont vérifiées. On cherche à savoir ici s'il est également possible de le faire pour que les contraintes d'écarts (3.11)-(3.12) soient vérifiées.

Si il n'y a aucun temps d'attente dans la route r, il est possible de calculer un départ au plus tôt t_0^{r-} et un départ au plus tard t_0^{r+} pour cette route. Le départ au plus tôt t_0^{r-} correspond au premier instant de départ du dépôt qui respecte les fenêtres horaires chez les clients et ne génère pas de temps d'attente le long de la route r. Le départ au plus tard t_0^{r+} est le dernier départ du dépôt qui permet de ne pas violer de fenêtre horaire le long de la route r. Comme il n'y a aucun temps d'attente le long de cette route, chaque décalage effectué entre t_0^{r-} et t_0^{r+} sur le départ du dépôt induit exactement le même décalage sur l'arrivée de tous les clients de la route. Pour une route r donnée dont le départ est fixé au plus tôt, i.e. à t_0^{r-}, le décalage Δ_r possible est compris entre 0 et $\Delta_r^{max} = t_0^{r+} - t_0^{r-}$. Partant d'une solution du $PVRPTS$ dont toutes les routes r ont leurs départs initialisés au plus tôt, i.e. à t_0^{r-}, il s'agit alors de savoir s'il existe des décalages $\Delta_r \in [0, \Delta_r^{max}]$ tels que cette solution soit faisable.

Ici la séquence des routes est fixée, il ne s'agit plus à cette étape d'optimiser la distance parcourue mais seulement de décider des instants de départ comme l'illustre la figure 3.1. Ce problème de décision, que nous nommerons Starting Time Problem (STP), consiste à déterminer les instants de départ pour chacune des routes $r \in R$ en définissant leurs Δ_r afin que les arrivées chez un même client d'une route à l'autre soit espacées temporellement d'au moins ϵ. Si $R(i)$ est l'ensemble des routes de R qui contiennent le client i et si t_i^r est l'instant d'arrivée au client i dans la route r lorsque le départ de cette route est pris le plus tôt possible, le problème STP peut être écrit comme suit :

(STP) : Fixer les $\Delta_0, \Delta_1, ..., \Delta_r$ avec $r \in R$ tels que :

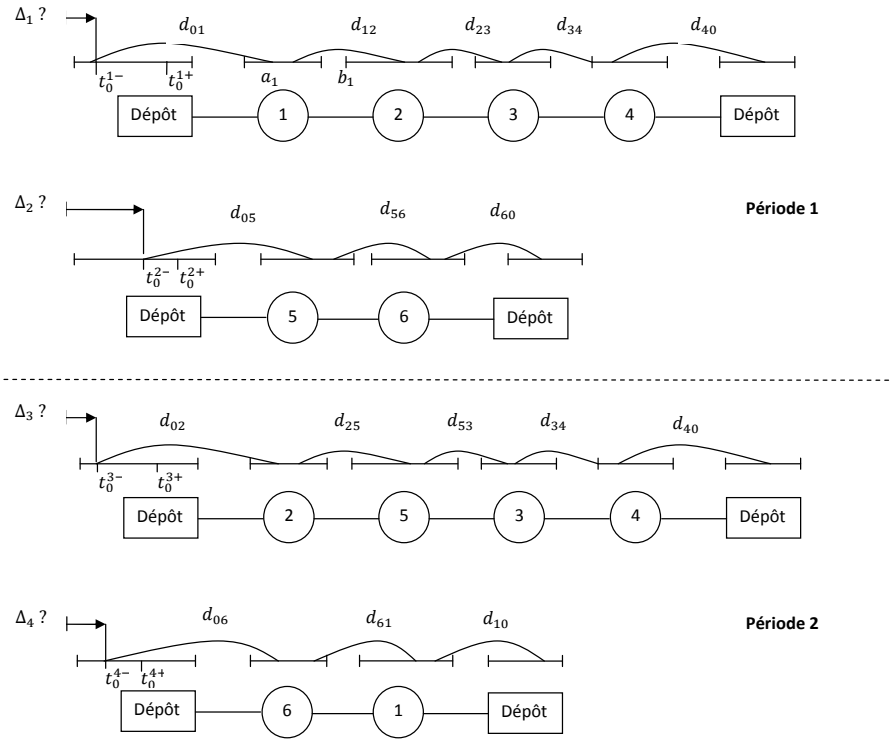


FIGURE *3.1: Exemple avec six clients et deux périodes*

$$\left|(t_i^r + \Delta_r) - (t_i^{r'} + \Delta_{r'})\right| \geq \epsilon \qquad \forall i \in V, \forall r, r' \in R(i), r \neq r', \tag{3.18}$$

$$\Delta_r \leq \Delta_r^{max} \qquad \forall r \in R, \tag{3.19}$$

$$\Delta_r \in \mathbb{R}^+ \qquad \forall r \in R. \tag{3.20}$$

3.4.2 Complexité

Le STP étant modélisé mathématiquement, il reste à en déterminer la complexité. Ceci est fait avec le théorème et la preuve qui suivent.

THEOREME 3.1 *STP est NP-complet*

Preuve 1 *Vérifier qu'une solution satisfait STP se fait en $O(np^2)$, les instants d'arrivée sont vérifiés pour chaque client et chaque paire de période. La vérification en temps polynomial implique que que STP est dans NP.*

On propose à présent de réduire le problème de coloration de graphe à ce problème : Soit $G = (V', E)$ *un graphe à colorier. On construit un STP de la manière suivante : A chaque sommet* i *de* V' *on associe une route* r_i *dans* R*. Pour chaque arrête* (i, j) *de* E*, on associe un client* n_{ij} *dans* V *que l'on affecte respectivement aux routes* r_i *et* r_j*.*

On considère de plus que pour chaque client i *dans chaque route* r *on a :*

$$t_i^r = 0 \quad \forall i \in V, \forall r \in R. \tag{3.21}$$

On impose ensuite un espacement temporel d'au moins une unité de temps des instants d'arrivée chez un client pour deux routes distinctes (ceci revient à interdire que deux sommets adjacents dans le graphe $G = (V', E)$ *aient la même couleur),*

$$\left|(t_i^r + \Delta_r) - (t_i^{r'} + \Delta_{r'})\right| \geq 1 \qquad \forall i \in V, \forall r, r' \in R(i), r \neq r'. \tag{3.22}$$

Finalement, soit k *un entier positif tel que* $k \geq 3$*, on impose que :*

$$\Delta_r \leq k - 1 \quad \forall r \in R. \tag{3.23}$$

La subdivision équidistante de pas 1 de l'intervalle $[0, k-1]$ *est une partie finie qui contient* k *éléments (i.e.* $\{0, 1, \ldots, k-1\}$*). Il est donc possible de définir au plus* k *instants de départs distincts distants de 1 deux à deux.*

Le problème de coloration est satisfaisable si STP l'est. La construction d'une instance de STP à partir d'une instance de coloration de graphe est faite en temps polynomial. Le problème de k-coloration est NP-complet pour $k \geq 3$*, donc STP est NP-complet ce qui termine la preuve.*

La figure 3.2 donne une représentation de la réduction du problème de coloration de graphe au STP utilisé pour la preuve du théorème 3.1. Les sommets du graphe à colorier deviennent les routes du STP et les arrêtes deviennent les clients. Une arrête entre deux sommets du graphe du problème de coloration signifie que les deux routes qu'ils représentent ont un client en commun. Ce résultat de complexité nous renseigne d'avantage sur la difficulté du *PVRPTS* qui contient un problème de décision NP-complet comme sous-problème.

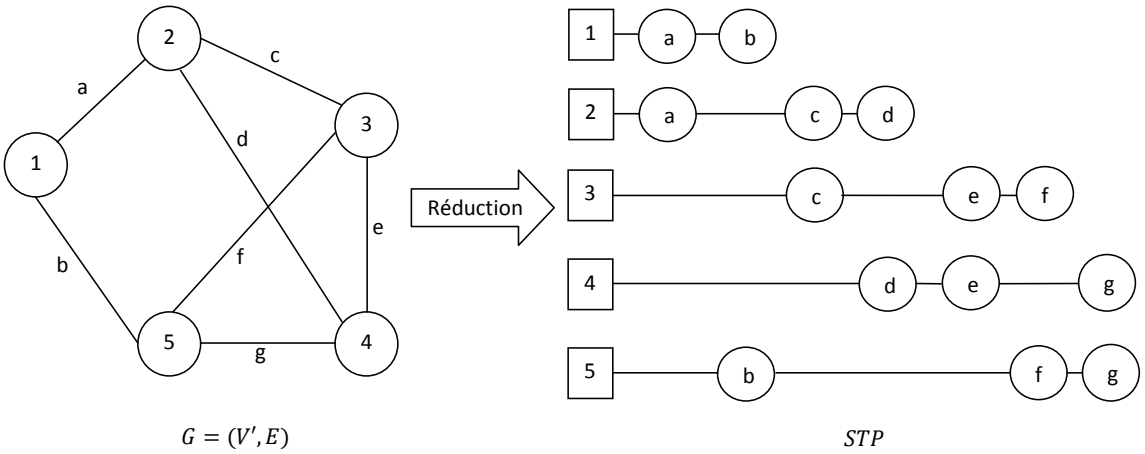


FIGURE *3.2: Représentation du graphe $G = (V', E)$ du problème de coloration et de l'instance construite par la réduction au STP.*

3.5 Heuristiques constructives

3.5.1 Motivations

La première approche que nous avons eu de ce problème est une approche heuristique. Le but étant de pouvoir fournir des solutions valides dans un temps très court. Deux façons d'aborder le problème ont été retenues : La première est une approche séquentielle consistant à construire les périodes de livraison les unes après les autres. Chaque décision prise pour une période de livraison ayant un impact sur les suivantes. La seconde consiste à séparer en classes temporelles la livraison à chaque période de manière à "casser" l'interdépendance qui existe entre ces périodes et de pouvoir ensuite les résoudre séparément. Cette section est dédiée à la présentation de ces deux approches ainsi qu'à leur évaluation numérique.

3.5.2 Heuristique de construction séquentielle

L'idée ici est de considérer le problème une période après l'autre. La première période est résolue comme un problème de VRPTW dans lequel les temps d'attente ne sont pas autorisés. Dans ce problème, on considère alors les fenêtres horaires originales des clients et on le résout sans tenir compte d'aucune autre période. Une fois la résolution de la première période terminée, on reconstruit pour chaque client i une nouvelle fenêtre horaire de livraison pour la seconde période. Cette nouvelle fenêtre de livraison dépend de l'instant t_i^1 auquel le client i a été servi au cours de la première période. Il ne faut en effet pas qu'une livraison ait

lieu à moins de ϵ durant la seconde période. La fenêtre horaire allant de $t_i^1 - \epsilon$ à $t_i^1 + \epsilon$ est donc soustraite à la fenêtre horaire originale. De cette opération peut résulter deux types de fenêtres horaire différents : Soit la partie retirée se trouve à une extrémité de la fenêtre horaire originale qui est alors simplement tronquée. Soit la partie retirée se trouve à l'intérieur de la fenêtre horaire originale et dans ce cas l'opération conduit à produire une fenêtre horaire discontinue. La figure (3.3) illustre ce principe de construction pour la période $p+1$ d'un client i.

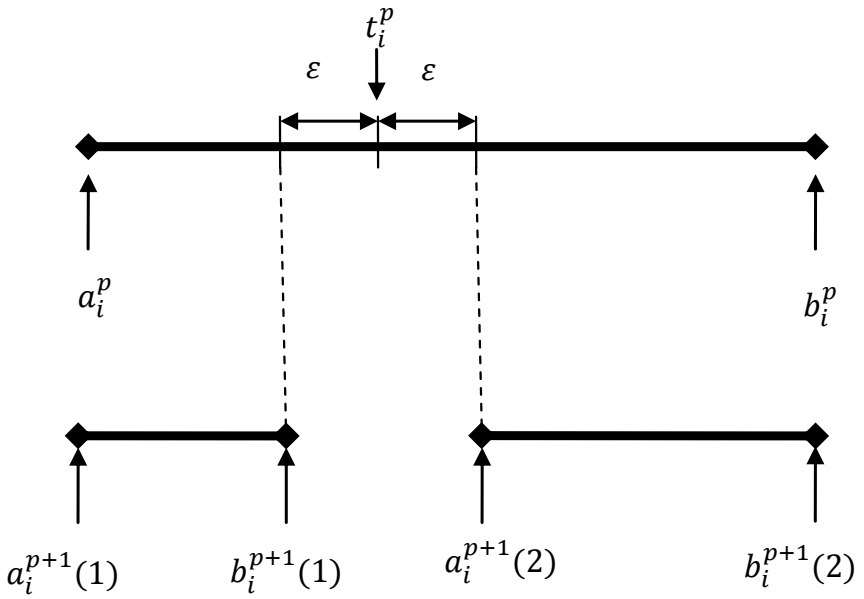


FIGURE *3.3: Construction de la nouvelle fenêtre horaire du client i pour la période* $p+1$

L'heuristique d'insertion de Solomon pour le VRPTW est ensuite utilisée pour résoudre la seconde période. Une fois la seconde période résolue, de nouvelles fenêtres horaires pour la période suivante sont générées et la résolution effectuée. Le processus s'arrête lorsque le nombre de périodes voulu est atteint ou lorsqu'il n'est plus possible d'en produire de nouvelle. En effet, il est possible au cours de l'algorithme qu'un ou plusieurs clients n'aient plus de fenêtres horaires réalisables. Cette éventualité nous amène à définir les conditions de résolubilité d'une instance par cette heuristique :

Lors de la construction des fenêtres horaires de la période suivante, il faut qu'il reste assez de fenêtres horaires pour chaque client i. Pour obtenir m périodes, il faut faire appel $m-1$ fois à la procédure de construction de fenêtres horaires. Au cours de cette procédure, une largeur d'au plus 2ϵ est enlevée à chaque fenêtre horaire. La plus petite fenêtre horaire de

l'instance est $min_{i \in V}\{b_i - a_i\}$, il faut alors que :

$$min_{i \in V}\{b_i - a_i\} > 2\epsilon(m-1) \tag{3.24}$$

On en déduit la condition suivante sur ϵ :

$$\epsilon < \frac{min_{i \in V}\{b_i - a_i\}}{2(m-1)} \tag{3.25}$$

Que l'on peut exprimer en fonction de ϵ_{max} :

$$\epsilon < \frac{\epsilon_{max}}{2} \tag{3.26}$$

L'heuristique H1 ne peut trouver une solution que si ϵ est choisi tel que : $\epsilon < 0.5\epsilon_{max}$.

Les nouvelles fenêtres horaires obtenues au cours de l'algorithme peuvent être discontinues et aucun temps d'attente n'est admis. Une adaptation de l'heuristique d'insertion gloutonne de Solomon est donc nécessaire. Lors de l'évaluation de la faisabilité et du coût d'insertion pour chaque nœud à chaque position, il faut être en mesure d'affirmer si la route résultante peut être effectuée sans temps d'attente. Les variables "Forward Time Slack" et "Backward Time Slack" initialement introduites par Savelsbergh [112] sont alors utilisées pour faire cette vérification en temps constant. Les fenêtres de temps discontinues sont traitées de la manière suivante : Chaque partie de fenêtre horaire est considérée comme un seul client. Dès qu'une partie de fenêtre horaire est insérée, les autres correspondants au même client sont retirées de la liste des candidats à l'insertion et mémorisés pour la construction de la période suivante. L'heuristique d'insertion gloutonne de Solomon ainsi modifiée peut être implémentée soit de manière classique en ne considérant qu'une seule route à la fois, soit de manière "parallèle" en créant plusieurs routes qui sont remplies de manière simultanées. L'algorithme 1 décrit la version parallèle basée sur celle introduite par Potvin et Rousseau [98] pour le VRPTW.

Algorithme 1 Insertion Parallèle

1: R=un ensemble de routes ne contenant que les dépôts
2: V=un ensemble de nœuds non affectés
3: **tant que** $V \neq 0$ **faire**
4: $C_2^* \leftarrow -\infty$
5: **pour** $j \in V$ **faire**
6: $C_1^* \leftarrow +\infty$
7: **pour** $r \in R$ **faire**
8: **pour** $i \in r$ **faire**
9: **si** $Faisable(i, j, r)$ et $C_1(i, j, r) < C_1^*$ **alors**
10: $r_1^* \leftarrow r$
11: $i_1^* \leftarrow i$
12: $j_1^* \leftarrow j$
13: $C_1^* \leftarrow C_1(i, j, r)$
14: **fin si**
15: **fin pour**
16: **fin pour**
17: **si** $C_2(i_1^*, j_1^*, r_1^*) < C_2^*$ **alors**
18: $r_2^* \leftarrow r_1^*$
19: $i_2^* \leftarrow i_1^*$
20: $j_2^* \leftarrow j_1^*$
21: $C_2^* \leftarrow C_2(i_1^*, j_1^*, r_1^*)$
22: **fin si**
23: **fin pour**
24: $Inserer(i_2^*, j_2^*, r_2^*)$
25: $Retirer(j_2^*, V)$
26: **fin tant que**
27: **retourner** R

3.5.3 Heuristique de construction parallèle

Une autre manière de construire une solution est d'effectuer préalablement le découpage des fenêtres horaires en parties espacées de ϵ. Il est alors possible de créer à l'aide de ces parties autant de problèmes de VRPTW sans temps d'attente que de périodes requises. Ces problèmes peuvent être résolus séparément dans la mesure où les contraintes qui génèrent la dépendance entre les routes sont ici satisfaites par construction. Il y a un grand nombre

de possibilités pour recombiner toutes les sous-fenêtres. En effet, il est possible de produire $m!^{n-1}$ configurations. Dans une première implémentation, nous allons considérer que pour chaque client i, sa fenêtre horaire est découpée en m parties de même longueur δ_i avec :

$$\delta_i = \frac{(b_i - a_i) - (m-1)\epsilon}{m} \tag{3.27}$$

Où m est le nombre de périodes considérées ($Card(P)$). La figure (3.4) illustre la procédure de découpe d'une fenêtre horaire de service d'un client i. Une fois la découpe effectuée, les sous-fenêtres horaires sont affectées de manière aléatoire à une période du problème. Chaque période représente alors un problème de VRTW sans temps d'attente qui peut être résolu en utilisant l'heuristique basée sur celle de Solomon présentée en 2.4.1.3. La procédure d'affectation/résolution est répétée jusqu'à ce qu'un nombre maximum d'itérations soit atteint.

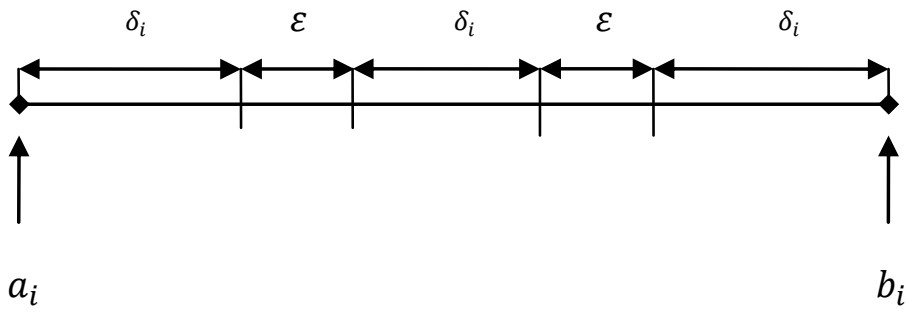


FIGURE *3.4: Génération des sous-fenêtres de service pour le client i sur 3 périodes*

3.5.4 Expérimentation et résultats

Comme il n'existe pas d'instances pour le *PVRPTS* dans la littérature, les tests sont effectués sur les instances pour le VRPTW de Solomon modifiées pour correspondre à notre problème. Ce jeu de tests est composé de six groupes de problèmes. Ils sont spécifiquement conçus pour reproduire les facteurs qui affectent en pratique le comportement des algorithmes de résolution. Les données de chaque problème sont le nombre de clients, le nombre de véhicules à considérer et leur capacité. Pour chaque client i sont définis : ses coordonnées x_i et y_i dans le plan euclidien, sa demande q_i, sa fenêtre horaire de livraison $[a_i, b_i]$ et pour finir son temps de service s_i. Les groupes R1 et R2 contiennent des coordonnées générées aléatoirement, les groupes C1 et C2 contiennent des clients regroupés en "grappes" et les groupes RC1 et RC2 contiennent un mélange de clients en grappes et de clients générés aléatoirement. Les

problèmes de type "1" ont un horizon de planification court et ne peuvent contenir qu'un faible nombre de clients par route. Inversement, les problèmes de type "2" ont un horizon de planification étendu qui permet de servir plusieurs clients avec le même véhicule. Certains problèmes ont des fenêtres horaires très étroites alors que d'autres en possèdent de plus larges. Les problèmes contiennent 100 clients et les temps de trajet entre clients sont égaux aux distances euclidiennes qui les séparent. Pour générer un problème de *PVRPTS* à partir de ces données, il faut considérer les n premiers clients, ajouter un nombre p de périodes et une valeur de ϵ telle que $\epsilon = \lfloor \gamma \cdot \epsilon_{max} \rfloor$ avec $\gamma \in [0, 1]$. Dans un premier temps, afin d'évaluer nos heuristiques face aux solutions optimales, les tests sont conduits sur de petites instances. Pour pouvoir comparer les deux heuristiques entre elles, les tests sont ensuite effectués sur de plus grosses instances avec des tailles plus proches des cas pratiques.

3.5.5 Résultats sur les petites instances

Les instances utilisées contiennent $n = 5$ clients, qui sont les cinq premiers clients des instances de Solomon, avec un nombre de périodes $p = 3$. La table 3.1 donne les résultats de ces tests pour $\epsilon = \lfloor \epsilon_{max} \rfloor$. Les colonnes **Instance** et ϵ donnent respectivement le problème et l'écart minimum considéré. Les résultats obtenus par H1 et H2 sont reportés ensuite, le coût est donné dans la colonne **Coût** et le temps de calcul en millisecondes dans la colonne **Tps(ms)**. Pour H2, le résultat donné est obtenu pour une exécution avec 100 itérations. Pour H1 est également reporté dans la colonne **%H2** le pourcentage d'écart avec H2 calculé comme ($Coût$(H1)-$Coût$(H2))/$Coût$(H2) $\times$100. Lorsque l'heuristique ne retourne pas de solution réalisable, le signe $-$ est indiqué. Comme ϵ est choisi tel que $\epsilon = \lfloor \epsilon_{max} \rfloor$, l'heuristique H1 n'offre pas de garantie de résolution et ne parvient à résoudre que trente des cinquante-six instances proposées. Toutefois, lorsque H1 produit une solution, celle-ci est en moyenne meilleure que celle de H2 (3.27%). Les temps de résolution sont quant à eux très courts, en dessous la milliseconde pour les deux méthodes.

Les instances considérant $\epsilon = \lfloor \epsilon_{max} \rfloor$ ne permettant pas de comparer équitablement les heuristiques H1 et H2, des tests sont effectués avec une valeur de ϵ fixé à $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$. Afin de voir si la formulation du *PVRPTS* en programme linéaire mixte peut être exploitée en pratique, un solveur commercial est également utilisé. La taille des instances proposées n'est pas aussi petite qu'elle en a l'air car chaque client doit être visité au cours de chaque période. Le nombre de variables du programme linéaire mixte correspond alors à $5 \times 3 = 15$ clients, ce qui est une limite classique lorsque l'on tente de résoudre un problème de tournées de véhicules directement avec un solveur commercial. Le solveur utilisé dans notre cas est CPLEX 12.4. Les résultats de ces tests sont reportés dans la table 3.2. Pour chacune des

		H2		H1		
Instance	ϵ	**Coût**	**Tps(ms)**	**Coût**	**Tps(ms)**	**%H2**
C101	22	392.02	0.16	254.22	0.02	-35.15
C102	27	169.98	0.29	212.47	0	24.99
C103	27	176.36	0.3	212.47	0	20.48
C104	27	167.82	0.33	212.47	0	26.6
C105	45	399.06	0.16	223.97	0	-43.88
C106	22	327.97	0.17	254.22	0	-22.49
C107	89	354.72	0.16	-	0	-
C108	91	399.06	0.16	223.97	0	-43.88
C109	179	384.98	0.16	-	0	-
C201	80	708.96	0.16	-	0	-
C202	80	336.46	0.27	319.6	0.01	-5.01
C203	80	334.04	0.27	319.6	0.01	-4.32
C204	80	300.04	0.27	319.6	0.01	6.52
C205	159	618.14	0.16	-	0	-
C206	177	567.07	0.16	346.75	0	-38.85
C207	88	527.32	0.16	406.59	0	-22.9
C208	319	657.89	0.16	-	0	-
R101	5	607.25	0.16	-	0	-
R102	5	424.72	0.28	479.77	0	12.96
R103	5	425.4	0.29	479.77	0	12.78
R104	5	437.79	0.29	479.77	0	9.59
R105	15	607.25	0.16	-	0	-
R106	15	453.48	0.26	-	0	-
R107	15	447.51	0.26	-	0	-
R108	15	449.82	0.26	-	0	-
R109	22	521.33	0.16	-	0	-
R110	14	471.32	0.22	-	0	-
R111	15	434.5	0.25	474.97	0	9.31
R112	44	505.02	0.16	-	0	-
R201	28	561.66	0.19	568.4	0	1.2
R202	61	465.14	0.25	470.29	0	1.11
R203	61	465.81	0.25	470.29	0	0.96
R204	61	459.19	0.25	470.29	0	2.42
R205	119	607.25	0.16	-	0	-
R206	120	442.36	0.23	-	0	-
R207	120	463.48	0.22	-	0	-
R208	120	468.12	0.23	-	0	-
R209	57	501.94	0.21	-	0	-
R210	60	471.94	0.25	492.51	0	4.36
R211	177	509.54	0.16	-	0	-
RC101	15	1107.65	0.16	-	0	-
RC102	15	430.65	0.25	484.77	0	12.57
RC103	15	441.9	0.25	484.77	0	9.7
RC104	15	439.42	0.25	484.77	0	10.32
RC105	5	378.68	0.29	429.56	0	13.44
RC106	29	1107.65	0.17	-	0	-
RC107	22	553.2	0.17	-	0	-
RC108	28	444.37	0.17	294.21	0	-33.79
RC201	60	1107.65	0.17	-	0	-
RC202	60	399.69	0.25	407.11	0	1.86
RC203	60	468.58	0.26	407.11	0.01	-13.12
RC204	60	460.61	0.25	407.11	0.01	-11.62
RC205	30	692.21	0.26	663.07	0.01	-4.21
RC206	119	1107.65	0.16	-	0	-
RC207	57	704.26	0.21	-	0	-
RC208	177	723.71	0.16	-	0	-
Moyenne		510.56	0.22	391.82	0.00	-3.27
Résolues		56/56		30/56		

TABLE *3.1: Résultats de H1 et H2 avec* $n = 5$, $p = 3$ *et* $\epsilon = \epsilon_{max}$

trois méthodes, le coût obtenu est reporté dans la colonne **Coût**. Le temps d'exécution est donné en seconde pour CPLEX 12.4 dans la colonne **Tps(s)** et en millisecondes pour H1 et H2 dans les colonnes **Tps(ms)**. Le temps de résolution nécessaire à CPLEX 12.4 pouvant se révéler très long, il a été limité à une heure pour ces tests. La colonne **%LB** donne l'écart de CPLEX 12.4 avec sa borne inférieure à la fin de l'exécution (au bout d'une heure de calcul ou lorsque la solution optimale est trouvée). Lorsque le coût reporté est prouvé optimal, il est affiché en gras souligné. Pour les heuristiques H1 et H2, le pourcentage d'écart avec CPLEX 12.4 calculé comme $(Coût\text{-}Coût(CPLEX))/Coût(CPLEX) \times 100$ est reporté dans la colonne **%CPLEX**. En fin de tableau, la ligne *Moyenne* reporte la moyenne de chaque quantité sur l'ensemble des instances et *Moyenne** sur les seules instances pour lesquelles CPLEX 12.4 fourni la solution optimale. Le solveur parvient à trouver une solution optimale pour 33 des 56 instances proposées en moins d'une heure, avec un temps d'exécution moyen de 261.23 secondes. Pour les autres instances, l'écart au bout d'une heure à la borne inférieure générée peut aller jusqu'à 55.8% pour RC208. Les temps de calcul observés pour H1 et H2 sont en comparaison négligeables, en moyenne inférieurs à la milliseconde. L'heuristique H1 retrouve huit solutions optimales contre quatre pour H2 et se trouve en moyenne à 28.35% de CPLEX 12.4 contre 25.83% pour H2. Ces écarts passent respectivement à 17.66% et 15.72% lorsque seules les instances résolues à l'optimum par le solveur sont considérées. Ces différences d'écart importantes laissent penser que les instances que CPLEX 12.4 ne parvient pas à résoudre en moins d'une heure sont "difficiles". Il est en effet probable pour ces instances que le coût renvoyé par le solveur soit optimal mais que la difficulté réside dans la preuve de cette optimalité. Ceci peut s'expliquer par une mauvaise performance de la borne inférieure utilisée (relaxation continue) sur ces instances. Le groupe RC semble particulièrement concerné puisque sur les 16 instances qu'il compte, seules 3 sont résolues en moins d'une heure. Ces jeux de tests de petite taille semble donner l'avantage à H2 sur H1.

Afin de trouver les limites de la résolution d'instances de *PVRPTS* par CPLEX, nous avons essayé d'augmenter le nombre de clients pour $p = 3$ sur l'instance R101. Le résultat de cette expérience est représenté sur la figure 3.5. Le temps CPU semble nul à cause de l'échelle pour $n = 5$ mais il est en fait de 8.53 secondes. Pour $n = 8$ et $p = 3$ (taille équivalente à un CVRP avec 24 clients), CPLEX s'arrête au bout de 2016 secondes pour insuffisance mémoire avec un écart de 10.52% entre ses bornes supérieures et inférieures.

3.5.6 Résultats sur les grandes instances

Les heuristiques H1 et H2 sont comparées ici sur des instances plus grandes que précédemment, plus proches des tailles rencontrées en pratique. Les tests sont faits en considérant les

		CPLEX 12.4			H2			H1		
Instance	ϵ	**Coût**	**%LB**	**Tps(s)**	**Coût**	**Tps(ms)**	**%CPLEX**	**Coût**	**Tps(ms)**	**%CPLEX**
C101	11	**223.96**	0	0.55	284.48	0.24	27.02	**223.96**	0.02	0.00
C102	13	127.26	40.13	3600	169.11	0.31	32.89	212.47	0.01	66.96
C103	13	127.26	40.14	3600	168.76	0.3	32.61	212.47	0.01	66.96
C104	13	127.26	40.13	3600	176.36	0.32	38.58	212.47	0.01	66.96
C105	22	**223.96**	0	4.52	**223.96**	0.22	0.00	**223.96**	0	0.00
C106	11	**223.96**	0	1	254.22	0.27	13.51	259.09	0	15.69
C107	44	**223.96**	0	2.43	254.22	0.18	13.51	**223.96**	0	0.00
C108	45	**223.96**	0	9.06	254.22	0.21	13.51	**223.96**	0	0.00
C109	89	**223.96**	0	34.2	290.67	0.16	29.79	**223.96**	0.01	0.00
C201	40	**376.67**	0	42.28	436.5	0.16	15.88	406.59	0	7.94
C202	40	211.34	6.69	3600	322.86	0.28	52.77	319.6	0	51.23
C203	40	211.34	7.69	3600	296.14	0.28	40.12	319.6	0	51.23
C204	40	211.34	7.09	3600	287.34	0.28	35.96	319.6	0	51.23
C205	79	**346.75**	0	12.14	**346.75**	0.16	0.00	**346.75**	0	0.00
C206	88	**346.75**	0	30.75	**346.75**	0.17	0.00	397.52	0	14.64
C207	44	**346.75**	0	5.57	416.41	0.17	20.09	416.41	0	20.09
C208	159	**346.74**	0	296.76	379.06	0.16	9.32	**346.74**	0	0.00
R101	2	**540.16**	0	0.58	**540.16**	0.25	0.00	**540.16**	0.01	0.00
R102	2	**391.8**	0	12.28	433.41	0.32	10.57	479.77	0	22.40
R103	2	**391.8**	0	12.29	421.77	0.33	7.60	479.77	0	22.40
R104	2	**391.8**	0	12.23	421.77	0.32	7.60	479.77	0	22.40
R105	7	498.07	4.02	3600	525.4	0.16	5.49	498.74	0	0.13
R106	7	**380.65**	0	194.86	435.07	0.27	14.30	481.45	0	26.48
R107	7	**380.65**	0	196	410.65	0.28	7.88	481.45	0	26.48
R108	7	**380.65**	0	202.49	436.81	0.28	14.75	481.45	0	26.48
R109	11	459.22	1.3	3600	494.28	0.21	7.63	466.32	0.01	1.55
R110	7	**358.45**	0	0.67	430.36	0.26	20.06	451.06	0	25.84
R111	7	**380.65**	0	735.34	447.93	0.28	17.68	483	0	26.89
R112	22	**358.45**	0	7.1	437.07	0.21	21.93	454.14	0	26.70
R201	14	**538.87**	0	3.53	553.64	0.26	2.74	568.4	0	5.48
R202	30	**358.45**	0	27.83	456.21	0.28	27.27	479.1	0	33.66
R203	30	**358.45**	0	27.55	467.86	0.28	30.52	479.1	0	33.66
R204	30	**358.45**	0	27.58	461.24	0.27	28.68	479.1	0	33.66
R205	59	494.15	4.66	3600	501.51	0.16	1.49	516.51	0	4.52
R206	60	**358.45**	0	12.95	443.03	0.25	23.60	470.29	0	31.20
R207	60	**358.45**	0	19.41	449.83	0.25	25.49	470.29	0	31.20
R208	60	**358.45**	0	12.87	447.94	0.25	24.97	470.29	0	31.20
R209	28	472.85	0.22	3600	472.85	0.25	0.00	472.85	0	0.00
R210	30	**358.45**	0	8.98	481.52	0.27	34.33	492.51	0	37.40
R211	88	404.00	3.97	3600	463.91	0.21	14.83	463.83	0	14.81
RC101	7	**574.83**	0	3600	579.24	0.16	0.77	653.68	0	13.72
RC102	7	252.81	41.36	3600	359	0.28	42.00	423.1	0	67.36
RC103	7	252.81	41.39	3600	418.73	0.28	65.63	423.1	0.01	67.36
RC104	7	252.81	41.38	3600	381.47	0.28	50.89	423.1	0.01	67.36
RC105	2	261.90	0.21	3600	362.56	0.32	38.43	429.56	0	64.02
RC106	14	420.94	0.06	3600	622.18	0.16	47.81	524.24	0	24.54
RC107	11	277.40	8.87	3600	506.09	0.22	82.44	441.09	0	59.01
RC108	14	**252.81**	0	2372	345.43	0.24	36.64	294.21	0	16.38
RC201	30	**723.71**	0	3.81	784.62	0.16	8.42	784.62	0	8.42
RC202	30	252.81	42.07	3600	470.72	0.25	86.20	409.58	0	62.01
RC203	30	252.81	42.06	3600	468.02	0.25	85.13	409.58	0	62.01
RC204	30	252.81	42.09	3600	471.06	0.25	86.33	409.58	0	62.01
RC205	15	468.58	26.97	3600	631.56	0.26	34.78	663.07	0	41.51
RC206	59	583.65	0.51	3600	646.31	0.16	10.74	653.68	0	12.00
RC207	28	**580.69**	0	691	640.38	0.25	10.28	710.4	0	22.34
RC208	88	314.28	55.8	3600	423.62	0.22	34.79	440.23	0	40.08
Moyenne		346.98	8.91	1632.51	422.38	0.24	25.83	430.74	0.00	28.35
Moyenne*		371.00	0.00	261.23	424.64	0.24	15.72	438.09	0.00	17.66
NbOpt/NbInst		33/56			4/56			8/56		

TABLE *3.2: Comparaison de H1 et H2 avec CPLEX pour* $n = 5$, $p = 3$ *et* $\epsilon = 0.5 \times \epsilon_{max}$

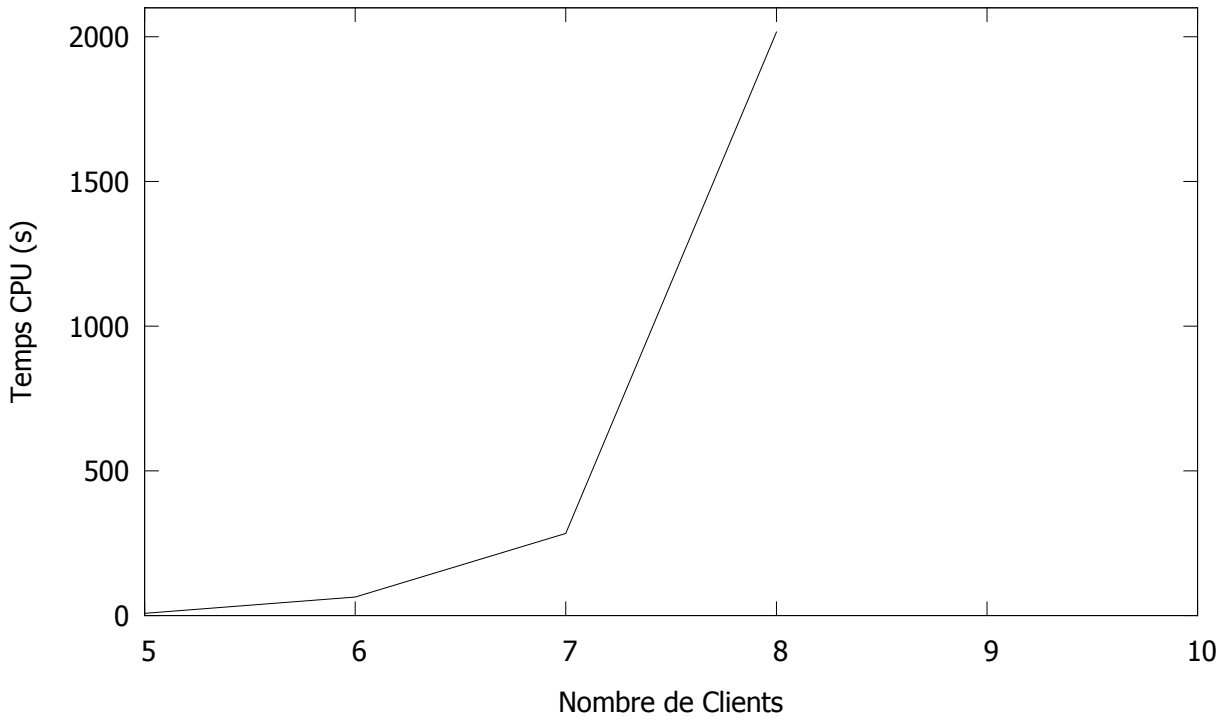


FIGURE *3.5: Temps CPU pour le* PVRPTS *avec 3 périodes et* $\epsilon = \lfloor \epsilon_{max} \rfloor$ *sur R101*

50 premiers puis la totalité des clients des instances de Solomon. Le nombre de périodes de l'horizon de planification est pris tel que $m = 3$ et l'écart temporel minimum entre deux services chez un clients est fixé à $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$. Cette valeur de ϵ est choisie de manière à ce que l'heuristique H1 parvienne à résoudre la totalité des instances proposées. La taille des instances considérées ici ne rend plus possible la résolution avec CPLEX 12.4. Les heuristiques H1 et H2 sont donc comparées entre elles. La colonne **%H2** indique le pourcentage d'écart entre H1 par rapport à H2, il est calculé comme ($Coût$(H1)-$Coût$(H2))/$Coût$(H2) $\times$100. Contrairement aux petites instances, H1 domine ici nettement H2 dont le temps d'exécution est environ 100 fois plus important puisque qu'exécuté sur 100 itérations. La "mise à l'échelle" de ces deux heuristiques profite donc largement à H1 qui avec un temps d'exécution qui reste très faible produit des résultats jusqu'à 81.38% meilleur que H2. Un bémol à cette performance est toutefois que, contrairement à H2, H1 ne peut être utilisée que pour des valeurs de ϵ inférieures à $0.5 \times \epsilon_{max}$.

		H2		H1		
Instance	ϵ	**Coût**	**Tps(ms)**	**Coût**	**Tps(ms)**	**%H2**
C101	9	3724.15	17.62	1544	0.19	-58.54
C102	10	3671.27	17.97	2505.58	0.17	-31.75
C103	10	3143.72	20.84	2673.98	0.18	-14.94
C104	10	2480.4	26.54	2181.54	0.2	-12.05
C105	18	3568.26	17.57	1569.17	0.19	-56.02
C106	7	3341.71	21.52	1504.87	0.22	-54.97
C107	44	3846.03	12.32	1739.45	0.19	-54.77
C108	37	3062.83	18.25	1741.4	0.18	-43.14
C109	89	3401.54	10.8	2176.53	0.14	-36.01
C201	40	4867.02	9.79	1589.47	0.16	-67.34
C202	40	4540.32	12.54	1923.58	0.15	-57.63
C203	40	3762.5	17.73	2504.16	0.18	-33.44
C204	40	2728.63	26.2	2264.1	0.2	-17.02
C205	79	4187.19	10.28	1781.27	0.16	-57.46
C206	86	3446.84	14.9	2166.43	0.15	-37.15
C207	44	2439.23	26.18	2196.95	0.19	-9.93
C208	159	3547.02	10.74	1891.47	0.15	-46.67
R101	2	6161.68	14.52	5494.61	0.13	-10.83
R102	2	5635.88	17.2	5152.2	0.14	-8.58
R103	2	4702.59	20.89	4362.04	0.15	-7.24
R104	2	3399.19	27.44	2808.22	0.18	-17.39
R105	7	5337.25	9.91	4016.66	0.14	-24.74
R106	7	4990.64	11.89	3940.76	0.14	-21.04
R107	7	4342.34	15.45	3288.67	0.15	-24.26
R108	7	3432.39	21.93	2855.92	0.17	-16.79
R109	9	3876.46	18.72	3428.75	0.15	-11.55
R110	7	3375.96	22.58	3047.3	0.16	-9.74
R111	6	3710.31	21.29	3056.74	0.17	-17.62
R112	19	3415.48	16.41	3081.7	0.14	-9.77
R201	9	4924.9	24.78	4607.06	0.18	-6.45
R202	9	4379.46	27.19	3910.53	0.2	-10.71
R203	9	3767.59	28.08	3607.17	0.19	-4.26
R204	19	2949.57	28.21	2507.67	0.2	-14.98
R205	59	4687.15	11.16	3785.29	0.13	-19.24
R206	59	4342.02	12.83	3639.8	0.15	-16.17
R207	59	3870.59	16.26	3310.27	0.17	-14.48
R208	60	3089.25	22.51	2356.94	0.17	-23.71
R209	28	3792.65	22.59	3228.6	0.19	-14.87
R210	24	4118.01	23.45	3432.81	0.17	-16.64
R211	79	3859.09	16.79	2847.54	0.16	-26.21
RC101	7	7206.55	10.28	4815.21	0.13	-33.18
RC102	7	6284.93	11.87	3981.39	0.14	-36.65
RC103	7	5244.29	15.62	3602.81	0.15	-31.3
RC104	7	3571.47	21.48	2991.58	0.18	-16.24
RC105	2	4780.01	24.24	3708.24	0.16	-22.42
RC106	14	5551.57	9.95	3560	0.13	-35.87
RC107	11	4042.09	17.32	3086.2	0.14	-23.65
RC108	9	2754.2	24.32	2517.11	0.16	-8.61
RC201	29	7725.22	10.87	5442.33	0.14	-29.55
RC202	29	6649.22	13.98	5267.85	0.14	-20.77
RC203	29	5329.71	16.98	4129.38	0.17	-22.52
RC204	30	3399.54	24.77	2562.73	0.19	-24.62
RC205	15	5972.56	22	5152.16	0.16	-13.74
RC206	59	6419.96	11	4243.74	0.14	-33.9
RC207	28	4550.05	22.69	3588.57	0.17	-21.13
RC208	79	4502.06	16.52	2667.88	0.13	-40.74
Moyenne		4283.97	18.17	3161.40	0.16	-25.91

TABLE *3.3: Résultats de H1 et H2 avec* $n = 50$, $p = 3$ *et* $\epsilon = 0.5 \times \epsilon_{max}$

		H2		H1		
Instance	ϵ	**Coût**	**Tps(ms)**	**Coût**	**Tps(ms)**	**%H2**
C101	9	9861.91	90.35	3067.43	0.86	-68.9
C102	10	9056.13	79.94	5167.38	0.81	-42.94
C103	10	8121.87	99.11	6092.46	0.8	-24.99
C104	10	6230.4	117.85	5511.69	0.84	-11.54
C105	18	8534.79	76.34	3284.24	0.7	-61.52
C106	7	7075.68	113.35	3788.11	0.75	-46.46
C107	44	9351.13	45.62	3878	0.7	-58.53
C108	37	7652.36	74.79	4164.24	0.73	-45.58
C109	89	8278.78	45.3	4991.74	0.59	-39.7
C201	39	11034.08	46.84	2054.32	0.74	-81.38
C202	39	10150.6	55.79	3717.11	0.73	-63.38
C203	39	8271.33	75.34	5959.94	0.72	-27.94
C204	39	6474.31	107.91	4550.32	0.87	-29.72
C205	79	9938.16	49.1	3178.79	0.76	-68.01
C206	74	6849.34	84.39	3003.38	0.73	-56.15
C207	44	5735.88	103.92	2975.46	0.91	-48.13
C208	159	7826.62	51.61	3452.43	0.62	-55.89
R101	2	10733.92	62.08	8508.87	0.5	-20.73
R102	2	9764.96	72.41	8013.23	0.57	-17.94
R103	2	7929.63	91.5	6602.4	0.67	-16.74
R104	2	5978.73	115.94	5088.01	0.75	-14.9
R105	7	8862.76	44.54	6351.3	0.53	-28.34
R106	7	7908.51	54.11	5755.34	0.56	-27.23
R107	7	7017.45	90.43	5389.9	0.63	-23.19
R108	7	5765.21	122.43	4268.08	0.74	-25.97
R109	9	6335.66	87.4	5357.85	0.62	-15.43
R110	5	5405.95	103.39	5335.8	0.72	-1.3
R111	4	5534.1	103.55	4977.06	0.75	-10.07
R112	18	5361.59	71.55	4467.34	0.61	-16.68
R201	6	7461.27	122.41	6312.3	0.82	-15.4
R202	6	6718.61	124.23	6044.02	0.9	-10.04
R203	6	5832.22	131.55	5114.68	0.93	-12.3
R204	6	4566	138.7	3559.82	1	-22.04
R205	59	7666.74	49.28	5790.81	0.62	-24.47
R206	59	6975.1	55.85	4996.75	0.64	-28.36
R207	59	6224.54	72.23	4788.88	0.66	-23.06
R208	59	4930.18	96.97	3586.64	0.79	-27.25
R209	22	5558.48	109.21	4832.53	0.77	-13.06
R210	18	5933.08	108.94	5223.58	0.81	-11.96
R211	73	5724.89	78.31	4248.83	0.73	-25.78
RC101	7	11579.29	40.09	7904.39	0.53	-31.74
RC102	7	10152.92	48.73	6943.59	0.56	-31.61
RC103	7	8698.09	64.57	6517.8	0.6	-25.07
RC104	7	7066.06	86.19	5537.72	0.73	-21.63
RC105	2	8605.21	95.66	7516	0.67	-12.66
RC106	14	9122.91	40.93	7001	0.53	-23.26
RC107	10	6963.95	76.53	5969.87	0.63	-14.27
RC108	6	5713.48	107.39	4863.09	0.76	-14.88
RC201	29	11899.74	48.12	8347.67	0.63	-29.85
RC202	29	10007.41	55.67	7601.72	0.66	-24.04
RC203	29	8234.15	72.98	6533.95	0.73	-20.65
RC204	29	6139.29	101.33	4842.5	0.9	-21.12
RC205	14	9316.6	87.98	8119.89	0.79	-12.84
RC206	59	9795.38	48.39	6883.81	0.64	-29.72
RC207	22	7083.93	107.94	5945.28	0.8	-16.07
RC208	73	7144.82	77.41	5141.46	0.71	-28.04
Moyenne		7717.07	81.87	5341.44	0.72	-28.94

TABLE *3.4: Résultats de H1 et H2 avec* $n = 100$, $p = 3$ *et* $\epsilon = 0.5 \times \epsilon_{max}$

3.6 Synthèse

Dans ce chapitre, un modèle mathématique du problème de tournées des convoyeurs de fonds est établi. Il est montré que ce modèle peut être mis sous la forme d'un programme linéaire en variables mixtes. Une condition de résolubilité sur l'écart imposé dans les visites entre les périodes est établie. Le problème de détermination des instants de départ lorsque la séquence des routes est fixée est étudié et la preuve qu'il est NP-complet est exposée. Deux heuristiques constructives simples pour ce problème sont décrites et comparées entre elles puis avec le solveur CPLEX 12.4. La supériorité de l'heuristique séquentielle est notable mais celle ci ne garantie pas de solution réalisable pour les valeurs de ϵ prises supérieures à $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$. L'heuristique parallèle permet quant à elle de générer un grand nombre de solutions différentes pour n'importe quelle valeur de ϵ pourvu que $\epsilon < \epsilon_{max}$. Les travaux décrit dans ce chapitre ont fait l'objet de présentation aux congrès internationaux IESM2011 [80] et MIC2011 [79].

Références

J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vitry. Vehicle routing problem with overlap constraints. In *Proceedings of the International Conference on Industrial Engineering and Systems Management (IESM'2011)*, pages 1311 – 1320, Metz, France. École nationale d'ingénieurs de Metz (ENIM), International Institute for Innovation, Industrial Engineering and Entrepreneurship (I^4e^2). ISBN 978-2-9600532-3-4.

J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vitry. A periodic vehicle routing problem with time windows and time spread constraints on services. In *MIC 2011 : The IX Metaheuristics International Conference*, Udine, Italy, 2011.

Chapitre 4

Métaheuristique MS-ILS efficace pour le *PVRPTS*

4.1 Introduction

Dans le chapitre précédent, nous avons pu voir que les solveurs commerciaux de programmation linéaires atteignent rapidement leurs limites et que leur utilisation n'est pas envisageable dans un contexte opérationnel. De plus, nous avons également constaté que les heuristiques constructives proposées, bien que rapides, présentent un écart non négligeable avec l'optimum sur les jeux de tests de petites tailles. Le développement d'une solution concrète, cadre de cette thèse CIFRE, implique de trouver un compromis entre rapidité d'exécution et qualité de la solution délivrée. Ce chapitre est consacré à la résolution du PVRPTS au moyen d'une métaheuristique de type MS-ILS permettant d'obtenir rapidement des résultats proches de l'optimum pour de grandes instances. Le choix de cette méthode a été guidé par le faible nombre de paramètres qu'elle comporte, ce qui la rend particulièrement intéressante dans la perspective d'un usage industriel. Elle détaillée dans un premier temps puis des tests sont effectués sur des instances de la littératures et des instances réelles. La méthode est ensuite testée sur un problème classique de la littérature afin de la comparer avec d'autres travaux.

4.2 MS-ILS

4.2.1 Principes des méthodes ILS et MS-ILS

La recherche locale itérée (ILS pour Iterated Local Search en anglais) est une métaheuristique basée sur le principe de proximité des optima locaux (*proximate optimality principle*) énoncé par Glover and Laguna [53]. Selon ce principe empirique, les optima locaux sont souvent regroupés en grappes dans l'espace de recherche. ILS initie sa recherche à partir d'une bonne solution réalisable calculée à l'aide d'une heuristique gloutonne. Cette solution est ensuite améliorée au moyen d'une procédure de recherche locale de manière à obtenir un premier optimum. Chaque itération applique ensuite une perturbation à la solution courante avec l'espoir de sortir de son bassin d'attraction. La solution une fois perturbée retourne dans la recherche locale pour atteindre un autre optimum local, proche du précédent. La solution courante est remplacée par la nouvelle seulement en cas d'amélioration. Ce cycle (perturbation + recherche locale) est répété jusqu'à ce qu'un critère d'arrêt soit atteint. La force de la perturbation est critique dans la méthode ILS. Si le niveau de perturbation est trop faible, la recherche ne parvient pas à sortir du bassin d'attraction courant, une perturbation trop forte au contraire aura tendance à générer des solutions dé-corrélées à l'instar de la méthode GRASP.

Un récent chapitre de livre de Lourenço et al. [76] est entièrement dédié à la méthode ILS, avec une longue liste de références. ILS appartient à la classe des méthodes de recherche à base de voisinage telle que la recherche adaptative à voisinage large (ALNS pour Adaptive Large Neighborhood Search en anglais) ou encore la recherche à voisinage variable (VNS pour Variable Neighborhood Search en anglais). La diversification est basée sur des mouvements de destruction et réparation dans l'ALNS, sur une procédure de perturbation (similaire à la procédure de mutation de l'algorithme génétique) dans l'ILS et sur des étapes d'agitations dans la VNS. Cette classe de méthodes a produit des méthodes faisant partie des meilleures pour les problèmes de tournées de véhicules ces cinq dernières années. Citons par exemple Pisinger et Ropke [96] qui proposent la méthode ALNS pour cinq variantes de VRP et Hemmelmary et al. [61] une VNS pour le PVRP. L'algorithme ILS a été appliqué avec succès au CVRP par Prins [100], au VRP avec prises et déposes simultanées par Subramanian et al. [114], au VRPTW par Ibaraki et al. [64] et même à un problème plus complexe avec temps de trajet et coûts dépendants du temps par Hashimoto et al [60].

Comme l'ILS peut se retrouver enfermé dans une grappe de bassins d'attractions ou dans un bassin très large, il est généralement plus fructueux de redémarrer la recherche dans des régions inexplorées au lieu de perdre du temps en itérations inutiles. Cette stratégie

appelée *multi-start ILS* (MS-ILS), dont le fonctionnement est décrit par l'algorithme 2, est celle qui a été choisie pour la résolution du PVRPTS. La boucle principale (lignes 3-24) exécute des procédures ILS qui mettent à jour la meilleure solution globale S^*. La solution de chaque ILS est générée en utilisant une heuristique constructive randomisée *RNDH* basée sur l'heuristique H2 précédemment décrite. Chaque itération de l'ILS (lignes 8-19) appelle une procédure de perturbation *Perturb* et une recherche locale *LS* pour mettre à jour la solution en cours $\overline{S}$ en cas d'amélioration. L'ILS en cours s'arrête après un nombre maximum d'itérations sans amélioration *NbStuck*. Comme plus de 90% du temps d'exécution est passé dans la recherche locale, le critère d'arrêt de la MS-ILS est un nombre d'appel *NbLS* à la recherche locale. Les ILS de la boucle intérieure sont également stoppées lorsque ce budget d'appel à la recherche locale est atteint.

4.2.2 Heuristique randomisée pour le *PVRPTS*

Cette heuristique découpe les fenêtres horaires originales de chaque client i en p sous-fenêtres telles que deux sous-fenêtres consécutives soient espacées temporellement de ϵ. Un découpage de fenêtres de temps pour un client i consiste à trouver des largeurs de sous-fenêtres $\{\delta_i^1, ...\delta_i^p\}$ $\in \mathbb{R}^+$ telles que :

$$\sum_{n=1}^{p} \delta_i^n = (b_i - a_i) - (p-1)\epsilon \tag{4.1}$$

Il y a donc une infinité de façons de découper une fenêtre horaire. Pour obtenir un découpage faisable, il faut procéder de la manière suivante : La taille δ_i^1 de la première sous-fenêtre est choisie aléatoirement dans l'intervalle $[0, (b_i - (p-1)\epsilon) - a_i]$. La partie de largeur $(p-1)\epsilon$ qui est retirée de l'intervalle $[0, b_i - a_i]$ est en quelque sorte une réserve "d'écarts", elle permet d'assurer qu'il sera possible de générer une sous-fenêtre pour la période qui suit. Si par exemple trois périodes sont considérées et que la taille choisie pour la première sous-fenêtre δ_i^1 est $(b_i - (p-1)\epsilon) - a_i$, il reste au moins deux points $a_i + \delta_i^1 + \epsilon$ et $a_i + \delta_i^1 + 2\epsilon$ pour créer les deux autres fenêtres horaires. Une fois que δ_i^1 est fixée, la taille de la fenêtre suivante δ_i^2 est ensuite choisie aléatoirement dans l'intervalle $[0, (b_i - (p-2)\epsilon) - (a_i + \delta_i^1 + \epsilon)]$. Tous les δ_i^k sont calculés dans $[0, (b_i - (p-k)\epsilon) - (a_i + \sum_{n=1}^{k-1}(\delta_i^n + \epsilon))]$. Cette procédure continue jusqu'à δ_i^{p-1}, δ_i^p étant fixé tel que $\delta_i^p = b_i - (a_i + \sum_{n=1}^{p-1}(\delta_i^n + \epsilon))$. La figure 4.1 explique de manière plus visuelle cette procédure de découpage avec trois périodes.

Chaque sous-fenêtre obtenue est ensuite affectée aléatoirement à une des p périodes considérées. Un ensemble de p problèmes de VRPTW sans temps d'attente autorisé est alors obtenu. Chacun de ces problèmes peut être résolu séparément car les contraintes d'espacement des

Algorithme 2 – MS-ILS

```
1: f* ← +∞
2: ls ← 0
3: tant que ls < NbLS faire
4:    S̄ ← RNDH()
5:    S̄ ← LS(S̄)
6:    ls ← ls + 1
7:    stuck ← 0
8:    tant que stuck < NbStuck and ls < NbLS faire
9:       S ← S̄
10:      S ← Perturb(S)
11:      S ← LS(S)
12:      ls ← ls + 1
13:      si f(S) < f(S̄) alors
14:         S̄ ← S
15:         stuck ← 0
16:      sinon
17:         stuck ← stuck + 1
18:      fin si
19:   fin tant que
20:   si f(S̄) < f* alors
21:      S* ← S̄
22:      f* ← f(S̄)
23:   fin si
24: fin tant que
25: retourner S*
```

dates d'arrivées sont garanties par la procédure de découpage. Une version adaptée de la procédure d'insertion gloutonne de Solomon telle que décrite dans 2.4.1.3, est utilisée pour résoudre chaque problème.

4.2.3 Recherche locale pour les problèmes avec fenêtres horaires

Comme il a déjà été évoqué en 2.4.2, la recherche locale est une procédure qui évolue dans l'espace de recherche en explorant un ou plusieurs voisinages de la solution courante. Les voisinages sont souvent définis par des "mouvements". Ces mouvements décrivent des chan-

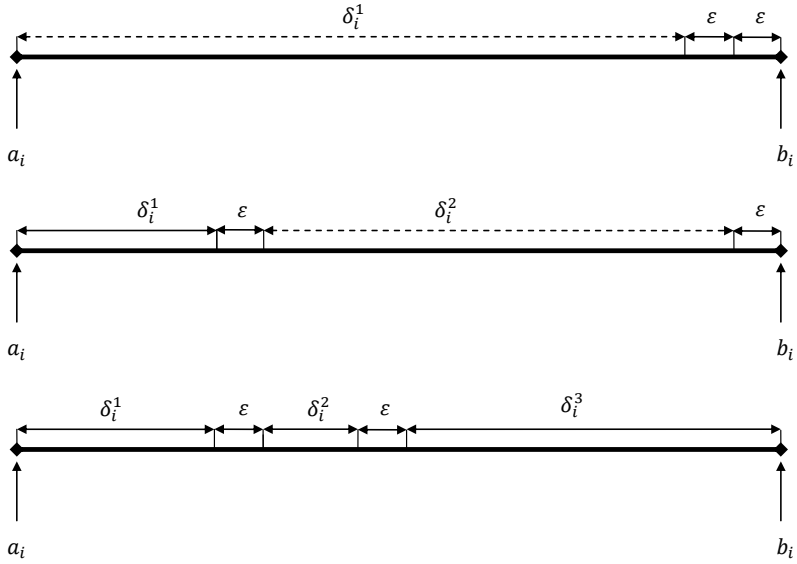


FIGURE *4.1: Procédure de découpage des fenêtres horaires pour le client i avec $p = 3$*

gements élémentaires qu'il faut appliquer à une solution donnée pour en obtenir une autre. La conception d'une procédure rapide d'évaluation de ces mouvements constitue la pierre angulaire de toute méthode de recherche locale. En effet, mettre à jour l'ensemble de la solution courante et recalculer son coût pour l'évaluer peut rapidement s'avérer gourmand en temps. Le fameux voisinage 2-opt comporte par exemple $n(n-3)/2$ mouvements comme l'a montré Lin [72]. En considérant une procédure nécessitant de reparcourir toute la solution courante pour évaluer un mouvement ($O(n)$), l'exploration de ce voisinage se ferait en temps $O(n^3)$. Heureusement, il est possible dans la plupart des cas de limiter l'effort de calcul par l'usage de variables intermédiaires qui sont mises à jour au cours de la recherche. Pour le VRPTW, Savelsbergh [112] propose de maintenir à jour des variables nommées "Forward Time Slack" et "Backward Time Slack" sur chaque client de façon à pouvoir vérifier la faisabilité et calculer le coût d'un mouvement en temps constant. Le "Forward Time Slack" représente la quantité de temps de laquelle il est possible d'avancer le départ de chez un client sans rendre la route qui suit infaisable, il s'agit en fait du décalage maximum "en avant" qu'il est possible de faire. Cette quantité peut être calculée récursivement en avant ou en arrière sur la séquence. Il est possible de calculer en temps constant le "Forward Time Slack" du premier client d'une séquence résultant de la mise bout à bout de deux séquences distinctes. Ce principe est appelé théorème de concaténation (concatenation theorem en anglais) par Savelsbergh et permet ainsi de vérifier la faisabilité d'un mouvement très rapidement. En effet, si le "Forward Time Slack" de la route résultante est négatif, celle ci n'est pas faisable.

Il est possible de définir inversement, le “Backward Time Slack" qui est la quantité de temps dont il est possible de reculer le départ de chez un client sans générer de temps d'attente dans la suite de la route. Cette nouvelle quantité permet de contrôler le temps d'attente total accumulé le long de la séquence, il est alors possible de connaitre la longueur de la route résultant d'une concaténation. Donati et al. [29] ont adapté ce principe dans le cas où les temps de trajet sont dépendants du temps. Le principe ici est de calculer la date de départ ldt_i du client i qui permet d'arriver exactement à l'instant b_{i+1} (au plus tard) chez le client $i+1$. Le “Forward Time Slack" est alors le minimum entre $b_i - t_i$ et $ldt_i - (t_i + s_i)$. Une méthode de recherche locale pour le VRPTW dans le cas où les coûts d'arrivée aux clients sont modélisés comme des fonctions linéaires par morceaux est proposée par Ibaraki et al. [63]. Si δ_k est le nombre total de parties linéaires des fonctions des n_k clients associés à une route k et δ_{max} le plus grand δ_k, le calcul de la date de départ optimale d'un véhicule est fait en $O(n_k\delta_k)$ et le calcul du coût d'une solution dans le voisinage en $O(\delta_{max})$ amorti. Plus tard, les mêmes auteurs ont proposé un algorithme de programmation dynamique [64] qui calcule le coût minimum d'une route en $O(\delta_k \log \delta_k)$ et évalue une solution du voisinage en $O(\log \delta_{max})$ amorti sous réserve que les coûts associés aux arrivées chez les clients soient convexes. Hashimoto et al.[60] ont étendu les travaux de Ibaraki et al. [63] à un problème de tournées de véhicules plus général où ils considèrent, en plus des fenêtres horaires, des temps et des coûts de trajet dépendant du temps. Plus récemment, Vidal et al.[119] ont proposé une vue très générale sur les problèmes comportant un aspect temporel. Ils proposent une méthode de ré-optimisation basée sur le fait que maintenir de bonnes informations sur les $O(n^2)$ sous-séquences d'une solution permet un gain de temps conséquent dans l'évaluation du voisinage par rapport à une approche naïve.

4.2.3.1 Recherche locale pour le PVRPTS

La procédure de recherche locale pour le *PVRPTS* est complexe parce que a) toutes les visites chez un client doivent être inspectées pour vérifier les contraintes d'écart temporel, b) les contraintes de fenêtres horaires (3.8-3.10) interdisent les temps d'attente (contrairement au VRPTW classique), et c) les contraintes de capacité (3.7) doivent également être respectées. Comme trop peu de mouvements sont possibles si toutes les contraintes sont respectées strictement, nous avons décidé de les relaxer et d'ajouter leur violation dans la fonction objectif. La MS-ILS n'a pas besoin des variables x_{ij}^{kp} et y_i^{kp} du modèle mathématique. Une solution est codée comme une table de m listes de routes (une liste par jour) et chaque route k de la période p est définie par une séquence de nœuds, un coût $C(p,k)$ et une charge $Q(p,k)$. Si l'on note t_i^p le temps d'arrivée au nœud i pendant le jour p (l'algorithme connait

le véhicule concerné), la fonction-objectif peut être écrite de manière plus simple (4.2).

$$\min z = \sum_{\substack{k\in K\\ p\in P}} C(p,k) + \sum_{\substack{i\in V\\ p\in P}} U_i^p(t_i^p) + \gamma \sum_{\substack{k\in K\\ p\in P}} \max\{Q(p,k) - W, 0\} \tag{4.2}$$

Dans le second terme, la fonction $U_i^p(t)$ (qui est détaillée par la suite) combine une pénalité si la fenêtre horaire $[a_i, b_i]$ du client i est violée par une arrivée à l'instant t pendant la période p, et une autre pénalité qui mesure l'écart entre t et les autres visites chez ce client. Le troisième terme pénalise les violations de capacité de véhicules pondérées par un coefficient γ.

4.2.3.2 Fonctions de pénalité

Dans le même esprit qu'Ibaraki et al. [63], des fonctions de pénalité linéaires par morceaux sont considérées ici. Pour une solution donnée du *PVRPTS*, une fonction de pénalité linéaire par morceaux $U_i^{kp}(t)$ est associée à chaque client i au cours de chaque période p. Cette fonction représente le coût de pénalité associé à une arrivée à l'instant t chez le client i. Elle combine l'expression de la violation des contraintes de fenêtres horaires $TW_i(t)$ et la violation des contraintes d'écart $TS_i^p(t)$. Pénaliser la violation des contraintes de fenêtres horaires revient à considérer qu'il est possible d'arriver chez le client i avant son heure d'ouverture a_i ou après son heure de fermeture b_i, puis de le servir mais en payant le prix d'une pénalité. Notons ici que même si l'arrivée se produit avant l'heure d'ouverture du client, aucun temps d'attente n'est compté : le schéma de relaxation considère que le véhicule sert le client i immédiatement. La fonction de pénalité associée à la violation des fenêtres horaires est une fonction de coût qui mesure la distance entre t et la fenêtre horaire du client i. Cette fonction est convexe et linéaire par morceaux avec trois parties comme l'indique (4.3).

$$TW_i(t) = \begin{cases} \beta(a_i - t) & \text{if } 0 < t < a_i \\ 0 & \text{if } a_i < t < b_i \\ \beta(t - b_i) & \text{if } b_i < t < +\infty \end{cases} \tag{4.3}$$

Le coefficient β donne le poids de la pénalisation et par conséquent la pente de la partie linéaire correspondante comme le montre la figure 4.2.

Il a été choisi de pénaliser la violation des contraintes d'espacement temporel par le biais d'une fonction en forme de dent comme sur la figure 4.3. Les arrivées trop proches les unes des autres chez un même client sont alors pénalisées. A cause des "dents", la fonction de



FIGURE *4.2: Fonction de pénalité de fenêtre horaire au client i*

pénalisation n'est pas convexe. La situation est même encore plus complexe car les routes sont dépendantes. En effet, un changement sur l'instant d'arrivée à un client impacte non seulement la route considérée mais également les autres routes de la solution qui visitent ce client. L'idée est de pénaliser l'arrivée à un client si celle ci intervient à moins de ϵ de n'importe quelle autre arrivée chez ce même client. Ceci est fait en définissant pour chaque client $i \in V$ à chaque période p une fonction $u_i^p(t)$ qui représente la pénalité induite par l'arrivée courante t_i^p chez ce client.

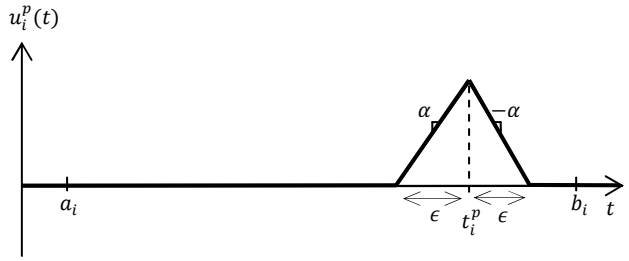


FIGURE *4.3: Pénalité d'écart générée par une arrivée à l'instant t_i^r chez le client i dans la route r*

La fonction appropriée est linéaire par morceaux en quatre parties. Le coefficient α désigne le poids de la pénalisation et donc la pente de la partie linéaire correspondante telle que le montre la figure 4.3.

$$u_i^p(t) = \begin{cases} 0 & \text{if } 0 < t < t_i^p - \epsilon \\ \alpha(t - t_i^p + \epsilon) & \text{if } t_i^p - \epsilon < t < t_i^p \\ -\alpha(t_i^p + \epsilon - t) & \text{if } t_i^p < t < t_i^p + \epsilon \\ 0 & \text{if } t_i^p + \epsilon < t < +\infty \end{cases} \tag{4.4}$$

La fonction de pénalité d'écart $TS_i^p(t)$ pour un client i au cours de la période p est obtenue en sommant les pénalités $u_i^p(t)$ générées à toutes les autres périodes $p' \neq p$ (4.5). La construction

de cette fonction est illustrée par la figure 4.4. Notons au passage que deux dents peuvent être amenées à se chevaucher et produire une fonction somme d'allure différente.

$$TS_i^p(t) = \sum_{\substack{p' \in P \\ p' \neq p}} u_i^{p'}(t) \tag{4.5}$$

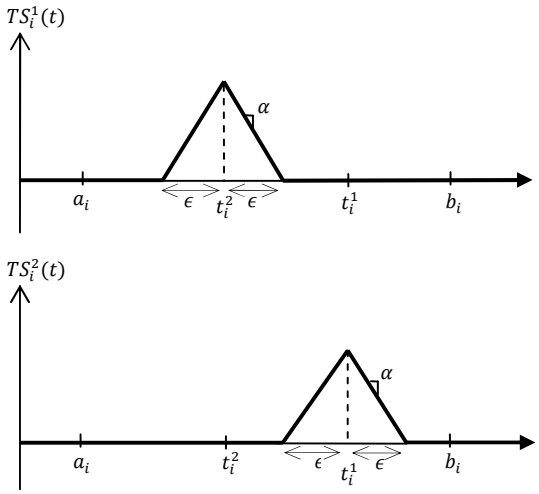


FIGURE *4.4: Fonctions de pénalité d'écart chez un client i aux périodes* 1 *et* 2.

Pour finir, la fonction de pénalité totale $U_i^p(t)$ est obtenue en sommant les fonctions de pénalité d'écart $TS_i^p(t)$ et les fonctions de pénalité de fenêtres horaires $TW_i(t)$.

$$U_i^p(t) = TW_i(t) + TS_i^p(t) \tag{4.6}$$

La fonction résultant de cette sommation constitue la fonction de pénalité du nœud i au cours de la période p. Cette fonction est également linéaire par morceaux et un exemple simple de son allure est donné par la figure 4.5. Une fois encore, les dents et les parties en "V" des pénalités de fenêtres horaires peuvent être amenées à se superposer donnant ainsi un résultat plus complexe.

4.2.3.3 Pré-calcul de données

Les mouvements de recherche locale peuvent être définis comme des découpes et des concaténations de séquences de nœuds. Soit σ une séquence de $\|\sigma\|$ clients dans une route, $\sigma_{(k)}$ son k^{th} client et $\sigma_{i,j}$ la sous-séquence de $\sigma_{(i)}$ à $\sigma_{(j)}$ inclus. Les pré-calculs effectués avant de commencer l'exploration du voisinage peuvent ensuite être utilisés pour réduire le temps

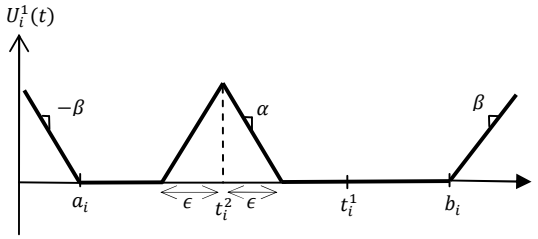


FIGURE *4.5: Fonction de pénalité totale chez le client i dans la route* 1.

passé dans l'évaluation de chaque mouvement. S'agissant du PVRPTS, nous avons besoin des données suivantes pour chacune des sous-séquences σ : sa charge $Q(\sigma)$, son coût de trajet $C(\sigma)$, sa durée $D(\sigma)$ ainsi que deux fonctions de coût linéaires par morceaux $F(\sigma)(t)$ et $B(\sigma)(t)$. La fonction de coût avant (*forward cost function* en anglais) $F(\sigma)(t)$ donne le coût d'effectuer la séquence σ si son dernier client est servi exactement à l'instant t. La fonction de coût arrière (*backward cost function* en anglais) $B(\sigma)(t)$ donne le coût de servir cette séquence si le premier client est servi à l'instant t. Toutes les données peuvent être calculées récursivement comme le montre la figure 4.6, où $\tau_{ij} = s_i + d_{ij}$ représente le temps qui sépare l'arrivée chez deux clients successifs i et j. En pratique, les informations des préfixes et suffixes de séquences $\sigma_{1,k}$ et $\sigma_{k,n}$ sont stockées avec le nœud k de la route σ dans notre implémentation.

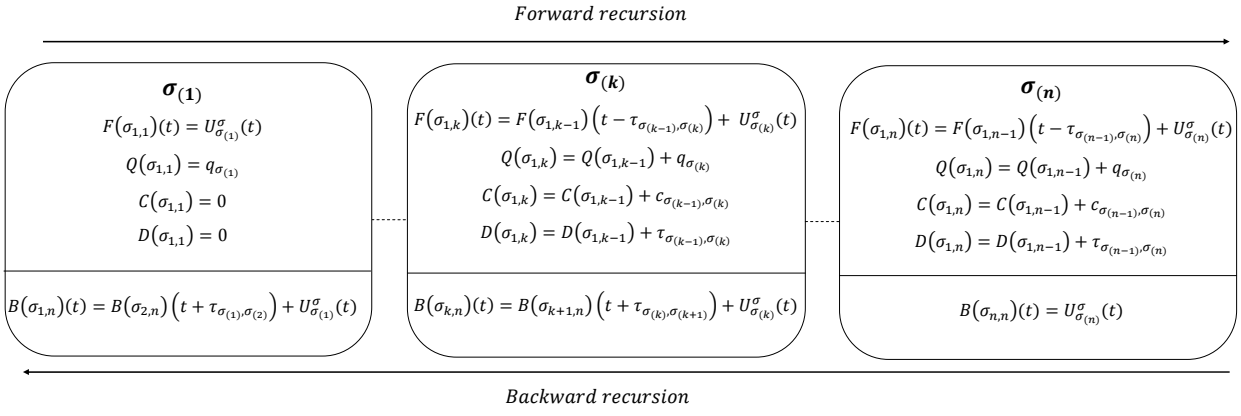


FIGURE *4.6: Parcours avant/arrière et données sur les nœuds le long de la séquence σ*

Effectuer un parcours en avant (resp. en arrière) revient à concaténer un nœud à la fin de la séquence en cours. Ce principe peut être généralisé à deux séquences de longueur quelconque comme le montrent Vidal et al. [119]. Soit $\sigma \oplus \sigma'$ la séquence résultant de la concaténation de la séquence σ' à la suite de la séquence σ, pour le PVRPTS les informations sur la nouvelle

séquence peuvent être évaluées de la manière suivante :

$$
\begin{aligned}
D(\sigma \oplus \sigma') &= D(\sigma) + D(\sigma') + \tau_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}} \\
Q(\sigma \oplus \sigma') &= Q(\sigma) + Q(\sigma') \\
C(\sigma \oplus \sigma') &= C(\sigma) + C(\sigma') + c_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}} \\
F(\sigma \oplus \sigma')(t) &= F(\sigma)(t - \tau_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}} - D(\sigma')) + F(\sigma')(t) \\
B(\sigma \oplus \sigma')(t) &= B(\sigma')(t + \tau_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}} + D(\sigma)) + B(\sigma)(t)
\end{aligned}
$$

Et le coût minimum Z^* tel que :

$$
Z^*(\sigma \oplus \sigma') = \min_t(F(\sigma)(t) + B(\sigma')(t + \tau_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}})) \tag{4.7}
$$

Il est également possible d'obtenir ces informations pour des séquences issues de découpes. Soit $\sigma \ominus^b \sigma'$ (resp. $\sigma \ominus^f \sigma'$) la séquence obtenue en retirant le suffixe (resp. le préfixe) σ' de la séquence σ. La séquence résultante peut être évaluée comme suit :

$$
\begin{aligned}
D(\sigma \ominus^b \sigma') &= D(\sigma) - D(\sigma') - \tau_{\sigma_{(\|\sigma'\|)}\sigma_{(\|\sigma'\|+1)}} \\
Q(\sigma \ominus^b \sigma') &= Q(\sigma) - Q(\sigma') \\
C(\sigma \ominus^b \sigma') &= C(\sigma) - C(\sigma') - c_{\sigma_{(\|\sigma'\|)}\sigma_{(\|\sigma'\|+1)}} \\
F(\sigma \ominus^b \sigma')(t) &= F(\sigma)(t) - F(\sigma')(t - (D(\sigma) - D(\sigma'))) \\
B(\sigma \ominus^b \sigma')(t) &= B(\sigma_{\|\sigma'\|+1,\|\sigma\|})(t)
\end{aligned}
$$

$$
\begin{aligned}
D(\sigma \ominus^f \sigma') &= D(\sigma) - D(\sigma') + \tau_{\sigma_{(\|\sigma\|-\|\sigma'\|)}\sigma'_{(1)}} \\
Q(\sigma \ominus^f \sigma') &= Q(\sigma) - Q(\sigma') \\
C(\sigma \ominus^f \sigma') &= C(\sigma) - C(\sigma') + c_{\sigma_{(\|\sigma\|-\|\sigma'\|)}\sigma'_{(1)}} \\
F(\sigma \ominus^f \sigma')(t) &= F(\sigma_{1,\|\sigma\|-\|\sigma'\|})(t) \\
B(\sigma \ominus^f \sigma')(t) &= B(\sigma)(t) - B(\sigma')(t + (D(\sigma) - D(\sigma')))
\end{aligned}
$$

4.2.3.4 Voisinages considérés

Quatre types de mouvements sont considérés :

Relocate. Ce mouvement consiste à enlever une séquence de clients de sa position courante et à la réinsérer dans une route différente ou à une autre position dans la même route. La

séquence peut être inversée avant sa réinsertion. Notre MS-ILS limite la longueur des séquences considérées à $\mu = 2$.

Swap. Le mouvement Swap appelé aussi λ-interchange échange deux séquences de 1 à λ clients appartenant à deux routes distinctes. Les deux séquences considérées peuvent avoir des longueurs différentes et être inversées avant leur réinsertion. L'implémentation présentée considère $\lambda = 3$.

2-opt. Le mouvement 2-opt retire deux arcs (i, j) et (u, v) d'une même route et reconnecte les routes concernées en utilisant les arcs (i, u) et (j, v). Ce mouvement peut aussi être défini par le retournement d'une sous-séquence de j à u inclus.

2-opt*. Dans ce mouvement, deux arcs (i, j) et (u, v) de deux routes différentes sont remplacés par (i, v) et (u, j). Contrairement au mouvement 2-opt, aucune sous-séquence n'est retournée après cette opération.

4.2.3.5 Evaluation des déplacements inter-routes

L'évaluation du mouvement de déplacement inter-route se fait de la manière suivante : soit σ^O la séquence qui représente la route d'origine de la sous-séquence de nœuds σ^m qui va être déplacée. On notera respectivement σ^{or} et σ^{ol} les sous-séquences de σ^O se trouvant respectivement à droite et à gauche de σ^m. La réinsertion de σ^m se fait dans la séquence de destination σ^D entre σ^{dl} et σ^{dr} les deux sous-séquences qui se trouveront respectivement à sa droite et à sa gauche. Une représentation des séquences et sous-séquences les composant est donnée par la figure 4.7.

Le mouvement est évalué en deux étapes, la première est l'évaluation du coût de retirer la séquence σ^m de la séquence σ^O. Ce coût est obtenu en évaluant la concaténation des deux sous-séquences restantes σ^{ol} et σ^{or} en utilisant la relation (4.8) qui dérive de (4.7). Ceci peut être fait facilement en utilisant les données pré-calculées.

$$Z^*_{remove} = Z(\sigma^{ol} \oplus \sigma^{or}) = \min_t(F(\sigma^{ol})(t) + B(\sigma^{or})(t + \tau_{\sigma^{ol}_{\|\sigma^{ol}\|}\sigma^{or}_{(1)}})) \tag{4.8}$$

Le calcul de cette expression est immédiat car $F(\sigma^{ol})(t)$ a été mémorisée au niveau du client $\sigma^{ol}_{\|\sigma^{ol}\|}$ lors du parcours de récursivité avant sur la séquence σ^O. Le parcours arrière a permis de mémoriser $B(\sigma^{or})(t)$ sur le client $\sigma^{or}_{(1)}$. Le coût de retrait de σ^m est donc obtenu en prenant le minimum de la somme de ces deux fonctions en décalant $B(\sigma^{or})(t)$ de $\tau_{\sigma^{ol}_{(\|\sigma^{ol}\|)}\sigma^{or}_{(1)}}$.

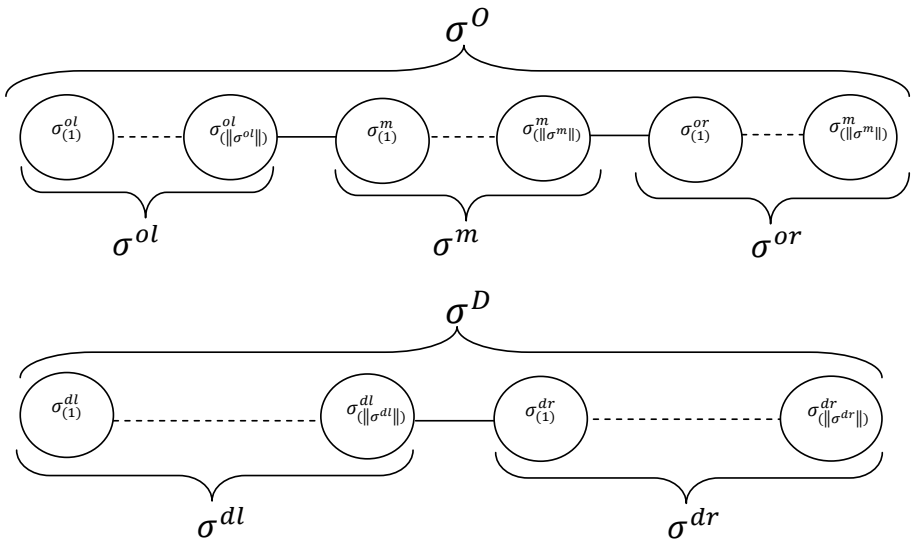


FIGURE *4.7: Séquences impliquées dans un mouvement de déplacement inter-routes*

La seconde partie de l'évaluation porte sur le calcul du coût de l'insertion de σ^m entre σ^{dl} et σ^{dr}. Ce coût est obtenu en évaluant la concaténation de ces trois sous-séquences selon (4.9).

$$\begin{aligned} Z^*_{insert} =& Z^*((\sigma^{dl} \oplus \sigma^m) \oplus \sigma^{dr}) \\ =& \min_t(F(\sigma^{dl} \oplus \sigma^m)(t) + B(\sigma^{dr})(t + \tau_{\sigma^m_{(\|\sigma^m\|)}\sigma^{dr}_{(1)}})) \end{aligned} \tag{4.9}$$

Cette tâche est un peu plus coûteuse : $B(\sigma^{dr})(t)$ est mémorisée au niveau de l'élément $\sigma^{dr}_{(1)}$, par contre il est nécessaire de calculer $F(\sigma^{dl} \oplus \sigma^m)(t)$.

$$F(\sigma^{dl} \oplus \sigma^m)(t) = F(\sigma^{dl})(t - \tau_{\sigma^{dl}_{(\|\sigma^{dl}\|)}\sigma^m_{(1)}} - D(\sigma^m)) + F(\sigma^m)(t) \tag{4.10}$$

L'information $F(\sigma^{dl})(t)$ a été pré-calculée pour le client $\sigma^{dl}_{(\|\sigma^{dl}\|)}$ mais le calcul de $F(\sigma^m)(t)$ reste à faire.

$$\begin{aligned} F(\sigma^m)(t) =& F(\sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^m\|} \ominus^b \sigma^{ol})(t) \\ =& F(\sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^m\|})(t) - F(\sigma^{ol})(t - \tau_{\sigma^{ol}_{(\|\sigma^{ol}\|)}\sigma^m_{(1)}} - D(\sigma^m)) \end{aligned} \tag{4.11}$$

$F(\sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^m\|})(t)$ et $F(\sigma^{ol})(t)$ sont respectivement mémorisées avec les éléments $\sigma^m_{(\|\sigma^{\sigma^m}\|)}$ et $\sigma^{ol}_{(\|\sigma^{\sigma^m}\|)}$, donc $F(\sigma^m)(t)$ peut être directement calculée en effectuant la soustraction avec le bon décalage.

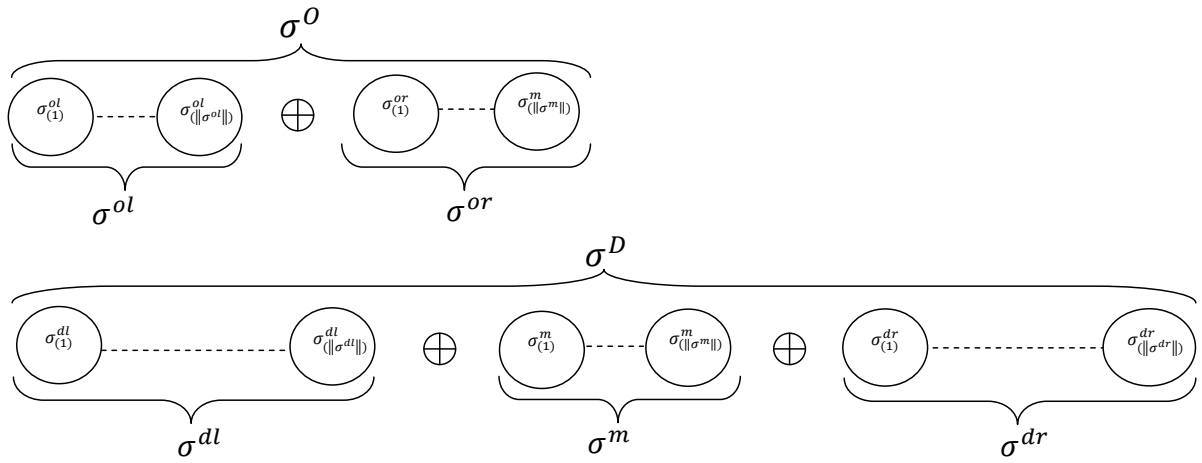


FIGURE *4.8: Concaténations évaluées pour le mouvement de déplacement inter-routes*

4.2.3.6 Evaluation d'un déplacement intra-route

Si la sous-séquence de clients déplacée est replacée à l'intérieur de la même route (i.e. $\sigma^D = \sigma^O$), l'évaluation consiste en l'évaluation de la concaténation de quatre sous-séquences. La séquence déplacée σ^m est réinsérée dans la même route, après ou avant une sous-séquence intérieure qui sera ici nommée σ^i. La concaténation à évaluer est donc soit $\sigma^{ol} \oplus \sigma^i \oplus \sigma^m \oplus \sigma^{or}$ si c'est une insertion "en avant" soit $\sigma^{ol} \oplus \sigma^m \oplus \sigma^i \oplus \sigma^{or}$ si c'est une insertion en arrière. En plus du calcul de $F(\sigma^m)(t)$ qui a déjà été défini en (4.11), il est nécessaire de calculer $F(\sigma^i)(t)$.

$$\text{Insertion avant} \begin{cases} F(\sigma^i)(t) = F(\sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^m\|+\|\sigma^i\|} \ominus^b \sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^m\|})(t) \\ F(\sigma^m)(t) = F(\sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^m\|} \ominus^b \sigma^{ol}) \end{cases}$$

$$\text{Insertion arrière} \begin{cases} F(\sigma^i)(t) = F(\sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^i\|} \ominus^b \sigma^{ol}) \\ F(\sigma^m)(t) = F(\sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^i\|+\|\sigma^m\|} \ominus^b \sigma^O_{1,\|\sigma^{ol}\|+\|\sigma^i\|})(t) \end{cases}$$

Une fois que ces données sont calculées, le coût minimum de cette réinsertion est obtenu comme suit :

$$Z^*_{insert_{forward}} = \min_t(F(\sigma^{ol} \oplus \sigma^i \oplus \sigma^m)(t) + B(\sigma^{or})(t + \tau_{\sigma^m_{(\|\sigma^m\|)}\sigma^{or}_{(1)}}))$$

$$Z^*_{insert_{backward}} = \min_t(F(\sigma^{ol} \oplus \sigma^m \oplus \sigma^i)(t) + B(\sigma^{or})(t + \tau_{\sigma^i_{(\|\sigma^i\|)}\sigma^{or}_{(1)}}))$$

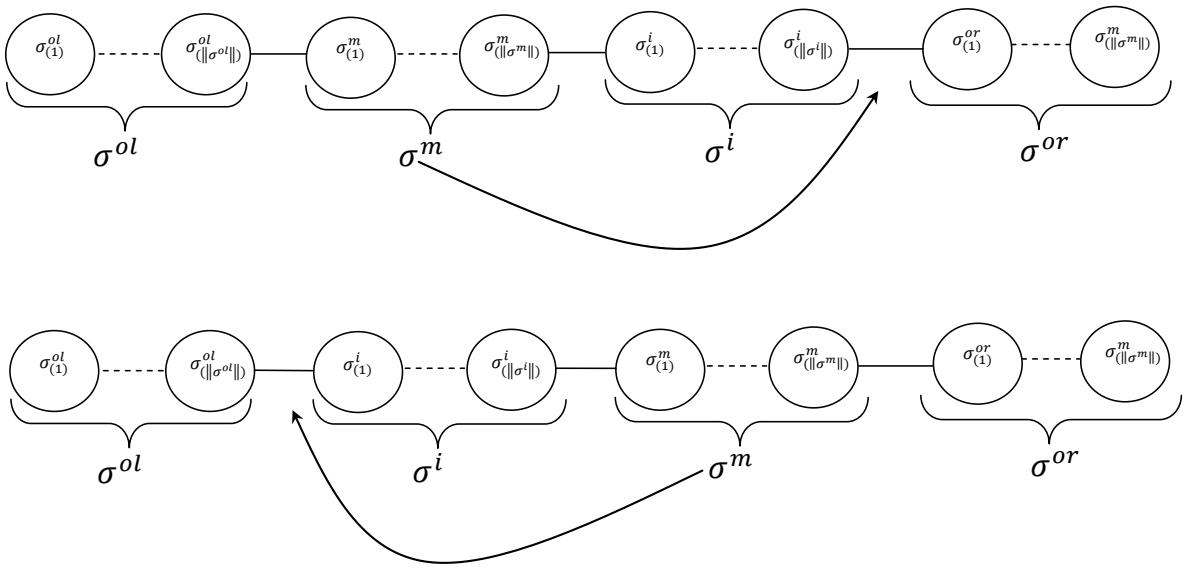


FIGURE *4.9: Déplacement dans une même route "en avant" ou "en arrière"*

Il faut remarquer ici que l'opérateur de découpe permet d'évaluer ces mouvements intra-routes dans n'importe quel ordre efficacement. Il n'est en effet pas nécessaire de respecter un ordre séquentiel d'exploration pour maintenir la complexité.

4.2.3.7 Implémentation

Au début de la procédure de recherche locale, les pré-calculs expliqués en 4.2.3.3 sont effectués et les résultats sont stockés sur les nœuds de chaque route de la solution courante. La recherche locale parcourt ensuite les quatre voisinages dans un ordre aléatoire pour essayer de trouver une meilleure solution.

Comme les mouvements améliorants introduisant de longs arcs dans la solution sont rares, une liste de voisins $LN(i)$ est construite pour chaque nœud i lorsque l'instance à résoudre est chargée. Cette liste stocke les autres nœuds, classés par ordre croissant de distance avec i. Seuls les mouvements qui introduisent de nouveaux arcs se trouvant dans les $\pi\%$ premiers sont choisis aléatoirement pour être évalués. Chaque fois qu'un mouvement améliorant est découvert, il est appliqué et les données pré-calculées sont mises à jour. Ce processus est répété jusqu'à ce que plus aucun mouvement améliorant ne soit détecté.

Au cours de la recherche locale, les coûts de retrait et d'insertion doivent être calculés pour certaines sous-séquences intérieures $\sigma_{i,j}$. Ces coûts ne sont pas calculés parce que seule une

fraction des mouvements est testée. Toutefois, une fois calculés, ils sont stockés dans une matrice $n \times n$ de manière à pouvoir être immédiatement réutilisés si aucun mouvement n'a été appliqué depuis leur calcul.

Les coefficients de pénalité α, β and γ associés aux écarts temporels, aux fenêtres de temps et aux contraintes de capacité des véhicules sont mis à jour chaque *peniter* appels à la recherche locale. Pour chaque type de contraintes, le coefficient correspondant est multiplié (respectivement divisé) par un paramètre $\omega > 1$ si le nombre de solutions infaisables pour cette contrainte sur les derniers *peniter* appels est plus grand qu'un seuil ∇_{max} (resp. plus petit qu'un seuil ∇_{min}).

4.2.4 Perturbation

Utilisée comme un mécanisme permettant de s'échapper des optima locaux, la procédure de perturbation de la *MS-ILS* vise à "sauter" dans un bassin d'attraction adjacent à celui de l'optimum local courant. La perturbation ne doit pas être un des mouvements de la recherche locale pour ne pas être réparée trop facilement. Dans notre MS-ILS, cette perturbation consiste à échanger les positions de deux clients à l'intérieur d'une même route (les mouvements d'échanges sont effectués entre deux routes distinctes dans la procédure de recherche locale).

4.3 Évaluations numériques

4.3.1 Implémentation et instances

La méthode MS-ILS a été implémentée avec le langage C# et testée sur un PC Intel Core i5 cadencé à 2,8 GHz avec 4 GB de mémoire sous Windows 7 Professional en version 32 bits. Les résultats donnés sont le coût moyen et celui de la meilleure solution sur trois exécutions. Les évaluations numériques ont été conduites sur quatre jeux d'instances. Le premier est utilisé pour confronter MS-ILS à des solutions optimales. Il contient de petits problèmes de PVRPTS obtenus en ne gardant qu'une fraction des clients des instances de Solomon [113] pour le VRPTW. Le second jeu est utilisé pour comparer MS-ILS avec H1. Il est également construit avec les instances de Solomon [113] pour le VRPTW mais tous les clients sont considérés. Le troisième jeu est composé de deux instances réelles fournies par Nexxtep Technologies. Pour finir, comme aucune métaheuristique n'a encore été publiée pour le PVRPTS, nous comparons MS-ILS avec deux algorithmes publiés sur un problème

mono-périodique : le VRP avec fenêtres de temps souples (VRPSTW pour VRP with Soft Time Windows).

4.3.2 Résultats sur les instances de Solomon

Tout comme pour pour les heuristiques H1 et H2 dans le chapitre 3, la méthode *MS-ILS* est évaluée sur les 56 instances proposées par Solomon [113]. Les tests sont tout d'abord conduits sur les instances de petite taille ne comprenant que les cinq premiers clients des fichiers d'instances originaux avec des valeurs de $\epsilon = \lfloor \epsilon_{max} \rfloor$ et $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$. Cette taille d'instance permet de confronter l'algorithme *MS-ILS* aux résultats obtenus avec le solveur CPLEX. Par la suite, les 100 clients des instances originales sont considérés et la comparaison est faite avec les heuristiques constructives.

4.3.2.1 Résultats sur les petites instances

Un "budget" de $NbLS = 1000$ appels à la recherche locale est alloué à *MS-ILS*. Les autres paramètres sont $NbStuck = 10$ (nombre d'itérations sans amélioration dans chaque ILS), $\pi = 10\%$ (proportion du voisinage exploré), $peniter = 5$ avec $\nabla_{min} = 2$, $\nabla_{max} = 4$ et $\omega = 10$. Pour finir, les coefficients de pénalité α, β et γ prennent tous une valeur initiale de 100.

Les premiers résultats pour ϵ à $\epsilon = \lfloor \epsilon_{max} \rfloor$ sur l'ensemble R sont présentés dans le tableau 4.1, avec le nom original de l'instance de VRPTW dans la colonne **Instance**, la valeur d'écart minimum considérée dans la colonne ϵ, la solution optimale atteinte par CPLEX dans la colonne **z***, le coût trouvé par notre métaheuristique dans la colonne **MS-ILS** et les pourcentages d'écart de *MS-ILS* avec l'optimum dans la colonne **%Gap**. Toutes ces instances peuvent être résolues par CPLEX en moins de 10 minutes et par MS-ILS en moins d'une seconde. Les résultats sont encourageants dans la mesure ou toutes les solutions optimales sont retrouvées par *MS-ILS*.

Les résultats pour $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$ sont reportés dans la table 4.2. Pour chacune des quatre méthodes, le coût obtenu est reporté dans la colonne **Coût**. Le temps d'exécution est donné en seconde pour CPLEX 12.4 dans la colonnes **Tps(s)** et en millisecondes pour H1, H2 et *MS-ILS* dans les colonnes **Tps(ms)**. Le temps de résolution nécessaire à CPLEX 12.4 pouvant se révéler très long, il a été limité à une heure pour ces tests. La colonne **%LB** donne l'écart de CPLEX 12.4 avec sa borne inférieure à la fin de l'exécution (au bout d'une heure de calcul ou lorsque la solution optimale est trouvée). Pour les heuristiques H1 et H2 ainsi que pour *MS-ILS*, le pourcentage d'écart avec CPLEX 12.4 calculé comme $(Coût\text{-}Coût(CPLEX))/Coût(CPLEX) \times 100$ est reporté dans la colonne **%CP**. Les écarts

Instance	ϵ	z*	MS-ILS	%Gap
R101	5	607.2	607.2	0.0
R102	5	391.8	391.8	0.0
R103	5	391.8	391.8	0.0
R104	5	391.8	391.8	0.0
R105	15	607.2	607.2	0.0
R106	15	384.3	384.3	0.0
R107	15	384.3	384.3	0.0
R108	15	384.3	384.3	0.0
R109	22	494.0	494.0	0.0
R110	14	373.2	373.2	0.0
R111	15	384.3	384.3	0.0
R112	44	403.1	403.1	0.0

TABLE *4.1: Résultats pour le jeu R1 avec* $n = 5$, $p = 3$ *et* $\epsilon = \epsilon_{max}$

aux heuristiques constructives H1 et H2 sont calculés pour *MS-ILS* comme (*Coût(MS-ILS)-Coût(Hx))/Coût(Hx))* $\times 100$ et donnés en colonnes **%H1** et **%H2**. En fin de tableau, la ligne *Moyenne total* reporte la moyenne de chaque quantité sur l'ensemble des instances et *Moyenne opt* sur les seules instances pour lesquelles CPLEX 12.4 fourni la solution optimale.

Les résultats de ces tests confirment la bonne performance de *MS-ILS* qui se trouve à 0.02% de CPLEX sur les solutions prouvées optimales et n'est battue que sur 2 des 56 instances proposées avec un écart maximum de 0.77%. Sur l'instance RC108, *MS-ILS* produit même une solution améliorant de 16,67% celle proposée par CPLEX au bout d'une heure.

4.3.2.2 Grandes instances

Pour ces tests, les $n = 100$ clients des instances originales de VRPTW sont conservés. Nous considérons toujours $p = 3$ périodes et MS-ILS prend les mêmes paramètres que pour les petites instances sauf pour l'écart minimum qui est maintenant de $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$. La taille de la flotte de $K = 25$ véhicules dans les instances de Solomon n'est pas toujours suffisante pour obtenir des solutions réalisables. Nous rappelons en effet qu'en plus des contraintes d'espacement à respecter, les temps d'attente ne sont pas autorisés pour le *PVRPTS*. La taille de la flotte de véhicules est fixée à 30 pour toute les instances de manière à ce qu'une solution réalisable puisse toujours être trouvée. MS-ILS est comparée à la meilleure heuristique constructive (H1) décrite dans le chapitre précédent.

Les résultats pour les jeux d'instances R, C et RC sont respectivement donnés dans les tableaux 4.3, 4.4, et 4.5. Pour *MS-ILS*, sont indiqués la valeur moyenne et la meilleure solution sur trois exécutions dans les colonnes **Avg** et **Best**, l'écart type exprimé en pourcentage de la moyenne dans la colonne **%Dev** et le temps d'exécution en secondes dans la colonne **T**.

Instance		CPLEX 12.4			H2			H1			MS-ILS				
Nom	ϵ	**Coût**	**%LB**	**Tps(s)**	**Coût**	**Tps(ms)**	**%CP**	**Coût**	**Tps(ms)**	**%CP**	**Coût**	**Tps(ms)**	**%CP**	**%H2**	**%H1**
C101	11	223.96	0	0.55	284.48	0.24	27.02	223.97	0.02	0	223.96	20.45	0	-21.27	0
C102	13	127.26	40.13	3600	169.11	0.31	32.89	212.47	0.01	66.96	127.26	19.76	0	-24.75	-40.1
C103	13	127.26	40.14	3600	168.76	0.3	32.61	212.47	0.01	66.96	127.26	18.01	0	-24.59	-40.1
C104	13	127.26	40.13	3600	176.36	0.32	38.58	212.47	0.01	66.96	127.26	17.92	0	-27.84	-40.1
C105	22	223.96	0	4.52	223.97	0.22	0	223.97	0	0	223.96	20.91	0	0	0
C106	11	223.96	0	1	254.22	0.27	13.51	259.09	0	15.69	223.96	21.45	0	-11.9	-13.56
C107	44	223.96	0	2.43	254.22	0.18	13.51	223.97	0	0	223.96	20.14	0	-11.9	0
C108	45	223.96	0	9.06	254.22	0.21	13.51	223.97	0	0	223.96	20.32	0	-11.9	0
C109	89	223.96	0	34.2	290.67	0.16	29.79	223.97	0.01	0	223.96	20.15	0	-22.95	0
C201	40	376.67	0	42.28	436.5	0.16	15.88	406.59	0	7.94	376.67	22.88	0	-13.71	-7.36
C202	40	211.34	6.69	3600	322.86	0.28	52.77	319.6	0	51.23	211.34	20.9	0	-34.54	-33.87
C203	40	211.34	7.69	3600	296.14	0.28	40.12	319.6	0	51.23	211.34	22.28	0	-28.64	-33.87
C204	40	211.34	7.09	3600	287.34	0.28	35.96	319.6	0	51.23	211.34	20.57	0	-26.45	-33.87
C205	79	346.75	0	12.14	346.75	0.16	0	346.75	0	0	346.75	19.4	0	0	0
C206	88	346.75	0	30.75	346.75	0.17	0	397.52	0	14.64	346.75	19.66	0	0	-12.77
C207	44	346.75	0	5.57	416.41	0.17	20.09	416.41	0	20.09	346.75	21.85	0	-16.73	-16.73
C208	159	346.74	0	296.76	379.06	0.16	9.32	346.75	0	0	346.74	17.56	0	-8.53	0
R101	2	540.16	0	0.58	540.16	0.25	0	540.16	0.01	0	540.16	18.32	0	0	0
R102	2	391.8	0	12.28	433.41	0.32	10.57	479.77	0	22.4	391.8	20.94	0	-9.6	-18.34
R103	2	391.8	0	12.29	421.77	0.33	7.6	479.77	0	22.4	391.8	22.44	0	-7.11	-18.34
R104	2	391.8	0	12.23	421.77	0.32	7.6	479.77	0	22.4	391.8	19.1	0	-7.11	-18.34
R105	7	498.07	4.02	3600	525.4	0.16	5.49	498.74	0	0.13	498.07	19.64	0	-5.2	-0.13
R106	7	380.65	0	194.86	435.07	0.27	14.3	481.45	0	26.48	380.65	18.57	0	-12.51	-20.94
R107	7	380.65	0	196	410.65	0.28	7.88	481.45	0	26.48	380.65	17.53	0	-7.31	-20.94
R108	7	380.65	0	202.49	436.81	0.28	14.75	481.45	0	26.48	380.65	21.62	0	-12.86	-20.94
R109	11	459.22	1.3	3600	494.28	0.21	7.63	466.32	0.01	1.55	460.62	20.36	0.3	-6.81	-1.22
R110	7	358.45	0	0.67	430.36	0.26	20.06	451.06	0	25.84	358.45	17.48	0	-16.71	-20.53
R111	7	380.65	0	735.34	447.93	0.28	17.68	483	0	26.89	380.65	17.8	0	-15.02	-21.19
R112	22	358.45	0	7.1	437.07	0.21	21.93	454.14	0	26.7	358.45	22.15	0	-17.99	-21.07
R201	14	538.87	0	3.53	553.64	0.26	2.74	568.4	0	5.48	538.87	22.95	0	-2.67	-5.2
R202	30	358.45	0	27.83	456.21	0.28	27.27	479.1	0	33.66	358.45	19.48	0	-21.43	-25.18
R203	30	358.45	0	27.55	467.86	0.28	30.52	479.1	0	33.66	358.45	20.88	0	-23.39	-25.18
R204	30	358.45	0	27.58	461.24	0.27	28.68	479.1	0	33.66	358.45	22.37	0	-22.29	-25.18
R205	59	494.15	4.66	3600	501.51	0.16	1.49	516.51	0	4.52	494.15	18.89	0	-1.47	-4.33
R206	60	358.45	0	12.95	443.03	0.25	23.6	470.29	0	31.2	358.45	20.16	0	-19.09	-23.78
R207	60	358.45	0	19.41	449.83	0.25	25.49	470.29	0	31.2	358.45	21.54	0	-20.31	-23.78
R208	60	358.45	0	12.87	447.94	0.25	24.97	470.29	0	31.2	358.45	19.31	0	-19.98	-23.78
R209	28	472.85	0.22	3600	472.85	0.25	0	472.85	0	0	472.85	19.71	0	0	0
R210	30	358.45	0	8.98	481.52	0.27	34.33	492.51	0	37.4	358.45	19.25	0	-25.56	-27.22
R211	88	404	3.97	3600	463.91	0.21	14.83	463.83	0	14.81	404	18.51	0	-12.91	-12.9
RC101	7	574.83	0	3600	579.24	0.16	0.77	653.68	0	13.72	579.24	18.13	0.77	0	-11.39
RC102	7	252.81	41.36	3600	359	0.28	42	423.1	0	67.36	252.81	20.8	0	-29.58	-40.25
RC103	7	252.81	41.39	3600	418.73	0.28	65.63	423.1	0.01	67.36	252.81	20.9	0	-39.62	-40.25
RC104	7	252.81	41.38	3600	381.47	0.28	50.89	423.1	0.01	67.36	252.81	21.47	0	-33.73	-40.25
RC105	2	261.9	0.21	3600	362.56	0.32	38.43	429.56	0	64.02	261.9	21.2	0	-27.76	-39.03
RC106	14	420.94	0.06	3600	622.18	0.16	47.81	524.24	0	24.54	420.94	20.9	0	-32.34	-19.7
RC107	11	277.4	8.87	3600	506.09	0.22	82.44	441.09	0	59.01	277.4	18.85	0	-45.19	-37.11
RC108	14	252.81	0	2372	345.43	0.24	36.64	294.21	0	16.38	252.81	20.62	0	-26.81	-14.07
RC201	30	723.71	0	3.81	784.62	0.16	8.42	784.62	0	8.42	723.71	19.85	0	-7.76	-7.76
RC202	30	252.81	42.07	3600	470.72	0.25	86.2	409.58	0	62.01	252.81	17.98	0	-46.29	-38.28
RC203	30	252.81	42.06	3600	468.02	0.25	85.13	409.58	0	62.01	252.81	22.24	0	-45.98	-38.28
RC204	30	252.81	42.09	3600	471.06	0.25	86.33	409.58	0	62.01	252.81	21.32	0	-46.33	-38.28
RC205	15	468.58	26.97	3600	631.56	0.26	34.78	663.07	0	41.51	468.58	18.22	0	-25.81	-29.33
RC206	59	583.65	0.51	3600	646.31	0.16	10.74	653.68	0	12	583.65	19.42	0	-9.7	-10.71
RC207	28	580.69	0	691	640.38	0.25	10.28	710.4	0	22.34	580.69	22.29	0	-9.32	-18.26
RC208	88	314.28	55.8	3600	423.62	0.22	34.79	440.23	0	40.08	261.9	22.6	-16.67	-38.18	-40.51
Moyenne total		346.98	8.91	1632.51	422.38	0.24	25.83	430.74	0	28.35	346.14	20.18	-0.28	-18.53	-19.9
Moyenne opt		371	0	261.23	424.64	0.24	15.72	438.09	0	17.66	371.12	20.23	0.02	-12.84	-13.99

TABLE *4.2: MS-ILS, H1, H2 et CPLEX pour* $n = 5$, $p = 3$ *et* $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$

Pour H1, sont reportés le coût trouvé et le temps d'exécution. La dernière colonne **%Gap** donne l'économie réalisée par *MS-ILS*, calculée comme suit : $(Best\text{-}H1)/H1 \times 100$. L'économie obtenue par MS-ILS varie de 0.73% (RC201) à 14.46% (C204).

Le tableau 4.6 indique les valeurs moyennes d'écart type, de temps d'exécution et d'écart à H1 pour chaque groupe d'instances. L'économie moyenne par rapport à H1 varie entre 4.62% pour R2 et 6.42% pour C2. L'écart type est au plus de 0.54% sur tous les jeux de problèmes, ce qui indique que notre métaheuristique est très stable.

		MS-ILS				H1		
Instance	ϵ	**Avg**	**Best**	**%Dev**	**T(s)**	**H1**	**T(s)**	**%Gap**
R101	2	8430.75	8427.48	0.04	531	8508.87	0.5	-1.22
R102	2	7721.54	7695.02	0.31	910	8013.23	0.57	-4.08
R103	2	6319.67	6299.24	0.35	1489	6602.4	0.67	-4.59
R104	2	4564.06	4505.84	1.87	1893	5088.01	0.75	-11.44
R105	7	6119.17	6101.42	0.27	763	6351.3	0.53	-3.93
R106	7	5612.56	5597.3	0.24	971	5755.34	0.56	-2.75
R107	7	5180.19	5154.01	0.46	1639	5389.9	0.63	-4.38
R108	7	4060.53	4056.45	0.12	1676	4268.08	0.74	-4.96
R109	9	5132.92	5123.24	0.16	1009	5357.85	0.62	-4.38
R110	5	4804.51	4785.11	0.44	1860	5335.8	0.72	-10.32
R111	4	4758.13	4742.08	0.47	1819	4977.06	0.75	-4.72
R112	18	4261.69	4252.33	0.24	1067	4467.34	0.61	-4.81
R201	6	6134.42	6127.61	0.11	3255	6312.3	0.82	-2.93
R202	6	5760.71	5735.91	0.37	5074	6044.02	0.9	-5.1
R203	6	4943.03	4909.65	0.6	6028	5114.68	0.93	-4.01
R204	6	3302.31	3270.08	0.93	6904	3559.82	1	-8.14
R205	59	5694.75	5673.26	0.41	3486	5790.81	0.62	-2.03
R206	59	4846.25	4837.38	0.24	4094	4996.75	0.64	-3.19
R207	59	4644.08	4625.53	0.35	5041	4788.88	0.66	-3.41
R208	59	3325.32	3314.46	0.3	7570	3586.64	0.79	-7.59
R209	22	4668.46	4664.83	0.07	5484	4832.53	0.77	-3.47
R210	18	5015.57	5000.25	0.5	4885	5223.58	0.81	-4.28
R211	73	3977.18	3963.57	0.41	6145	4248.83	0.73	-6.71

TABLE *4.3: Ensemble R (23 problèmes),* $n = 100$, $p = 3$ *et* $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$

Les instances de type 1 (avec des fenêtre horaires étroites) ont besoin de plus de temps que celles de type 2 (avec des fenêtre horaires large), ce qui peut être expliqué par le fait qu'il existe moins de mouvements ne violant pas les fenêtres horaires dans la recherche locale et donc un temps plus important passé sur des solutions pénalisées. MS-ILS peut paraître lente, mais il ne faut pas oublier que $100 \times 3 = 300$ visites aux clients doivent être définies tout en gérant les fonctions de pénalité linéaires par morceaux. Le temps passé à résoudre les instances reste finalement très acceptable pour un problème périodique qui n'a pas besoin d'être résolu tous les jours.

		MS-ILS				H1		
Instance	ϵ	**Avg**	**Best**	**%Dev**	**T(s)**	**H1**	**T(s)**	**%Gap**
C101	9	3038.2	3029.15	0.26	401	3067.43	0.86	-1.25
C102	10	4868.9	4862.24	0.15	1247	5167.38	0.81	-5.91
C103	10	5521.24	5485.11	0.63	1559	6092.46	0.8	-9.97
C104	10	5224.88	5191.77	0.56	2996	5511.69	0.84	-5.8
C105	18	3180.64	3157.08	0.64	406	3284.24	0.7	-3.87
C106	7	3544.69	3527.48	0.42	533	3788.11	0.75	-6.88
C107	44	3833.89	3833.57	0.01	468	3878	0.7	-1.15
C108	37	3977.57	3902.28	1.67	765	4164.24	0.73	-6.29
C109	89	4698.42	4682.31	0.5	959	4991.74	0.59	-6.2
C201	39	1987.46	1987.46	0	1716	2054.32	0.74	-3.25
C202	39	3563.59	3558.53	0.22	1980	3717.11	0.73	-4.27
C203	39	5395.05	5362.88	0.52	3295	5959.94	0.72	-10.02
C204	39	3907.27	3892.57	0.62	4353	4550.32	0.87	-14.46
C205	79	3139.77	3137.26	0.14	1696	3178.79	0.76	-1.31
C206	74	2894.48	2892.69	0.05	2504	3003.38	0.73	-3.69
C207	44	2662.55	2648.55	0.63	3185	2975.46	0.91	-10.99
C208	159	3378.01	3335.15	1.1	1991	3452.43	0.62	-3.4

TABLE *4.4: Ensemble C (17 problèmes),* $n = 100$, $p = 3$ *et* $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$

4.3.3 Problèmes réels

Deux instances réelles ont été fournies par Nexxtep Technologies. Elles sont extraites de données réelles de clients. Le but ici est de minimiser la durée totale des routes en minutes. La matrice des temps de trajet correspond au réseau routier réel et est par conséquent asymétrique. Les contraintes de sécurité ont été ajoutées de manière à coller au mieux avec la réalité des entreprises qui transportent des marchandises de valeurs. Ces problèmes contiennent chacun 150 clients et doivent être résolus sur 4 périodes (soit un total de 600 visites) ; L'écart minimum requis est défini à $\lfloor 0.5 \cdot \epsilon_{max} \rfloor = 20$ minutes. Les fichiers de ces instances peuvent être téléchargés aux adresses internet suivantes :

`http://81.23.34.38/nexxtep-technologies/VRPTS\_RealData/RD\_1.txt`

`http://81.23.34.38/nexxtep-technologies/VRPTS\_RealData/RD\_2.txt`

Deux versions, *MS-ILS 250* et *MS-ILS 1000*, avec respectivement 250 et 1000 appels à la recherche locale ont été testées sur trois exécutions. Les autres paramètres sont les mêmes que dans les tests précédents. Le tableau 4.7 donne les résultats de cette expérimentation. En moins de 15 minutes, MS-ILS améliore les résultats de H1 de 4%. En détaillant par exemple pour RD_1 une économie de $8906 - 8509 = 397$ minutes est réalisée ce qui revient à 6.62 heures. Un véhicule de convoi de fonds nécessite la présence de trois personnes (deux convoyeurs plus le chauffeur), l'économie totale est alors de 19.86 heures $\times$ homme sur l'horizon de planification, soit près de 5 heures $\times$ homme par jour. Les résultats obtenus

		MS-ILS				H1		
Instance	ϵ	**Avg**	**Best**	**%Dev**	**T(s)**	**H1**	**T(s)**	**%Gap**
RC101	2	8430.75	8427.48	0.04	531	8508.87	0.5	-0.96
RC102	2	7721.54	7695.02	0.31	910	8013.23	0.57	-3.97
RC103	7	5965.26	5943.33	0.46	716	6517.8	0.6	-8.81
RC104	7	5074.7	5062.03	0.22	1500	5537.72	0.73	-8.59
RC105	2	7121.3	7100.31	0.27	862	7516	0.67	-5.53
RC106	14	6589.34	6555.32	0.53	696	7001	0.53	-6.37
RC107	10	5638.54	5623.64	0.31	1163	5969.87	0.63	-5.8
RC108	6	4522.39	4488.17	0.86	1469	4863.09	0.76	-7.71
RC201	29	8314.42	8286.77	0.29	1605	8347.67	0.63	-0.73
RC202	29	7409.24	7380.44	0.34	3058	7601.72	0.66	-2.91
RC203	29	6269.33	6255.47	0.31	4576	6533.95	0.73	-4.26
RC204	29	4510.67	4483.66	0.96	5257	4842.5	0.9	-7.41
RC205	14	7812.84	7787.73	0.28	2466	8119.89	0.79	-4.09
RC206	59	6682.86	6678.17	0.07	2401	6883.81	0.64	-2.99
RC207	22	5542.67	5527.54	0.42	5337	5945.28	0.8	-7.03
RC208	73	4708.56	4671.39	0.7	6619	5141.46	0.71	-9.14

TABLE *4.5: Ensemble RC (16 problèmes),* $n = 100$, $p = 3$ *et* $\epsilon = \lfloor 0.5 \times \epsilon_{max} \rfloor$

Instance set	**Avg %Dev**	**Avg T(s)**	**Avg %Gap**
R1	0.41	1302.25	-5.1
R2	0.39	5269.64	-4.62
C1	0.54	1037.11	-5.26
C2	0.41	2590	-6.42
RC1	0.39	927.5	-6.04
RC2	0.42	3914.88	-4.82

TABLE *4.6: Moyenne des indicateurs pour chaque jeu d'instances*

sont donc considérés comme très satisfaisants. Les temps d'exécution sont plus courts que ceux observés lors des tests précédents, ce qui montre que les problèmes réels peuvent être très différents des instances académiques.

Instance	MS-ILS 250/1000				H1		250/1000
Name	**Avg**	**%Dev**	**T(s)**	**Best**	**H1**	**T(s)**	**%Gap**
RD_1	8628.25/8561.5	0.95/1.05	750.45/2610.85	8509/8446	8906	14.14	-4.46/-5.17
RD_2	5723.25/5702	0.71/0.77	567.61/2053.27	5675/5649	5897	12.23	-3.76/-4.21

TABLE *4.7: Résultats pour les instances réelles*

La majeure partie de l'amélioration atteinte en 1000 itérations est en fait déjà obtenue en 250 itérations (86.3% pour RD_1 et 89.5% pour RD_2), ce qui montre qu'augmenter le nombre d'itérations n'amène qu'à une faible amélioration. Le comportement de l'algorithme au cours de la recherche est représenté sur la figure 4.10, où le coût de la solution courante (celle sur laquelle les perturbations et recherches locales sont appliquées) est tracé pour chaque itération. Les redémarrages successifs sont très visibles. La première descente est

progressive, suivie par des redémarrages au cours desquels le nombre maximum d'itérations sans amélioration (*NbStuck*) est atteint rapidement. La dernière amélioration est obtenue à l'itération 563.

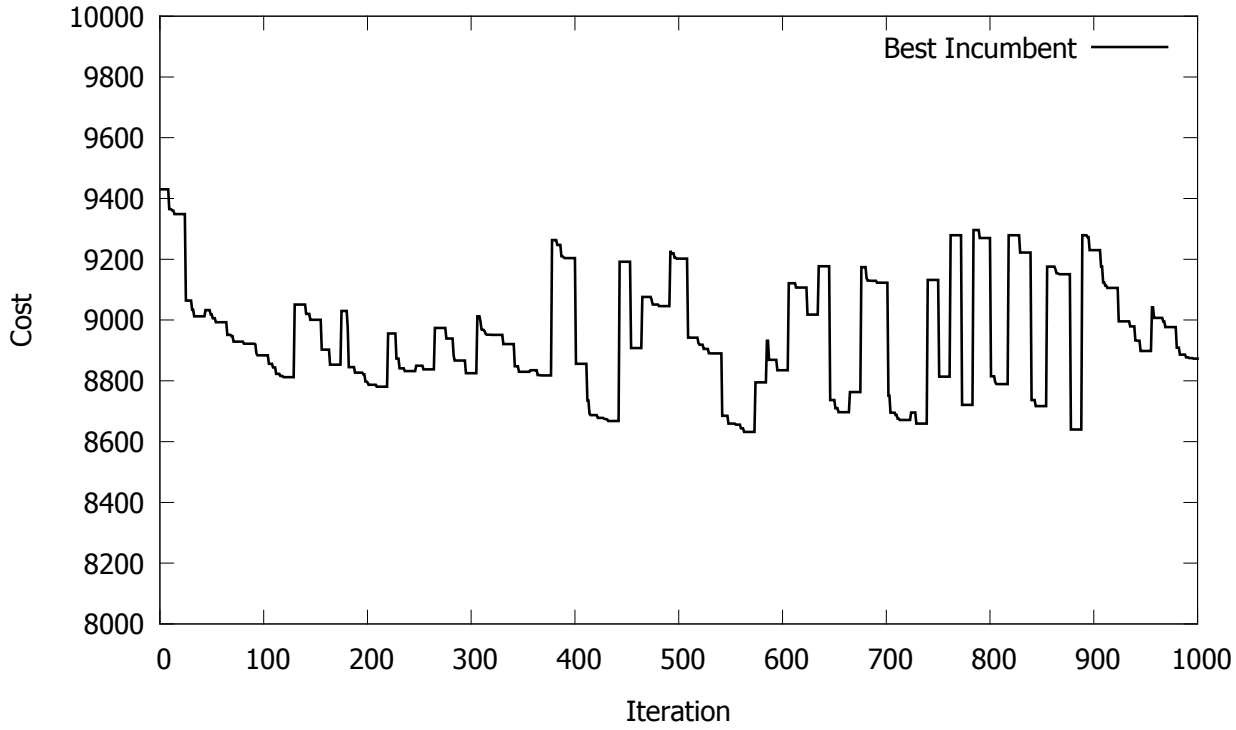


FIGURE *4.10: Comportement de* MS-ILS *sur l'instance RD_1*

4.3.4 Tests sur les instances de VRPSTW

Afin de rendre possible la comparaison avec des résultats publiés, un cas particulier mono-période du *PVRPTS* est considéré. Les contraintes d'espacement entre les dates d'arrivées n'ont alors plus de sens. Une comparaison sur le VRPTW classique aurait été difficile car dans ce cas, les temps d'attente sont autorisés et la fonction-objectif considère la minimisation de la taille de la flotte et de la longueur totale des routes hiérarchiquement. Dans notre problème, les temps d'attente sont interdits et le nombre de véhicules utilisés ne doit pas dépasser la taille de la flotte disponible.

Un problème plus proche est le VRP avec fenêtres de temps souples (VRPSTW pour VRP with soft time windows en anglais) où les livraisons peuvent se produire avant ou après la fenêtre horaire du client moyennant le paiement d'une pénalité. Fu et al. [42] ont étudié différentes formes de fonctions de pénalité et donnent les résultats de tests expérimentaux conduits avec une recherche taboue unifiée sur ce qu'ils appellent le VRPSTW de type 2,

avec des pénalités linéaires pour les arrivées prématurées et tardives. Leurs tests sont faits sur un sous-ensemble de 21 instances de celles de Solomon. Les résultats d'une heuristique précédemment proposée par Koskosidis et al. [67] sont également disponibles pour ce problème. Nous avons adapté notre MS-ILS au VRPSTW de type 2 en considérant $p = 1$, $\epsilon = 0$ et en remplaçant l'heuristique générant les solutions initiales par l'heuristique d'insertion de Solomon pour le VRPTW dont le fonctionnement a été rendu aléatoire : chaque insertion est sélectionnée aléatoirement dans une liste de candidates restreinte aux trois meilleures.

Fu et al. et Koskosidis et al. minimisent le nombre de véhicules, puis la pénalité totale induite par le non respect des fenêtres horaires et finalement la distance totale parcourue. Pour procéder à une comparaison valide, nous avons lancé notre MS-ILS sur chaque instance avec une flotte de véhicules égale au nombre maximum de véhicules spécifiés dans les fichiers d'instances de Solomon et nous avons décrémenté le nombre de véhicules tant que notre algorithme trouvait une solution valide pour le *PVRPTS* (sans temps d'attente, sans violation de capacité).

Les paramètres utilisés par MS-ILS pour cette comparaison sont $NbLS = 250$ appels à la recherche locale, $NbStuck = 10$, $\pi = 50\%$, $peniter = 5$ avec $\nabla_{min} = 2$, $\nabla_{max} = 4$ et $\omega = 2$. Pour finir, les coefficients de pénalité α, β et γ sont tous initialisés à 100. Les valeurs trouvés par Koskosidis et al., Fu et al. ainsi que les nôtres sont reportées dans le tableau 4.8.

La taille de la flotte, la distance totale et le pourcentage de client dont la fenêtre horaire a été respectée correspondent respectivement aux colonnes **Fleet**, **Dist** et **%TW**. Les chiffres en gras indiquent les meilleures valeurs pour chaque instance et pour chaque objectif. Elles sont soulignées quand MS-ILS domine les autres méthodes au sens de Pareto.

MS-ILS trouve toujours des solutions avec 100% de fenêtres horaires respectées. Pour les trois premières instances R, elle n'atteint pas le meilleur nombre de véhicules parce que seules les solutions avec 100% de fenêtres horaires respectées sont acceptées. On peut cependant noter que la meilleure distance connue est améliorée pour R103. A part ces trois cas, notre métaheuristique retrouve et même améliore dans quatre cas la meilleure taille de flotte. Toutes les meilleures solutions connues pour les instances C sont retrouvée. De plus, notre métaheuristique trouve six solutions qui dominent les meilleures connues au sens de Pareto (R104, RC102, RC103, RC104, RC106, RC108). Les excellents résultats de MS-ILS sur le VRPSTW laissent penser qu'elle est probablement efficace sur le *PVRPTS*.

Les temps d'exécution sont donnés dans le tableau 4.9 qui indique que MS-ILS est plus lente que les méthodes précédentes. Ceci est dû au fait que notre méthode est conçue pour un problème beaucoup plus général avec plusieurs périodes et des fonctions de pénalité plus complexes.

	Koskosidis et al.		Fu et al.		MS-ILS	
Instance	**Fleet/Dist**	**%TW**	**Fleet/Dist**	**%TW**	**Fleet/Dist**	**%TW**
R101	21/**1856**	**100**	**14**/1872.94	56	19/2225.14	**100**
R102	19/**1628**	**100**	**13**/1732.54	71	18/1704.74	**100**
R103	14/1428	**100**	**12**/1542.79	91	14/**1324.19**	**100**
R104	**10**/1114	**100**	**10**/1107.18	**100**	**10**/**1073**	**100**
R108	10/975	**100**	10/**968.34**	**100**	**9**/1018.12	**100**
R109	13/**1244**	98	**11**/1379.87	96	**11**/1294.84	**100**
C101	**10**/**829**	**100**	**10**/**828.94**	**100**	**10**/**828.94**	**100**
C102	**10**/**829**	**100**	**10**/**828.94**	**100**	**10**/**828.94**	**100**
C103	**10**/**829**	**100**	**10**/918.08	**100**	**10**/**828.94**	**100**
C104	**10**/**829**	**100**	**10**/899.00	**100**	**10**/**829.28**	**100**
C105	**10**/**829**	**100**	**10**/**828.94**	**100**	**10**/**828.94**	**100**
C106	**10**/**829**	**100**	**10**/**828.94**	**100**	**10**/**828.94**	**100**
C107	**10**/**829**	**100**	**10**/**828.94**	**100**	**10**/**828.94**	**100**
C108	**10**/**829**	**100**	**10**/**828.94**	**100**	**10**/**828.94**	**100**
C109	**10**/**829**	**100**	**10**/**828.94**	**100**	**10**/**828.94**	**100**
RC101	16/1815	95	**13**/1851.22	74	15/**1673.14**	**100**
RC102	14/1605	94	**13**/1772.42	99	**13**/**1552.89**	**100**
RC103	13/1390	**100**	**11**/1416.81	**100**	**11**/**1354.08**	**100**
RC104	**10**/1353	**100**	**10**/1262.55	**100**	**10**/**1177.03**	**100**
RC106	13/1541	92	**12**/1531.57	99	**12**/**1412.98**	**100**
RC108	11/1315	99	11/1224.72	**100**	**10**/**1221.35**	**100**

TABLE *4.8: Comparaison sur 21 instances de VRPSTW de type 2*

Algorithme	Ensemble de problèmes	Avg CPU (s)	Ordinateur
Koskosidis et al.	R1	560.9	
	C1	3.0	IBM3081/VM370
	RC1	689.4	
Fu et al.	R1	993.1	600MHz Pentium-II
	C1	315.1	PC avec 184 MB RAM
	RC1	810.5	
Michallet et al.	R1	1354.2	2.8GHz Intel Core i5
	C1	1283.8	PC avec 4 GB RAM
	RC1	1297.5	

TABLE *4.9: Temps CPU pour les instances de VRPSTW*

4.4 Conclusion

Dans ce chapitre, nous avons décrit une MS-ILS pour résoudre le *PVRPTS*. Cette méthode est basée sur une procédure de recherche locale élaborée, basée sur des fonctions de pénalités linéaires par morceaux. Comme aucune métaheuristique n'a encore été publiée pour notre problème, une comparaison avec une heuristique constructive est faite sur des instances dérivées de la littérature et sur des instances réelles provenant de chez Nexxtep Technologies. Les bons résultats obtenus sont confortés par une comparaison avec un solveur commercial sur de petites instances et avec deux algorithmes publiés sur un cas particulier mono-période, le VRPSTW. Notre MS-ILS permet d'économiser jusqu'à 14% par rapport à l'heuristique constructive, retrouve toutes les solutions optimales sur les petites instances et est compétitive avec les algorithmes existants pour le VRPSTW, produisant six meilleures solutions.

Le travail présenté dans ce chapitre a fait l'objet d'un article publié dans la revue *Computers and Operations Research* [81].

Par ailleurs, une méthode de type MS-ILS a également été utilisée à l'occasion d'une collaboration avec Thibaut Vidal à l'UTT, Renaud Masson et Hugues Dubedout à l'Ecole des Mines de Nantes, Puca Huachi Vaz Penna et Vinicius Petrucci à l'Université Fédérale Fulminense (Niteroi, Brésil) et Anand Subramanian à l'Université Fédérale de Prabaia (Joao Pessoa, Brésil). Les travaux effectués ont été publiés dans la revue *Expert Systems with Applications* [78].

Références

J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vitry. Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Computers & Operations Research*, 41 :196–207, 2014.

R. Masson, T. Vidal, J. Michallet, P. H. V. Penna, V. Petrucci, A. Subramanian, and H. Dubedout. An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. *Expert Systems with Applications*, 40(13) :5266–5275, Mar. 2013.

Chapitre 5

Transposition au problème de tournées régulières

5.1 Introduction

Dans le chapitre précédent, le problème de tournées de convoyeur de fonds est traité au moyen d'une MS-ILS. L'efficacité de la méthode réside dans son contrôle efficace de l'aspect temporel complexe du problème au moyen de fonctions de coûts linéaires par morceaux. La complexité provient de la dépendance que créent les contraintes d'irrégularité entre les dates de services chez les clients. L'interdépendance des arrivées chez les clients se retrouve dans le problème inverse, celui qui vise à établir un plan de transport régulier. Ce problème est traité dans la littérature où il est appelé *Consistent VRP*. Ce chapitre présente une adaptation de notre MS-ILS permettant de traiter simplement à la fois le *Consistent VRP* classique et sa variante dans laquelle les dates de départ des routes sont également des variables de décision. Le problème est tout d'abord décrit dans la section 5.2. Les tests sont effectués sur les instances classiques du problème et sur une extension récemment proposée par Kovacs et al.[68].

5.2 Présentation du problème

Les problèmes de tournées de véhicules portent dans la grande majorité de leurs cas sur la minimisation des coûts d'exploitation pour le transporteur. Les modélisations proposées sont rarement orientées "client", il s'agit pourtant d'un critère primordial. En effet, maintenir un haut niveau de satisfaction permet de gagner la fidélité des clients et contribue ainsi à la

pérennité de l'activité. La dimension de "qualité de service" dans le problème de tournées de véhicules a été introduite par Groër et al. [58] à l'occasion d'une collaboration avec le transporteur en messagerie UPS. Les auteurs proposent un problème qu'ils nomment le *Consistent VRP*. Il s'agit de planifier le service d'un ensemble de clients ayant des demandes intermittentes sur plusieurs jours. Certains clients ne requièrent qu'une livraison sur l'horizon de planification, d'autres en demandent plusieurs. Malgré l'irrégularité de la demande, chaque client doit être servi toujours par le même chauffeur et à heure régulière. La régularité en terme de livreur facilite la création de liens avec le client tandis que la régularité horaire tend à perturber le moins possible son activité. Les auteurs proposent alors une approche de résolution basée sur la construction de "squelettes" composés des clients réguliers obtenus par un algorithme de type "record-to-record travel" proposé auparavant par Li et al. [71] pour le VRP standard. Les squelettes obtenus sont ensuite utilisés pour générer des solutions réalisables de *Consistent VRP* en y insérant les clients non réguliers. La méthode présentée est testée sur de petites instances comportant de 10 à 12 clients où elle est comparée aux solutions obtenues par le solveur commercial CPLEX 11.0. Les auteurs créent également de plus grandes instances contenant 700 clients sur 5 jours et proposent une adaptation des instances classiques de VRP de Christofides et Eilon [16] à ce problème.

Feillet et al [36] soulignent l'importance de la régularité horaire dans le transport périodique de personnes handicapées, plus sensibles aux changements. Ils abordent le problème sous un angle bi-objectif en minimisant le temps total de parcours et le nombre de classes temporelles différentes dans lesquelles peut être servi un client. Ils proposent une heuristique détruisant à chaque itération une journée de livraison pour la reconstruire en tenant compte des autres. Le problème de reconstruction s'apparente à la résolution d'un VRP avec fenêtres horaires multiples et sans temps d'attente. Les auteurs ne retiennent toutefois pas la régularité de conducteurs mais introduisent une procédure visant à réduire le nombre de conducteurs par clients. La méthode est testée sur les petites instances proposées par Groër et al. [58] et celles adaptées de Christofides et Eilon [16].

Plus récemment, Tarantilis et al. [115] proposent une recherche taboue basée sur la notion de squelette introduite par Groër et al. [58]. L'heuristique fonctionne en deux étapes. Une première phase est dédiée à la création de squelettes de très bonne qualité. Ces squelettes sont complétés en solutions valides pour le *Consistent VRP* puis améliorés dans une seconde phase. Cette métaheuristique qu'ils nomment recherche taboue à base de patron (TTS pour Template-based Tabu Search en anglais) est évaluée sur les instances dérivées de celles de Christofides et Eilon [16] et comparée avec les résultats obtenus par Groër et al. [58].

Pour finir, Kovacs et al. [68] proposent une recherche à voisinages larges (ALNS) également

basée sur la notion de squelette pour ce problème. La méthode se concentre sur la génération du meilleur squelette possible. A chaque itération, le squelette en cours est détruit puis reconstruit et il est décidé s'il est accepté ou non comme nouvelle solution. La probabilité d'accepter une solution détériorant la précédente devient de plus en plus faible au cours des itérations, à la manière d'un recuit simulé. Les tests sont conduits sur les instances modifiées de Christofides et Eilon [16] et comparée avec les résultats de Groër et al. [58] et de Tarantilis et al. [115]. Les auteurs proposent ensuite de nombreuses variations de ce jeu de tests en considérant différentes proportions de clients réguliers et en resserrant plus ou moins la contraintes d'écart maximum entre deux visites chez un client. Ils proposent également une version relaxée du *Consistent VRP* qui consiste à autoriser le décalage des instants de départ des routes de manière à pouvoir améliorer encore la régularité de service horaire chez les clients. Ils testent ensuite cette version relaxée sur les mêmes instances que la version classique et soulignent les effets de la relaxation. Les auteurs proposent pour finir un jeu d'instances de très grande taille pour la version classique du *Consistent VRP* qu'ils ont adapté des instances pour le VRPTW de Gehring et Homberger [45]. Sur ces instances contenant 1000 clients et 25 jours de planification, les auteurs effectuent leurs tests sans imposer de limite pour la régularité horaire.

5.3 Modélisation

Le *Consistent VRP* peut être défini sur un graphe orienté $G = (V, A)$ et un horizon de planification P de m périodes ou "jours". L'ensemble des sommets $V = \{0, 1, ..., n\}$ comprend n clients numérotés à partir de 1 et le dépôt $V = \{0\}$ où est basé un ensemble K de véhicules identiques de capacité W. A est l'ensemble des arcs (i, j) avec $i, j \in V$. Le temps nécessaire au parcours d'un arc (i, j) est désigné par d_{ij}. Un client i requiert une livraison pendant la période p si le paramètre associé w_i^p vaut 1. Une route valide pour un véhicule k à la période p délivre une quantité q_i^p de marchandises avec un temps opératoire s_i^p à chaque client i qu'elle visite et retourne au dépôt au plus tard à l'heure T. Les exigences de régularité imposent que chaque client $i \in V$ doit toujours être livré par le même véhicule k sur l'ensemble des périodes p de l'horizon de planification. De plus, les heures de service de chaque période de livraison d'un client ne doivent pas être étalées sur plus de L unités de temps. Soient les variables binaires y_i^{kp} égales à 1 si le client i est servi par le véhicule k pendant la période p, et les variables binaires x_{ij}^{kp} valant 1 si l'arc (i, j) est traversé par le véhicule k pendant la période p. Soient les variables réelles non-négatives t_i^{kp} qui représentent l'instant d'arrivée du véhicule k chez le client i pendant la période p. Le *Consistent VRP* peut alors être modélisé par le programme linéaire mixte suivant :

$$\min z = \sum_{p\in P}\sum_{k\in K}\sum_{(i,j)\in A} d_{ij}x_{ij}^{kp} \tag{5.1}$$

$$y_0^{kp} = 1 \qquad \forall k \in K, \forall p \in P, \tag{5.2}$$

$$t_0^{kp} = 0 \qquad \forall k \in K, \forall p \in P, \tag{5.3}$$

$$\sum_{k\in K} y_i^{kp} = w_i^p \qquad \forall i \in V\setminus\{0\}, \forall p \in P, \tag{5.4}$$

$$\sum_{i\in V} q_{ip}y_i^{kp} \leq W \qquad \forall k \in K, \forall p \in P, \tag{5.5}$$

$$\sum_{i\in V} x_{ij}^{kp} = y_j^{kp} \qquad \forall j \in V, \forall k \in K, \forall p \in P, \tag{5.6}$$

$$\sum_{j\in V} x_{ij}^{kp} = y_i^{kp} \qquad \forall i \in V, \forall k \in K, \forall p \in P, \tag{5.7}$$

$$w_i^p + w_i^{p'} - 2 \leq y_i^{kp} - y_i^{kp'} \qquad \forall i \in V, \forall k \in K, \forall p, p' \in P, p \neq p', \tag{5.8}$$

$$t_i^{kp} + x_{ij}^{kp}(s_i^p + d_{ij}) - T(1 - x_{ij}^{kp}) \leq t_j^{kp} \qquad \forall i \in V, j \in V\setminus\{0\}, \forall k \in K, \forall p \in P, \tag{5.9}$$

$$t_i^{kp} + x_{ij}^{kp}(s_i^p + d_{ij}) + T(1 - x_{ij}^{kp}) \geq t_j^{kp} \qquad \forall i \in V, j \in V\setminus\{0\}, \forall k \in K, \forall p \in P, \tag{5.10}$$

$$t_i^{kp} + w_i^p(s_i^p + d_{i0}) \leq Tw_i^p \qquad \forall i \in V\setminus\{0\}, k \in K, \forall p \in P, \tag{5.11}$$

$$L - T(w_i^p + w_i^{p'} - 2) \geq t_i^{kp} - t_i^{kp'} \qquad \forall i \in V\setminus\{0\}, \forall k \in K, \forall p, p' \in P, p \neq p', \tag{5.12}$$

$$y_i^{kp}, x_{ij}^{kp} \in \{0,1\} \qquad \forall i, j \in V, i \neq j, \forall k \in K, \forall p \in P, \tag{5.13}$$

$$t_i^{kp} \in \mathbb{R}^+ \qquad \forall i \in V, \forall k \in K, \forall p \in P. \tag{5.14}$$

La fonction-objectif (5.1) considère la minimisation du temps total de parcours. Les contraintes (5.2) et (5.3) assurent l'utilisation du dépôt au cours de chaque période et fixent le départ de celui-ci à l'instant 0 pour chaque route. La desserte d'un client chaque jour où il requiert une livraison est imposée par les contraintes (5.4) et le respect la capacité des véhicules par les contraintes (5.4). Les jeux de contraintes (5.6) et (5.7) sont les classiques contraintes de flot du VRP, qui imposent que chaque client ait au moins un successeur et un prédécesseur. La régularité en terme de livreur est obtenue grâce à (5.8). Les contraintes (5.9) et (5.10) interdisent les temps d'attente chez les clients et (5.9) permettent également d'éliminer les sous-tours. La limitation de la durée des tournées à T est fixée par (5.11) et le respect de l'écart temporel maximal L entre deux visites chez un client est imposé par (5.12). Pour finir, les contraintes (5.13) et (5.14) fixent les domaines de définition des variables.

5.4 Approche proposée

Nous proposons d'adapter le principe des fonctions de coût linéaire par morceaux utilisées pour traiter le *PVRPTS* au *Consistent VRP*. La procédure de recherche locale obtenue est intégrée au sein d'une MS-ILS dont l'algorithme 3 donne une vue générale. La méthode est similaire à celle décrite dans le chapitre 4 mais quelques raffinements ont toutefois été apportés. Le premier concerne le nombre de points de départ de la recherche locale à chaque itération. Le concept de recherche locale évolutionnaire introduit par Wolf et Merz [122] montre que la génération de plusieurs enfants à partir de la solution courante permet une meilleur intensification dans le bassin d'attraction considéré. Cette idée a été adaptée avec succès par Prins [100] pour le VRP, elle est reprise dans l'algorithme que nous proposons. Une deuxième modification concerne l'espace de recherche. La MS-ILS décrite dans le chapitre 4 permet de considérer les solutions irréalisables mais celles-ci sont gérées avec les solutions réalisables. L'expression de ces solutions pénalisées est donc limitée car elles sont systématiquement écartées dès qu'une solution réalisable est disponible. Récement, Vidal et al. [121] ont montré l'intérêt que revêt la conservation de solutions irréalisables par une sélection spécifique. Dans la méthode proposée nous conservons donc la meilleure solution réalisable en cours mais aussi la meilleure irréalisable. Nous incluons également une procédure de réparation des solutions irréalisables trouvées, appelée suivant une probabilité déterminée. Cette procédure de réparation permet de tirer tout le bénéfice de la considération de solutions irréalisables. Une dernière modification concerne le critère d'acceptation de la nouvelle solution courante, celle sur laquelle va être effectuée la prochaine série de perturbations+recherches locales. Comme relevé par Lourenço et al [76], n'accepter pour nouvelles solutions courantes uniquement les solutions améliorantes favorise grandement l'intensification dans le bassin d'attraction en cours d'exploration. Le fait de générer plusieurs individus à chaque itération, comme nous le faisons ici, produit déjà une bonne intensification. Afin d'équilibrer le rapport intensification/diversification, nous considérons comme nouvelles solutions courantes pour l'itération n, les meilleures parmi celles générées à l'itération $n-1$ que celles-ci soit améliorantes ou non. Toutefois, si le nombre d'itérations sans améliorations devient trop important, la meilleure solution connue est utilisée comme nouvelle solution courante afin de recentrer la recherche.

Les approches existant dans la littérature pour le *Consistent VRP* sont toutes articulées autour de la construction de "squelettes" performants. La MS-ILS proposée se détache un peu de la notion de squelette et propose de travailler la solution dans son ensemble.

Algorithme 3 – MS-ILS-ConVRP

$S^* \leftarrow null$
$rl \leftarrow 0$
tant que $rl < NbRl$ **faire**
  $Actuels \leftarrow \{ConstAlea(), ConstAlea()\}$
  $manque \leftarrow 0$
  tant que $manque < NbManque$ et $rl < NbRl$ et $Temps() < TpsMax$ **faire**
    $S_{fai} \leftarrow null$
    $S_{inf} \leftarrow null$
    pour chaque individu s dans $Actuels$ **faire**
      si $manque > 0$ et $manque$ mod $seuil$ ==0 **alors** $s \leftarrow \bar{S}$
      pour le nombre d'enfants Ne **faire**
        $S \leftarrow s$
        $S \leftarrow Perturb(S)$
        $S \leftarrow Rl(S)$
        $rl \leftarrow rl + 1$
        si $Infaisable(S)$ et $Aleatoire() < \Pi$ **alors** $S \leftarrow Reparer(S)$
        si $Infaisable(S)$ et $(Null(S_{inf})$ ou $C(S) < C(S_{inf}))$ **alors** $S_{inf} \leftarrow S$
        si $Faisable(S)$ et $(Null(S_{fai})$ ou $C(S) < C(S_{fai}))$ **alors** $S_{fai} \leftarrow S$
      fin pour
    fin pour
    si $f(S_{fai}) < f(\bar{S})$ ou $f(S_{inf}) < f(\bar{S})$ **alors** $\bar{S} \leftarrow S_{fai}$ ou $\bar{S} \leftarrow S_{inf}$; $manque \leftarrow 0$
    sinon $manque \leftarrow manque + 1$
    $Actuels \leftarrow \{S_{fai}, S_{inf}\}$
    $MajPenalites()$
  fin tant que
  si $f(\bar{S}) < f(S^*)$ **alors** $S^* \leftarrow \bar{S}$
fin tant que
retourner S^*

5.4.1 Espace de recherche considéré

L'espace de recherche se compose de solutions réalisables mais aussi de solutions irréalisables. Nous avons choisi de relaxer certaines contraintes afin de faciliter la progression de notre MS-ILS dans l'espace de recherche. Ces contraintes sont celles de capacité des véhicules, de durée d'une route ainsi que les contraintes de régularité horaire et de chauffeur. Il en résulte que lorsque deux solutions sont comparées entre elles, trois cas de figure peuvent se produire :

a) soit les deux solutions sont réalisables auquel cas elles sont comparées uniquement sur le temps de trajet total (TT, pour Total Travel time en anglais). b) soit une solution est réalisable et l'autre ne l'est pas et dans ce cas la solution réalisable est préférée. c) soit les deux solutions sont irréalisables et dans ce cas la fonction de coût utilisée doit prendre en compte le degré de violation des contraintes pour trancher. Une solution S est codée comme une table de m liste de routes (une liste par jour) et chaque route k de la période p est définie par une séquence de nœuds, une durée de trajet $D(p,k)$ et une charge $Q(p,k)$. Le nombre de visites que doit recevoir un client dans l'horizon de planification est noté $N_i = \sum_{p \in P} w_i^p$. Si de plus on note t_i^p le temps d'arrivée au client i pendant le jour p (l'algorithme connait le véhicule concerné), la fonction d'évaluation $f(S)$ est donnée par (5.15).

$$f(S) = \sum_{\substack{k \in K \\ p \in P}} \left(D(p,k) + \gamma \max\{Q(p,k) - W, 0\} + \phi \max\{D(p,k) - T, 0\} \right) \tag{5.15}$$
$$+ \xi \sum_{\substack{k \in K \\ i \in V}} \left(\min\left\{ N_i \sum_{p \in P} y_i^{kp}, N_i \right\} - \sum_{p \in P} y_i^{kp} \right) + \psi \sum_{\substack{i \in V \\ p \in P}} w_i^p U_i^p(t_i^p)$$

La première somme de cette fonction comprend les coûts associés aux routes. On trouve dans l'ordre : le temps total $D(p,k)$ de chaque route, la quantité de violation de capacité du véhicule pénalisée par le coefficient γ et la quantité de violation de la durée maximale d'une route pénalisée par le coefficient ϕ. Les deux sommes suivantes indiquent respectivement les quantités de violation des contraintes de livreur et de régularité horaires pénalisées par les coefficients ξ et ψ. La violation de la contrainte de régularité de chauffeur est obtenue en mesurant pour chaque client i et sur chaque véhicule k le servant, l'écart entre le nombre de services que requiert le client i et le nombre de services que le véhicule k effectue chez lui. Cette mesure a pour effet de pénaliser fortement la "dispersion" sur plusieurs véhicules du service d'un client. Afin que l'écart ne soit mesuré que si le véhicule k dessert effectivement le client i, l'expression $\min\left\{N_i \sum_{p \in P} y_i^{kp}, N_i\right\}$ est utilisée dans le premier terme de la soustraction. Cette expression s'annule si $\sum_{p \in P} y_i^{kp}$ vaut 0 (si le véhicule k ne visite jamais le client i) et vaut N_i sinon. Pour finir, les fonctions $U_i^p(t)$ associent à chaque client i un coût de service à l'instant t pendant la période p. La quantité de violation des contraintes de régularité horaire est donc déterminée en mesurant le coût de service de chaque client i au cours de chaque période p. Cette mesure n'est nécessaire que si le client i requiert effectivement un passage lors de la période p, ce que figure ici la multiplication par w_i^p. Les fonctions $U_i^p(t)$ sont, comme pour le $PVRPTS$, des fonctions linéaires par morceaux et font l'objet d'une description plus précise dans la suite de ce chapitre.

5.4.2 Fonctions de pénalité horaires

Les fonctions $U_i^p(t)$ utilisée pour mesurer le coût de service d'un client i à la période p sont des fonctions linéaires par morceaux. Pour le *Consistent VRP*, ces fonctions sont différentes selon qu'elles sont associées aux dépôts de départ de chaque route, aux clients ne nécessitant qu'un seul service ou aux clients réguliers. Celles correspondant aux dépôts de départ de chaque route modélisent le coût de départ de la route à l'instant t (le dépôt a ici un temps opératoire nul). Les contraintes (5.3) du modèle mathématique indiquent que le départ de toutes les routes doit avoir lieu à l'instant 0. Cette fonction de pénalité décrite par (5.16), mesure l'écart entre 0 et l'instant de départ. Elle contient deux parties et forme un "V" sur 0 comme représenté par la figure 5.1.

$$U_0^p(t) = \begin{cases} -t & \text{si } t < 0 \\ t & \text{si } t > 0 \end{cases} \tag{5.16}$$

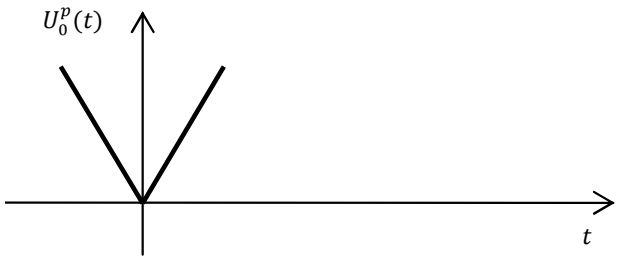


FIGURE *5.1: Fonction de pénalité du dépôt*

Les fonctions associées aux clients non réguliers associent un coût de 0 à leur service. Ces clients ponctuels ne dépendent en effet d'aucun autre et par conséquent aucune contrainte de régularité horaire ne pèse sur eux. La fonction de coût les caractérisant vaut simplement 0 partout comme indiqué par (5.17).

$$U_i^p(t) = 0 \qquad \forall i \in V \mid N_i = 1 \tag{5.17}$$

Les fonctions associées aux clients réguliers doivent pénaliser une arrivée se produisant à plus de L des arrivées chez ce même client aux autres périodes. Ceci est fait en définissant pour chacun de ces clients $i \in V \mid N_i > 1$ à chaque période $p \in P \mid w_i^p > 0$ une fonction $u_i^p(t)$ qui représente la pénalité induite par l'arrivée courante à t_i^p chez ce client. Cette fonction pénalise toute arrivée située à une distance de plus de L de t_i^p. Il s'agit d'une fonction convexe, linéaire

par morceaux en trois parties telle que décrite par (5.18). Une représentation graphique de cette fonction est donnée par la figure 5.2.

$$u_i^p(t) = \begin{cases} t_i^p - L - t & \text{si } t < t_i^p - L \\ 0 & \text{si } t_i^p - L < t < t_i^p + L \\ t - t_i^p - L & \text{si } t > t_i^p + L \end{cases} \tag{5.18}$$

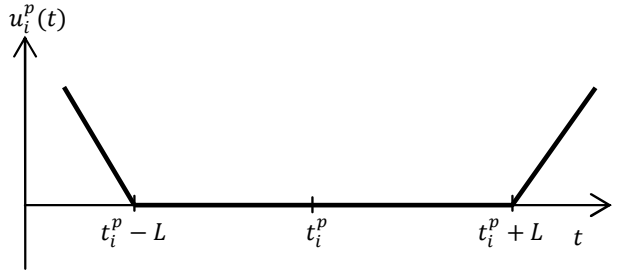


FIGURE *5.2: Pénalité induite par l'arrivée chez le client i à la période p*

La fonction de pénalité de régularité horaire du client $i \in V \mid N_i > 1$ au cours de la période $p \in P \mid w_i^p > 0$ est obtenue en sommant les pénalités $u_i^p(t)$ générées à toutes les autres périodes $p' \neq p$ (5.19). La construction de cette fonction est illustrée par la figure 5.3.

$$U_i^p(t) = \sum_{\substack{p' \in P \mid w_i^{p'} > 0 \\ p' \neq p}} u_i^{p'}(t) \qquad \forall i \in V \mid N_i > 1, \forall p \in P \mid w_i^p > 0 \tag{5.19}$$

5.4.3 Pré-calculs de données

Les mouvements effectués lors de la construction ou la modification d'une solution de *Consistent VRP* doivent pouvoir être évalués efficacement. De la même manière que décrit en 4.2.3.3 pour le *PVRPTS*, ces mouvements sont définis comme des découpes et des concaténations de séquences σ de clients. S'agissant du *Consistent VRP*, nous avons besoin des données suivantes pour chacune des séquences σ : sa charge $Q(\sigma)$, sa durée $C(\sigma)$ ainsi que deux fonctions de coût linéaires par morceaux $F(\sigma)(t)$ et $B(\sigma)(t)$ relatives au coût de régularité horaire. La fonction de coût avant (*forward cost function* en anglais) $F(\sigma)(t)$ donne le coût d'effectuer la séquence σ si son dernier client est servi exactement à l'instant t. La fonction de coût arrière (*backward cost function* en anglais) $B(\sigma)(t)$ donne le coût de servir cette séquence si le premier client est servi à l'instant t. Nous rappelons les modalités

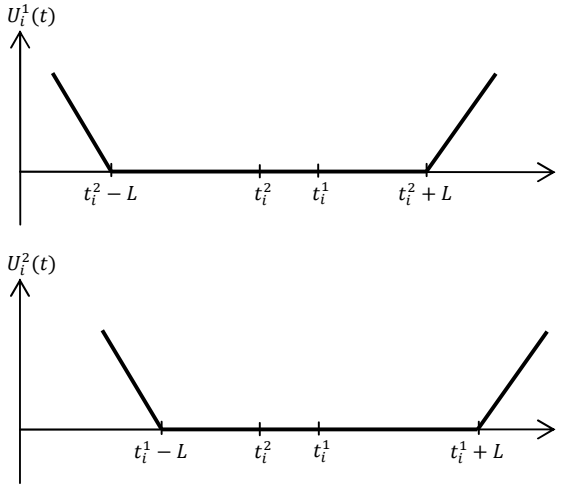


FIGURE *5.3: Fonctions de pénalité $U_i^p(t)$ pour le client i aux périodes 1 et 2*

d'extension de ces données pour une séquence $\sigma \oplus \sigma'$ résultant de la concaténation de deux séquences σ et σ' :

$$
\begin{aligned}
D(\sigma \oplus \sigma') &= D(\sigma) + D(\sigma') + \tau_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}} \\
Q(\sigma \oplus \sigma') &= Q(\sigma) + Q(\sigma') \\
F(\sigma \oplus \sigma')(t) &= F(\sigma)(t - \tau_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}} - D(\sigma')) + F(\sigma')(t) \\
B(\sigma \oplus \sigma')(t) &= B(\sigma')(t + \tau_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}} + D(\sigma)) + B(\sigma)(t)
\end{aligned}
$$

On peut, comme décrit dans 4.2.3.3, définir de manière identique les séquences $\sigma \ominus^b \sigma'$ et $\sigma \ominus^f \sigma'$ qui sont obtenues en retirant respectivement le suffixe ou le préfixe σ' de la séquence σ. Ainsi que son coût minimum de régularité horaire $Z^*(\sigma \oplus \sigma')$:

$$
Z^*(\sigma \oplus \sigma') = \min_t(F(\sigma)(t) + B(\sigma')(t + \tau_{\sigma_{(\|\sigma\|)}\sigma'_{(1)}})) \tag{5.20}
$$

Ces données sont calculées récursivement sur les séquences "préfixes" et "suffixes" des routes d'une solution complète ou partielle de *Consistent VRP*, de la même manière que décrit en 4.2.3.3. Ce calcul est entrepris à chaque fois que la solution se trouve modifiée.

5.4.4 Heuristique constructive aléatoire

La méthode proposée nécessite la construction de solutions de départ. Pour construire ces solutions, nous nous basons sur le concept du "squelette" utilisé dans la littérature. Une solution

de VRP classique est construite avec l'ensemble des clients de l'instance considérée au moyen de l'heuristique de Clarke et Wright [18] qui a été "randomisée". A chaque itération, l'arc ajouté à la solution en cours de construction est choisi aléatoirement parmi les 3 meilleurs. Cette solution constitue le squelette. Ce squelette est ensuite utilisé pour générer chaque jour de livraison. Les clients ne devant pas recevoir de livraison, ou ceux servis uniquement le jour considéré, sont retirés. Les clients servis sur plusieurs jours sont toujours placés dans la même route, la régularité de conducteur est donc respectée "par construction". Le fait qu'ils respectent également le même séquencement favorise une violation moindre de la contrainte de régularité horaire. Une fois toutes les journées de livraison construites, les clients ne requérant de service qu'à une seule journée sont placés au moyen d'une heuristique d'insertion gloutonne. Cette heuristique effectue à chaque itération la meilleure insertion possible au regard de la fonction de coût $f(S)$ prenant en compte les violations de contraintes. En effet, si la solution obtenue respecte forcement les contraintes de régularité de conducteur, il n'est pas garanti à cette étape de pouvoir produire une solution respectant également la régularité horaire, la capacité des véhicules ainsi que la durée maximum d'une tournée. C'est donc une solution la moins pénalisée possible qui est construite ici et non nécessairement une solution réalisable.

5.4.5 Perturbation

Afin de permettre l'exploration des bassins d'attractions adjacents, la solution en cours subit une perturbation avant d'être soumise à la procédure de recherche locale. Si d'ordinaire cette perturbation doit être éloignée des mouvements de recherche locale considérés, nous avons constaté que la structure relativement "figée" d'une solution de *Consistent VRP* due aux contraintes de chauffeurs demande une perturbation très faible. Ainsi, la perturbation choisie ici consiste en l'application d'un mouvement de type "Relocate" au hasard. Ce degré de perturbation suffit ici à entrainer la recherche locale vers des solutions voisines. Les solutions choisies pour entrer dans cette procédure de perturbation sont par défaut la meilleure réalisable et la meilleure irréalisable des $2 \times N_e$ derniers enfants générés. Toutefois, si le nombre d'itérations sans amélioration atteint la valeur *seuil*, la meilleure solution connue depuis le dernier redémarrage $\bar{S}$ est utilisée. Cette procédure a pour effet de recentrer la recherche dans un bassin d'attraction plus prometteur que celui vers lequel elle a cheminé.

5.4.6 Recherche Locale

La procédure de recherche locale pour le *Consistent VRP* comporte deux phases. La première est une phase d'amélioration de la solution du point de vue de la régularité en terme de chauffeur. Tant que la solution n'est pas valide au regard de cette contrainte, cette procédure effectue le meilleur mouvement "Relocate" possible avec un client régulier. Ce mouvement est choisi parmi ceux déplaçant un client régulier sur le véhicule avec lequel il est le plus souvent affecté dans la solution courante. Lorsque pour un tel client le nombre de services qu'il reçoit se partage de manière égale entre plusieurs véhicules, tous les mouvements impliquant ces véhicules sont évalués. Il résulte de cette opération une solution réalisable du point vue de la contrainte de régularité de chauffeur mais dont les autres contraintes ont pu être violées. Une seconde procédure de recherche locale, identique à celle employée pour le *PVRPTS*, est alors appelée. Les mouvements effectués sont de type "Relocate", "Swap", "2-opt" et "2-opt*". Ils sont choisis parmi ceux introduisant dans la solution les arcs se trouvant parmi les $\pi\%$ moins longs. La longueur maximum des séquences déplacées est fixée à $l = 2$. Toutes les informations calculées sur les sous-séquences intérieures $\sigma_{i,j}$ entre deux applications de mouvement sont conservées dans une matrice de taille $n \times n$ pour être réutilisées au besoin. On notera qu'aucune information ne concernant le degré de violation de la contrainte de régularité de chauffeur n'est pré-calculée. Cette quantité ne dépend en effet pas de la séquence mais du véhicule auquel elle est affectée. Elle n'est pas utilisée sur les déplacements intra-route et doit de toute manière être recalculée pour chaque client déplacé dans un mouvement inter-route. La longueur des séquences déplacées étant limitée à $l = 2$, le coût induit en temps de calcul reste acceptable. La première phase est ensuite rappelée pour réparer les violations de contraintes de régularité occasionnée et la solution obtenue est retournée.

5.4.7 Réparation

La procédure de réparation est appelée avec une probabilité Π lorsque la solution obtenue à la fin de la recherche locale n'est pas réalisable. Cette procédure renvoie la solution dans la recherche locale en multipliant les coefficients de pénalités concernés par 10. Si la solution en résultant n'est toujours pas réalisable, un dernier essai est tenté en multipliant les coefficients pénalités par 100. Lorsque la procédure de recherche locale est appelée pour réparation, la première phase n'est pas effectuée car la solution considérée ne viole pas la contrainte de régularité de chauffeur.

5.4.8 Mise à jour des coefficients de pénalités

Les coefficients de pénalité γ, ϕ, ξ et ψ associés aux contraintes de capacité des véhicules, aux contraintes de durées des routes, aux contraintes de régularité de conducteur et au contraintes horaires sont mis à jour à chaque génération des $2 \times N_e$ enfants. Pour chaque type de contraintes, le coefficient correspondant est multiplié (respectivement divisé) par 10 si le nombre de solutions infaisables pour cette contrainte sur les derniers $2 \times N_e$ enfants générés est plus grand qu'un seuil ∇_{max} (resp. plus petit qu'un seuil ∇_{min}). Notons que la comptabilité du nombre de solutions irréalisables du point de vue de la contrainte de chauffeur est faite à la fin de la deuxième phase de la procédure de recherche locale. En effet, il n'y a que lors de cette phase que la pénalité associée est utilisée. Les compteurs des autres contraintes sont mis à jour quant à eux à la fin de la procédure de recherche locale.

5.5 Problème de tournées régulières avec décalage des dates de départ des routes

Comme le suggèrent Tarantilis et al. [115], il est possible d'obtenir de meilleures solutions en relaxant la contrainte de départ des routes à l'instant 0, peu réaliste en pratique. Si les départs des routes peuvent être décalés vers l'avant, il permettent de satisfaire les contraintes de régularité horaire pour un temps de trajet moindre. Kovacs et al. [68] proposent alors le *Consistent VRP* avec instants de départ décalables (*Consistent VRP* with shiftable starting times). Nous nous intéressons dans cette section à cette version du problème.

5.5.1 Détermination des instants départ pour le *Consistent VRP*

De manière analogue à celle décrite en 3.4 pour le *PVRPTS*, un sous-problème d'ajustement des instants de départ des routes peut être défini pour le *Consistent VRP*. Une solution du *Consistent VRP* se compose de p périodes chacune composée de routes $r \in R$ avec $R \subseteq \mathscr{R}$ où $\mathscr{R}$ est l'ensemble des routes qu'il est possible d'extraire du graphe G. Le problème de décision, que nous nommerons Starting Time Problem for Consistent VRP ($STPConVRP$), consiste à déterminer les décalages Δ_r des instants de départ de chacune des routes $r \in R$ afin que les arrivées chez un même client d'une route à l'autre ne soit pas espacées temporellement de plus de L. L'ensemble des routes de R qui contiennent le client i est noté $R(i)$ et t_i^r est l'instant d'arrivée au client i dans la route r lorsque le départ de cette route est pris à l'instant 0 (i.e. sans décalage). Les temps d'attentes ne sont pas autorisés dans le *Consistent VRP*,

par conséquent un décalage sur l'instant de départ d'une route induit le même décalage sur les instants d'arrivée t_i^r de tous les clients de cette route. Pour que les arrivées aux clients ne surviennent pas trop tard, on impose qu'une route ne finisse pas après T. On définit ainsi un décalage maximum $\Delta_r^{max} = \min_{i \in r} \{T - t_i^r - s_i - d_{i0}\}$ que peut subir une route sans finir après T. le problème $STPConVRP$ peut alors être écrit comme suit :

(STPConVRP) : Fixer les $\Delta_0, \Delta_1, ..., \Delta_r$ avec $r \in R$ tels que :

$$(t_i^r + \Delta_r) - (t_i^{r'} + \Delta_{r'}) \leq L \qquad \forall i \in V \mid N_i > 1, \forall r, r' \in R(i), r \neq r' \tag{5.21}$$

$$\Delta_r \leq \Delta_r^{max} \qquad \forall r \in R \tag{5.22}$$

$$\Delta_r \in \mathbb{R}^+ \qquad \forall r \in R \tag{5.23}$$

Les contraintes (5.21) imposent le respect de l'espacement maximum L entre deux arrivées chez un même client. Elles ne sont définies que pour chaque paire de routes r et r' contenant un même client régulier i. Le décalage maximum acceptable pour chaque route est respecté grâce à (5.22) et pour finir (5.23) précisent le domaine de définition des variables.

Contrairement à son homologue défini en 3.4 pour le $PVRPTS$, ce sous-problème d'ajustement des instants de départ n'est pas NP-Complet. En effet, il faut remarquer que les contraintes de régularité des arrivées (5.21) sont linéaires. Ce n'est pas le cas des contraintes d'espacement (3.18) du $PVRPTS$ qui nécessitent, pour être linéaires, l'introduction de variables binaires. Le problème peut se ramener à un problème d'ordonnancement de projet et se modéliser au moyen de la méthode des potentiels Metra (MPM) [111] développée par Bernard Roy en 1958. Cette méthode s'appuie sur un graphe orienté $G = (V, A)$ dans lequel les arcs $(i, j) \in A$ associés aux délais δ_{ij} représentent les contraintes de précédence temporelles $t_j - t_i \geq \delta_{ij}$ entre les débuts t_i et t_j de deux activités $i, j \in V$. L'existence d'un ordonnancement satisfaisant ces contraintes est alors conditionnée par la non-existence d'un circuit de longueur positive dans le graphe G [25]. Il existe plusieurs algorithmes polynomiaux de recherche de circuit positif dans les graphes, citons par exemple l'algorithme BFCT de Tarjan [116] de complexité $O(|V|\,|A|)$ dérivé de l'algorithme de Bellman-Ford. Cet algorithme a récemment été utilisé par Masson et al. [77] pour vérifier la faisabilité de contraintes temporelles complexes dans un problème de transport de passagers avec points de transferts. La figure 5.4 illustre la modélisation du $STPConVRP$ par la méthode MPM. Un exemple simple avec deux routes est proposé. Dans la modélisation MPM, les deux sommets D_1 et D_2 modélisent les dépôts des routes 1 et 2, le sommet étiqueté "0" modélise, quant à lui, l'instant 0.

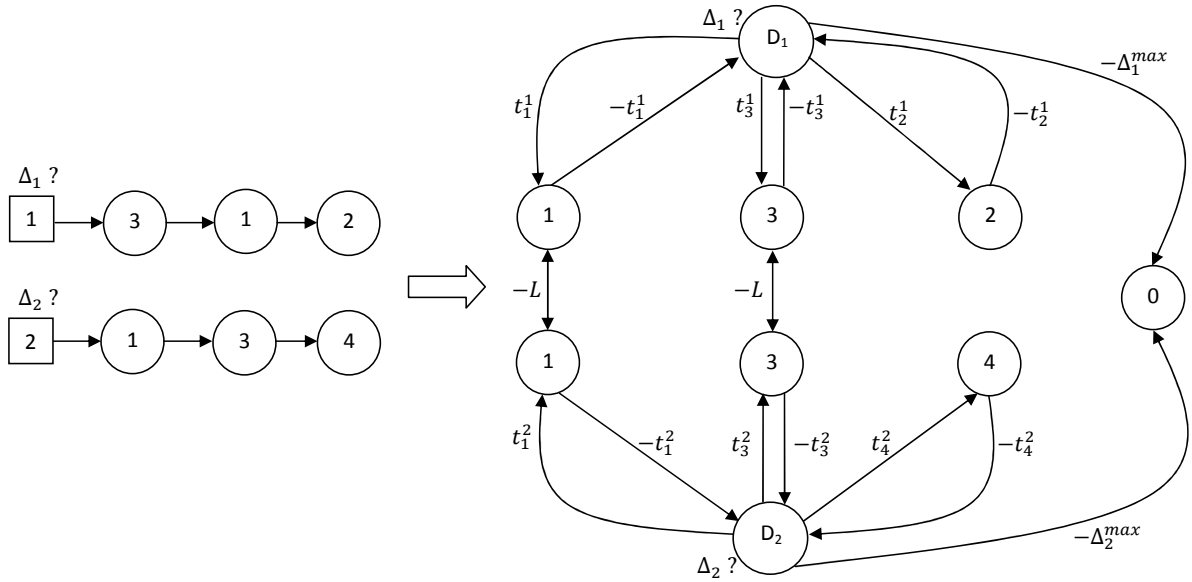


FIGURE *5.4: Le STPConVRP modélisé avec la méthode MPM*

5.5.2 Adaptation de la MS-ILS

La possibilité de décaler les instants de départ des routes est prise en compte très simplement par la MS-ILS proposée. La première modification consiste à comptabiliser le coût d'arrivée au dépôt de fin de route afin de pénaliser les routes se terminant après l'instant T. La seconde modification concerne la fonction de pénalisation horaire $U_0^p(t)$ associée au dépôt. Cette fonction comporte à présent trois parties et permet de pénaliser un départ avant l'instant 0 et une arrivée après l'instant T, elle est décrite par (5.24) et représentée graphiquement sur la figure 5.5.

$$U_0^p(t) = \begin{cases} -t & \text{si } t < 0 \\ 0 & \text{si } 0 < t < T \\ t - T & \text{si } t > T \end{cases} \tag{5.24}$$

Kovacs et al. [68] proposent de réajuster les instants de départ des routes dans une phase de post-optimisation au cours de laquelle ils font appel soit au solveur CPLEX, soit à une procédure heuristique. La méthode que nous proposons permet la prise en compte, dans l'évaluation même des mouvements de recherche locale, de la possibilité de différer le départ des routes. Lors de l'application d'un mouvement, l'instant de départ de chaque séquence σ concernée est fixé à $arg\ Z^*(\sigma)$. Les départs des routes impliquées sont donc réajustés lors de chaque mouvement.

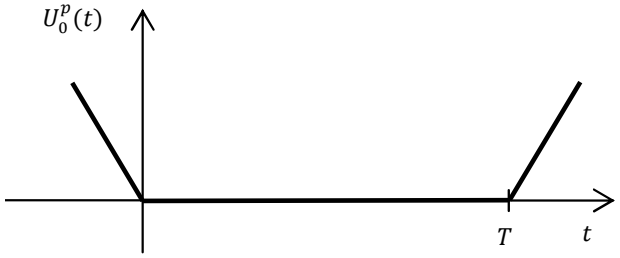


FIGURE *5.5: Adaptation de la fonction de pénalisation du dépôt*

5.6 Expérimentations numériques

5.6.1 Jeux de tests utilisés

La méthode a été implémentée avec le langage C# et testée sur un PC Intel Core i5 cadencé à 2.8 GHz avec 4 GB de mémoire sous Windows 7 Professional en version 32 bits. Les évaluations numériques ont été conduites sur deux jeux d'instances. Le premier est composé des 12 instances initialement définies par Groër et al. [58]. Il s'agit d'une adaptation des instances classiques de VRP de Christofides et Eilon [16] considérées sur cinq jours avec une probabilité $p = 0.7$ de service journalier chez un client. Ces instances permettent de confronter les résultats de la méthode à ceux de trois autres métaheuristiques de la littérature. Le second jeu d'instances est tiré de celui proposée par Kovacs et al. [68], il s'agit d'une extension du jeu de test précédent avec de nouvelles probabilités de service et des valeurs limites de régularité horaire L différentes. Un de ces jeux d'instance, avec des valeurs de L très faibles, est utilisé afin d'évaluer le gain que procure la possibilité de décaler les instants de départ des routes.

5.6.2 Calibrage des paramètres

Les paramètres de la MS-ILS à calibrer sont au nombre de cinq : le nombre d'enfant générés N_e, la proportion de voisinage exploré π, le nombre d'itérations sans amélioration $NbManque$, la valeur *seuil* de redémarrage à partir de la meilleure solution connue et pour finir la probabilité de réparation Π. Le paramètre $NbManque$ a été calibré expérimentalement en observant les "paliers" d'itérations sans amélioration sur plusieurs instances. Ces observations nous ont rapidement conduit à opter pour une valeur de $NbManque = 10$. Le paramètre *seuil* a ensuite pu être fixé et une valeur de 8 semble satisfaisante. Les expérimentations ont ensuite révélé que le nombre d'enfants N_e et le pourcentage de voisinage exploré

influaient grandement sur la qualité de la méthode. Il a été décidé de calibrer ces deux valeurs simultanément en fixant $NbManque = 10$ et $seuil = 8$. La réparation de solutions ne pouvant qu'améliorer la qualité de la méthode, nous considérons à cette étape $\Pi = 0$. La valeur finale de Π sera déterminée une fois celle de N_e connue. Un ensemble d'instances ont été choisies dans les jeux de tests proposés et ont été exécutées dix fois chacune durant 20 minutes pour tous les couples (N_e, π) avec $N_e \in \{2, 5, 10, 20\}$ et $\pi \in \{40, 20, 10, 5, 1\}$. Les valeurs des résultats obtenus pour chaque instance ont ensuite été ramenées entre 0 et 1. Cette procédure permet d'exprimer les coûts sur une même échelle, évitant qu'une instance dans laquelle les coûts de solution sont élevés n'ai trop d'influence. Les résultats de cette procédure sont donnés dans le tableau 5.1. Ils indiquent clairement que le nombre d'enfants permettant d'obtenir la meilleure qualité de solution est de 10. Un pourcentage de voisinage exploré de 40% ou 20% donne de bons résultats, un léger mieux tout de même pour 20% qui peut s'expliquer par une plus grande rapidité et donc un plus grand nombre de solutions visitées dans les 20 minutes imparties.

$\pi \quad \backslash \quad N_e$	2	5	**10**	20
40	0.536	0.298	**0.199**	0.547
20	**0.322**	**0.413**	**0.173**	**0.432**
10	0.324	0.412	**0.293**	0.439
5	0.487	0.305	**0.275**	0.258
1	0.749	0.612	**0.484**	0.538

TABLE *5.1: Résultats de calibration des paramètres N_e et π*

La probabilité de réparation Π est le dernier paramètre à être calibré. La méthode est à nouveau exécutée 10 fois sur chaque instance de calibration et pour chaque $\Pi \in \{0, 0.2, 0.7, 1\}$. Les résultats sont ramenés entre 0 et 1 pour chaque instance et une valeur de $\Pi = 0.2$ permet d'obtenir le meilleur score.

Les coefficients de pénalité γ, ϕ, ξ et ψ, sont initialisés à 1 au démarrage de l'algorithme. Nous souhaitons pour finir qu'au moins la moitié des individus générés à chaque itération soient réalisables. Pour $N_e \times 2 = 20$ individus générés, la valeur cible est donc de 10. Si nous fixons les seuils à + ou - 10% de cette valeur nous obtenons $\nabla_{min} = 9$ et $\nabla_{max} = 11$.

5.6.3 Résultats sur les instances classiques

Ces douze instances proposée par Groër et al. [58] contiennent entre 50 et 199 clients et un horizon de 5 jours. Elles ont été générées avec une probabilité de service journalier de 0.7 pour chaque client. Dans le modèle mathématique qu'ils donnent, Groër et al. [58] font figurer le paramètre L définissant le degré de régularité horaire. Ce paramètre n'est toutefois pas utilisé dans la méthode de résolution qu'il décrivent et n'est pas non plus défini pour les instances qu'ils fournissent. Comme ils reportent dans leurs résultats l'écart maximum l_{max} constaté entre deux arrivées à un client régulier, cette valeur a par la suite été utilisée par Tarantilis et al. [115] puis Kovacs et al. [68] comme valeur de L.

Les résultats sont présentés dans le tableau 5.2 avec le nom de chaque problème utilisé. Les colonnes *ConRTR*, *TTS* et *TALNS* désignent respectivement le Record-To-Record de Groër et al. [58], la Template based Tabu Search de Tarantilis et al. [115] et la Template Adaptative Large Neighborhood Search de Kovacs et al. [68]. Aucune information n'est donnée sur le nombre d'exécutions pour le ConRTR, Tarantilis et al. reportent la meilleure exécution sur 5, quant à Kovacs et al. [68], ils donnent la moyenne sur 10 exécutions. Pour permettre une comparaison avec toutes ces méthodes, les résultats sont donnés en moyenne sur 10 exécutions avec une limite de 20 minutes dans la colonne *MS-ILS Avg* et pour la meilleure exécution dans la colonne *MS-ILS Best*. Les valeurs reportées sont le temps total **TT** et l'écart maximum l_{max} constaté entre deux arrivées à un client régulier. Pour finir, la colonne *Gap % to* donnent le pourcentage d'écart avec chaque méthode X calculé comme suit : $(MS\text{-}ILS\text{-}X)/X \times 100$.

		ConRTR		TTS		TALNS		MS-ILS Avg		MS-ILS Best		Gap % to		
Instance	**Taille**	**TT**	l_{max}	**TT**	l_{max}	**TT**	l_{max}	**TT**	l_{max}	**TT**	l_{max}	**ConRTR**	**TTS**	**TALNS**
Pr.01	50	2282.14	24.38	2210.56	21.99	2154.47	23.67	2229.54	23.10	2162.69	23.90	**-5.23**	**-2.17**	0.38
Pr.02	75	3872.86	34.26	3622.71	27.75	3603.73	32.42	3744.31	32.22	3693.92	31.81	**-4.62**	1.97	2.50
Pr.03	100	3628.22	22.87	3451.10	21.92	3356.11	21.77	3614.91	21.39	3512.64	21.94	**-3.19**	1.78	4.66
Pr.04	150	4952.91	27.53	4572.00	25.15	4589.04	24.24	4860.27	26.10	4748.26	26.89	**-4.13**	3.86	3.47
Pr.05	199	6416.77	26.93	5732.62	19.99	5729.67	21.52	6081.27	24.54	5886.66	25.86	**-8.26**	2.69	2.74
Pr.06	50	4084.24	63.47	4096.87	55.38	4051.48	63.27	4102.47	56.21	4081.71	55.67	**-0.06**	**-0.37**	0.75
Pr.07	75	7126.07	83.96	6752.36	63.28	6814.48	72.31	6948.14	80.87	6806.58	80.59	**-4.48**	0.80	**-0.12**
Pr.08	100	7456.19	73.04	7279.39	62.01	7190.32	64.36	7592.28	68.18	7332.66	69.91	**-1.66**	0.73	1.98
Pr.09	50	11033.54	106.43	10585.10	84.76	10486.43	85.97	10963.13	89.41	10835.06	76.89	**-1.80**	2.36	3.32
Pr.10	199	13916.80	60.17	13120.40	57.17	13196.21	57.67	14015.74	58.26	13878.13	57.79	**-0.28**	5.78	5.17
Pr.11	120	4753.89	16.10	4721.09	15.68	4493.10	15.45	4765.15	15.55	4679.80	13.39	**-1.56**	**-0.87**	4.16
Pr.12	100	3861.35	17.58	3607.88	16.91	3554.50	16.12	3669.84	16.44	3615.94	15.53	**-6.36**	0.22	1.73
Moyenne		6115.42	46.39	5812.67	39.33	5768.30	41.56	6048.92	42.69	5936.17	41.68	**-3.47**	1.40	2.56

TABLE *5.2: Comparaison de MS-ILS avec les approches existantes sur les instances de Christofides modifiées : 5 jours avec une fréquence de livraison de* 0.7

Les résultats montrent que les solutions reportées par Groër et al. [58] sont à chaque fois améliorées sur le temps total de parcours TT avec des écarts allant jusqu'à 8.26%. De plus, l'écart

maximum l_{max} constaté est à chaque fois diminué. Ces résultats sont également proches de ceux obtenus par Tarantilis et al. et les améliorent par trois fois. La méthode s'approche des résultats obtenus par Kovacs et al. mais ne les améliorent que sur le problème 7. Les améliorations sur TTS et TALNS ont à chaque fois lieu sur les instances de plus petite taille. Il semble que la MS-ILS soit moins performante sur les plus grandes instances. Ceci peut s'expliquer par la taille limitée du voisinage exploré par la recherche locale due à l'utilisation coûteuse des fonctions linéaires par morceaux. Rappelons cependant que la méthode de Kovacs et al. [68] a été développée en parallèle de cette thèse et qu'elle n'est pas encore publiée.

5.6.4 Résultats sur les instances étendues

Ces instances ont été proposées par Kovacs et al. [68] et se basent sur le jeu de tests classique. Elles consistent en une variation de ce jeu de tests sur la fréquence de livraison des clients et sur le degré de régularité horaire L imposé. De nouvelles instances avec des fréquences de livraison de 0.5 et 0.9 en plus des originales à 0.7 sont générées. Les tests sont ensuite faits avec des valeurs de $L \in \{L_{0.4}, L_{0.6}, L_{0.8}\}$ valant 40%, 60% ou 80% de la valeur l_{max} obtenue sans borner L. Afin d'évaluer l'intérêt que revêt la possibilité de décaler les instants de départ des routes, un jeu d'instances avec $L_{0.4}$ est choisi, sa fréquence de livraison est de 0.5. Les résultats sont présentés dans le tableau 5.3. Le nom de l'instance, le nombre de clients qu'elle comporte ainsi que la valeur de L considérée sont indiqués dans les colonnes **Instance**, **Taille** et **L**. Les résultats sont donnés pour la TALNS de Kovacs et al. [68] et pour la MS-ILS sans possibilité de décaler les instants de départ des routes dans les parties "TALNS" et "MS-ILS" du tableau. Les sections "TALNS-shift" et "MS-ILS-shift" considèrent les versions avec ajustement des instants de départ. Pour toutes ces méthodes, les valeurs reportées sont le temps total **TT** et l'écart maximum l_{max} constaté entre deux arrivées à un client régulier. Ces valeurs sont des valeurs moyennes sur 10 exécutions, tant pour TALNS que pour MS-ILS. Les meilleures valeurs de temps total (TT) obtenues sont indiquées en gras. Pour finir, la partie "Gain%" du tableau donne le gain procuré par l'ajustement des heures de départ des routes pour chaque méthode X et calculé comme suit : $(X\text{-}shift\text{-}X)/X \times 100$. Pour la colonne **Best**, X est la meilleure méthode sur l'instance.

Les résultats montrent que MS-ILS s'avère très compétitive sur des instances ou les contraintes de régularité sont fortes, elle fait mieux que TALNS sur 11 des 12 instances proposées sans ajustement des instants de départ avec un écart des moyennes de plus de 50%. Une fois l'ajustement des instants de départ considéré, TALNS reprend l'avantage mais l'écart des moyennes n'est plus aussi fort avec 3.45%. Le gain que procure l'ajustement

			TALNS		MS-ILS		TALNS-shift		MS-ILS-shift		Gain%		
Instance	**Taille**	**L**	**TT**	l_{max}	**TT**	l_{max}	**TT**	l_{max}	**TT**	l_{max}	**TALNS**	**MS-ILS**	**Best**
Pr.01	50	27	1850.18	25.44	**1804.20**	23.93	**1683.10**	27.00	1712.90	27.00	-9.03	-5.06	-6.71
Pr.02	75	20	3585.00	18.39	**2807.10**	18.76	**2633.04**	20.00	2711.93	20.00	-26.55	-3.39	-6.20
Pr.03	100	23	4348.61	22.18	**3096.14**	22.72	**2718.16**	23.00	2847.99	23.00	-37.49	-8.01	-12.21
Pr.04	150	18	14136.58	16.52	**4176.59**	17.71	**3678.99**	18.00	3734.85	18.00	-73.98	-10.58	-11.91
Pr.05	199	15	20650.27	11.68	**5233.25**	14.87	**4116.86**	15.00	4580.82	15.00	-80.06	-12.47	-21.33
Pr.06	50	35	3820.50	31.37	**3136.61**	32.97	**2923.89**	35.00	3000.09	35.00	-23.47	-4.35	-6.78
Pr.07	75	30	8360.45	25.24	**5377.35**	29.01	**4790.57**	30.00	4980.51	30.00	-42.70	-7.38	-10.91
Pr.08	100	41	9908.53	39.10	**6143.72**	40.10	**5370.20**	41.00	5595.27	41.00	-45.80	-8.93	-12.59
Pr.09	50	39	15777.09	36.49	**9116.52**	38.24	**7601.88**	39.00	7892.92	39.00	-51.82	-13.42	-16.61
Pr.10	199	29	24670.81	28.03	**11563.61**	28.81	11476.61	29.00	**10782.93**	29.00	-53.48	-6.75	-6.75
Pr.11	120	19	**3633.71**	18.53	4187.67	17.99	**3371.71**	18.64	4114.39	19.00	-7.21	-1.75	-7.21
Pr.12	100	11	10287.67	8.47	**3398.77**	10.83	**2923.44**	11.00	3175.26	11.00	-71.58	-6.58	-13.99
Moyenne			10085.78	23.45	**5003.46**	24.66	**4440.70**	25.55	4594.16	25.58	-43.60	-7.39	-11.10

TABLE *5.3: Comparaison de MS-ILS avec TALNS sur les instances étendues avec une fréquence de livraison de* 0.5, $L = L_{0.4}$ *et dates de départ ajustables*

des instants de départ est bien visible. Il est très important, avec une valeur moyenne de 43.60% pour TALNS à cause des mauvais résultats obtenus sur la version des instances ne permettant pas de décalage. Pour MS-ILS, un gain moyen plus cohérent de 7.39% est obtenu. Une valeur de gain moyen plus juste calculée à partir des meilleures valeurs obtenues, avec et sans considérer le réajustement des instants de départ des routes, est de 11.10%.

5.7 Conclusion

Dans ce chapitre nous avons abordé le problème de tournées de véhicules régulières. Contrairement au *PVRPTS* qui exige l'irrégularité, il s'agit ici de maintenir un degré de régularité. Si leurs principes les opposent, ces deux problèmes ont en commun l'interdépendance des horaires de service chez les clients. De la même manière que pour le *PVRPTS*, un sous-problème d'ajustement des instants de départ a été défini pour le problème de tournées régulières. Ce sous-problème s'avère moins "difficile" que celui du *PVRPTS* qui est NP-Complet. La méthode MS-ILS proposée au chapitre 4 a été adaptée pour traiter le problème de tournée de véhicules régulières en conservant la procédure de recherche locale basée sur les fonctions de coût linéaires par morceaux. La méthode proposée s'avère efficace, se montrant compétitive avec les métaheuristiques de la littérature dédiées à ce problème. Elle se montre de plus extrêmement flexible, permettant de traiter le cas où les instants de départ des routes sont ajustables en modifiant simplement le profil des fonctions de coût associées au dépôt.

Chapitre 6

Mise en place des solutions proposées chez Nexxtep Technologies

6.1 Introduction

Dans le cadre de cette thèse CIFRE, une partie du travail consiste à mettre en application le fruit du travail de recherche. Dans ce chapitre sont décrit les développements qui ont conduit à la réalisation du logiciel d'optimisation de tournées de véhicules proposé par Nexxtep Technologies. Une première partie décrit le premier simulateur réalisé. L'évolution vers le logiciel commercial est ensuite décrite et une synthèse de ces développements est proposée.

6.2 Un premier simulateur

Au début du projet un premier simulateur basique a été réalisé avec le langage C# afin de se familiariser avec la programmation d'heuristiques d'optimisation de tournées de véhicules. Il permet de créer une instance de manière intuitive, de la résoudre au moyen d'une heuristique programmée et de visualiser ensuite la solution obtenue. Il est ainsi possible de comparer visuellement les différentes heuristiques. Le travail de développement est facilité par la possibilité de visualiser les résultats. Un problème est quasi immédiat à détecter sur une représentation graphique alors que l'analyse d'une solution encodée dans une structure de données du programme s'avère beaucoup plus fastidieuse. La figure 6.1 montre la vue principale de ce simulateur. La création de l'instance se fait par simples clics sur la zone blanche aux endroits ou l'on souhaite positionner les clients et le dépôt.

Il est ensuite possible, en cliquant dessus, d'éditer les clients afin de leur affecter un temps

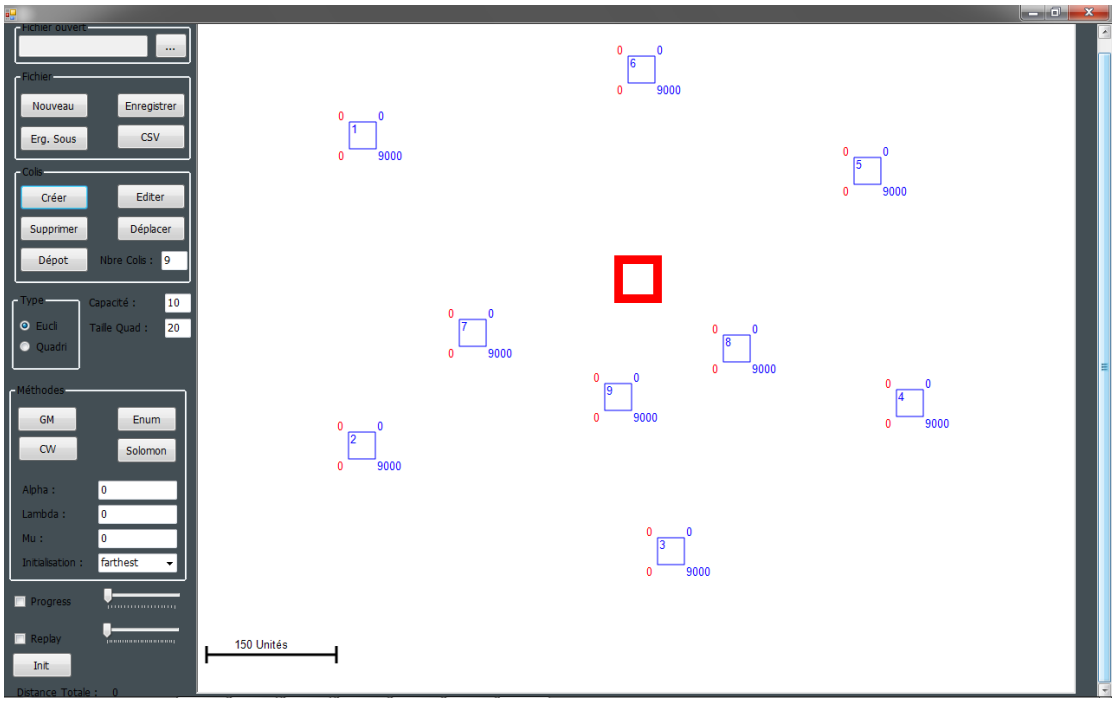


FIGURE *6.1: Vue générale du simulateur*

opératoire, des quantités de marchandise à livrer ou à récupérer et une fenêtre horaire. L'édition se fait dans un formulaire qui apparait au clic sur le client concerné comme le montre la figure 6.2. Les dates de début et de fin de fenêtre horaire d'un client sont indiquées en bleu sur sa gauche et ses dates de début et de fin de service effectives sont indiquées en rouge sur sa droite. Un client peut aussi être supprimé ou bien déplacé par un simple "glisser-déposer". La capacité des véhicules de la flotte considérée est ajustée dans le menu principal.

Une fois l'instance préparé la résolution peut ensuite être effectuée au moyen de l'heuristique choisie. La solution est alors tracée en faisant apparaitre les arcs utilisés et les dates d'arrivée et de départ chez les clients sont mises à jour. Il est alors possible de contrôler visuellement la cohérence d'une solution. L'impact des fenêtres horaires est facilement observable et une violation est identifiée rapidement.

Il est possible d'enregistrer une instance pour pouvoir la réutiliser ultérieurement. Afin de pouvoir traiter facilement des instances de la littérature comme le montre la figure 6.4, il est également possible de charger des instances au format CSV.

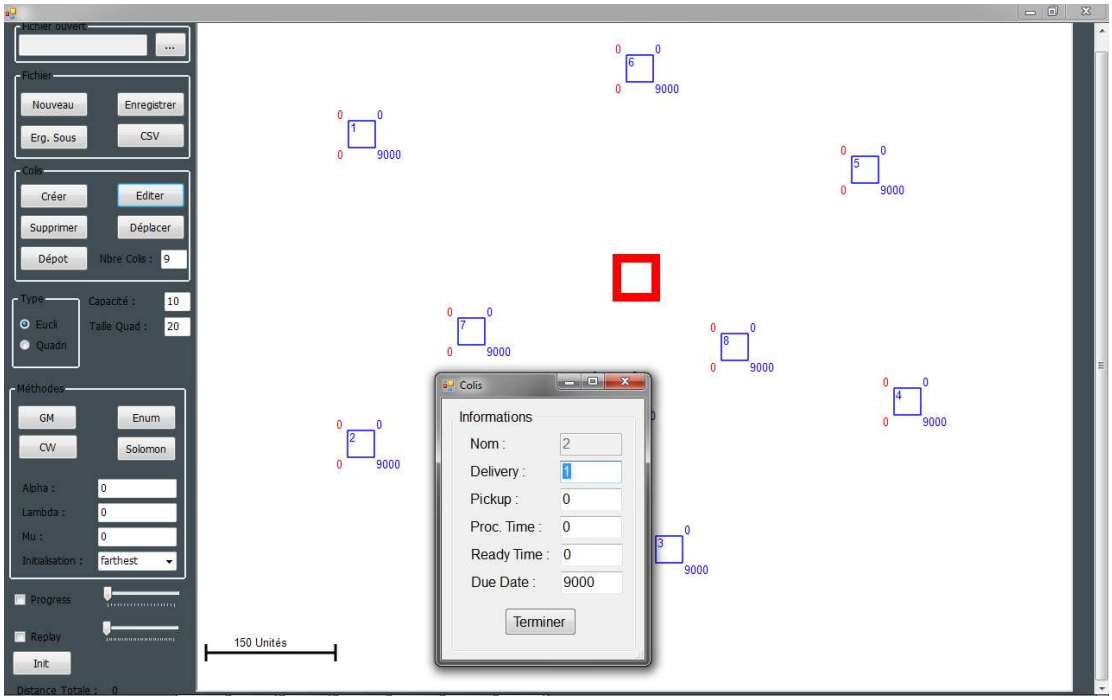


FIGURE *6.2: Exemple d'édition d'un client; ici le client 2.*

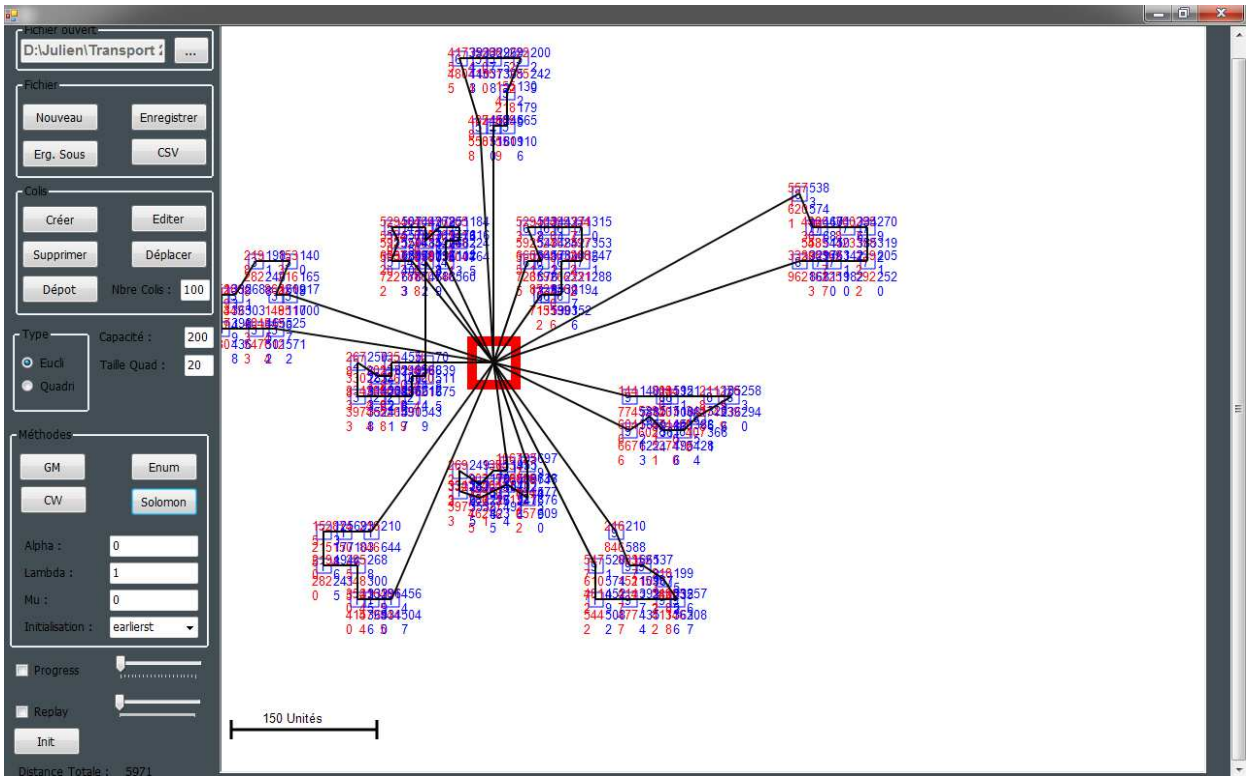


FIGURE *6.4: Instance C101 de Solomon*

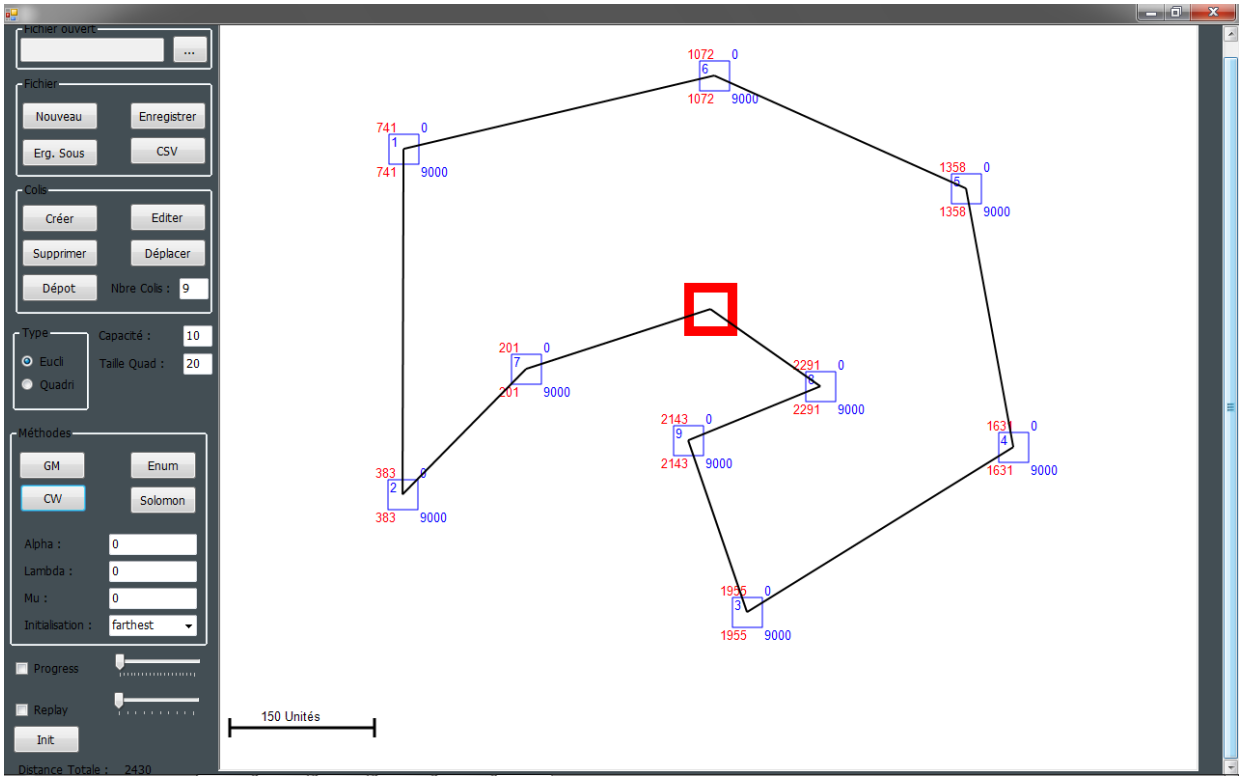


FIGURE *6.3: Résultat obtenu avec l'heuristique de Clarke et Wright*

6.3 Le logiciel

Le simulateur décrit précédemment est réservé à un usage de développement et n'a en aucun cas vocation à être manipulé par un client final. Son ergonomie et son mode d'affichage ne sont en effet pas adaptés. Le planificateur d'une entreprise de transport à besoin d'un outil avec lequel il puisse gérer ses clients et les afficher sur une vraie cartographie. Il a également besoin de travailler avec des distances et des temps de trajet qui correspondent à la réalité. En effet, la distances entre deux clients n'est pas la distance euclidienne mais la distance par la route. Comme toutes les routes ne sont pas en double sens, la distance réelle n'est pas symétrique (i.e. il est possible que $c_{ij} \neq c_{ji}$). Les temps de trajets dépendent également des limitations de vitesse sur les routes et des véhicules utilisés. Un véhicule de transport de marchandises de plus de 3,5 tonnes ne peut pas circuler à plus de 90 km/h sur autoroute comme le ferait un véhicule léger. Il manque donc à notre premier simulateur une architecture correcte, une base de données et une interface-utilisateur digne de ce nom. Cette section détaille ces trois éléments décrit comment ils sont intégrés dans le logiciel.

6.3.1 L'architecture

La première amélioration apportée a été l'ajout d'une cartographie afin de pouvoir y afficher des clients et représenter les trajet réels. La première cartographie que nous avons intégrée à été celle de Microsoft MapPoint 2009. Ce choix a été motivé par l'API .NET disponible avec le logiciel qui permettait une intégration facile. Une vue de cette première version est donnée par la figure 6.5.

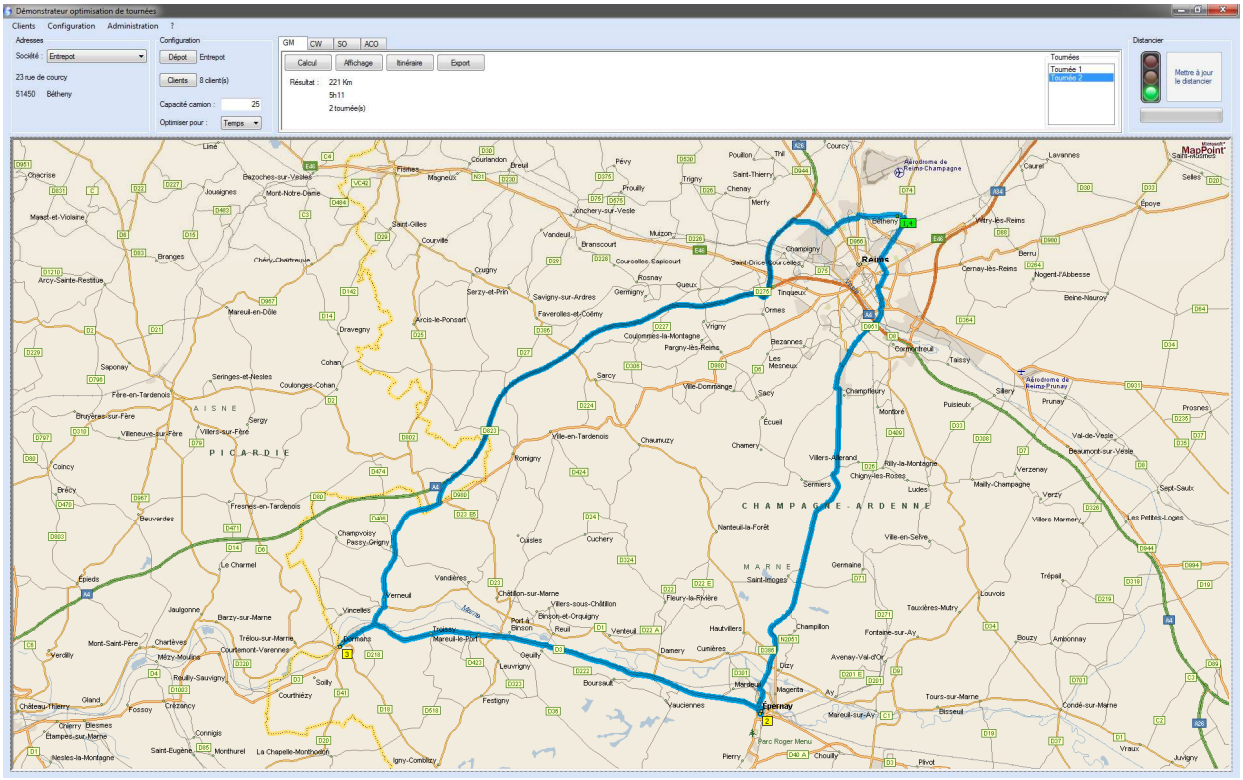


FIGURE *6.5: Première version du logiciel utilisant Microsoft MapPoint*

L'architecture de cette solution était alors très simple : tous les éléments, base de données et cartographie étaient installés sur le poste de l'utilisateur. Aucune connexion Internet n'était donc nécessaire à son fonctionnement, ce qui était son seul avantage. Les tests de montée en charge ont en effet fait apparaitre la lenteur de cette solution et son manque d'ergonomie. Le temps d'affichage des clients sur la cartographie étaient extrêmement long, rendant l'utilisation pénible. De part sa construction, cette solution rendait également difficile son installation et sa maintenance chez le client. L'installation d'un moteur de base de données sur un poste de travail n'est en effet jamais une chose aisée. La base de données doit pouvoir ensuite être maintenue facilement et suivre les mises à jour éventuelles du logiciel. Il en est de même pour le logiciel de cartographie qui doit être installé sur le poste utilisateur

et mis à jour régulièrement. Face à ce manque de flexibilité et d'ergonomie, il a donc été décidé de repenser entièrement la structure du système.

Nous avons opté pour une architecture dans laquelle la base de données se trouve sur un serveur déporté. Ce serveur peut se trouver soit chez le client, soit chez un hébergeur auquel cas les requêtes doivent se faire via Internet. La communication avec le serveur se fait de manière sécurisée grâce au protocole SSL (Secure Sockets Layer en anglais). Le logiciel de cartographie à également été supprimé et remplacé par un composant utilisant Google Maps via Internet. La gestion des licences utilisateurs est faite au moyen d'un serveur appartenant à Nexxtep Technologies. Au lancement de l'application l'utilisateur doit s'authentifier, une requête est alors envoyée au serveur de licence qui, en fonction des droits que possède cet utilisateur, retourne les accès correspondants. Ce fonctionnement permet de définir des utilisateurs "SMART" qui disposent par exemple d'un accès à une base de données privée pour effectuer des simulations, ou encore de droits étendus. La figure 6.6 illustre cette architecture.

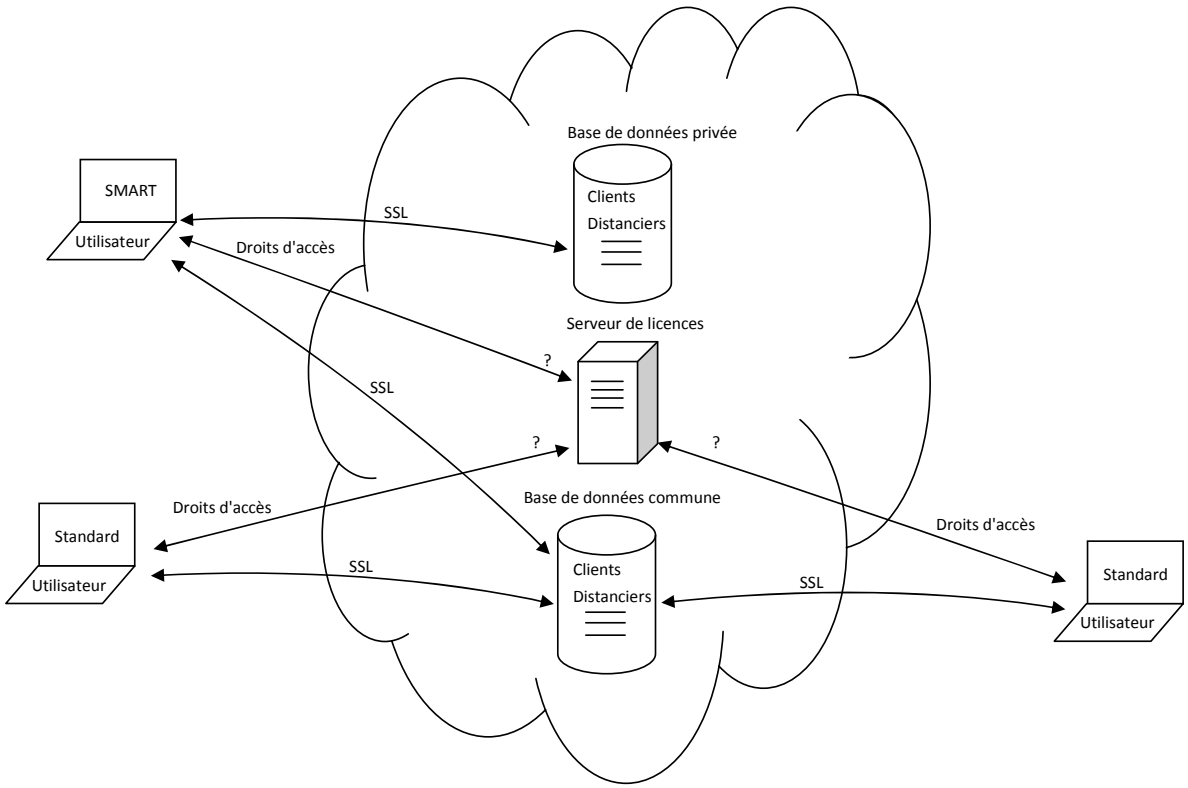


FIGURE *6.6: Architecture du système*

6.3.2 Base de données

Le moteur de base de données utilisé ici est Microsoft SQL Server 2012, il est installé sur le serveur dédié à cet effet. La base de données contient toutes les informations relatives aux points de livraisons nécessaires au planificateur et au logiciel. Ces données sont importées dans la base de données par l'utilisateur. L'importation de nouveaux clients peut se faire de deux manières : la première consiste à les renseigner un à un manuellement. Une seconde permet comme le montre la figure 6.7 d'ajouter rapidement un grand nombre de clients au moyen d'un fichier au format csv (fichier dans lequel les données sont séparées par le caractère " ;") contenant les informations nécessaires que l'utilisateur extrait de son outil métier. L'interface qui a été développée permet d'établir la correspondance entre les colonnes du fichier et les champs de la base de donnée à partir d'un aperçu. La coexistence de ces deux procédures rend l'opération d'ajout de clients dans la base particulièrement flexible, évitant le passage par un fichier auxiliaire pour un faible nombre de clients et une saisie fastidieuse dans le cas contraire.

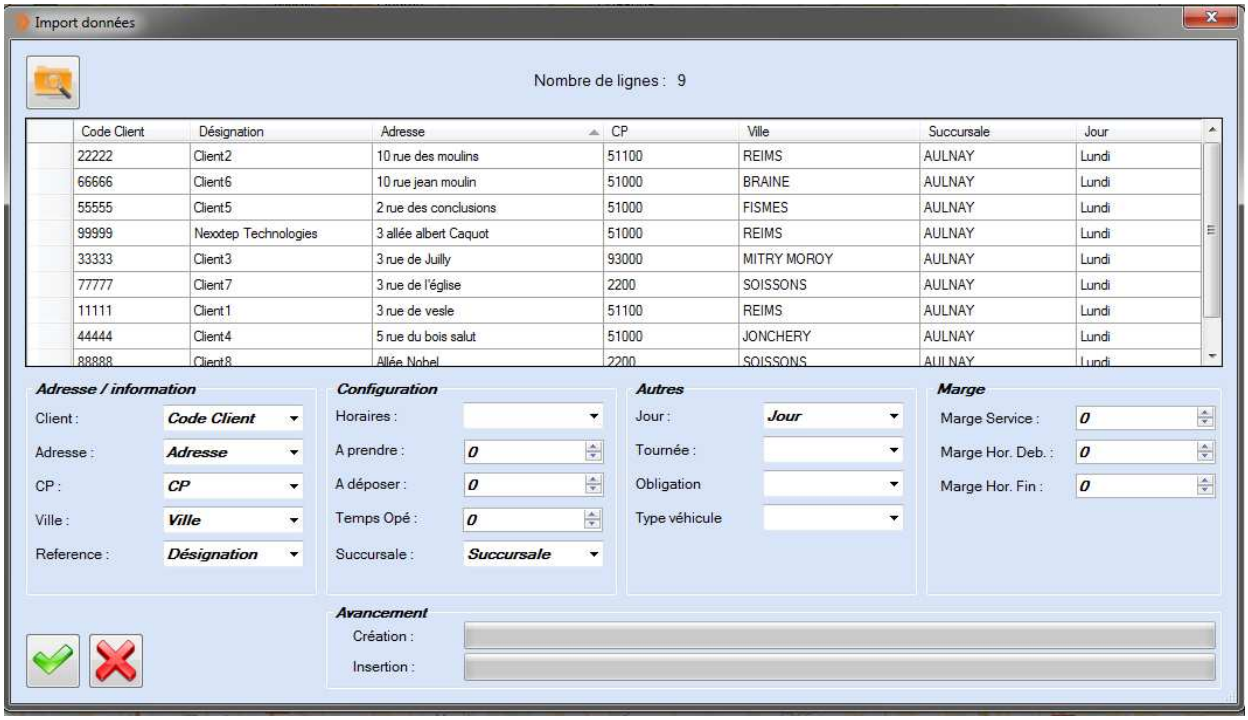


FIGURE *6.7: Import d'une liste de clients à partir d'un fichier*

Une donnée indispensable au placement d'un client sur la cartographie, qui est contenue dans la base de donnée mais n'est pas renseigné par l'utilisateur, est sa position GPS. Cette

position est connue grâce au géocodage. Le géocodage est l'opération qui consiste à faire correspondre à une adresse des coordonnées GPS, elle est réalisée en interrogeant un service spécialisé via internet. Une requête est envoyée avec les éléments d'adresse, des coordonnées GPS et une précision sont retournées. Bien que relativement fiable, ce processus nécessite une validation de la part de l'utilisateur. Il arrive en effet que le service de géocodage ne parvienne pas à localiser l'adresse (la plupart du temps parce que mal renseignée) ou ne le fasse pas de manière suffisamment précise. Un menu est consacré à cette opération dans le logiciel et permet à l'utilisateur de confirmer la position des clients sur la cartographie. Comme le montre la figure 6.8, les clients sont positionnés sur le fonds de carte qui ont pu être géocodés apparaissent en vert, les autres en rouge. Il est alors possible d'ajuster manuellement sur la carte le placement des clients pour lequel un géocodage a pu être obtenu (ceux en vert) et d'identifier et corriger les autres.

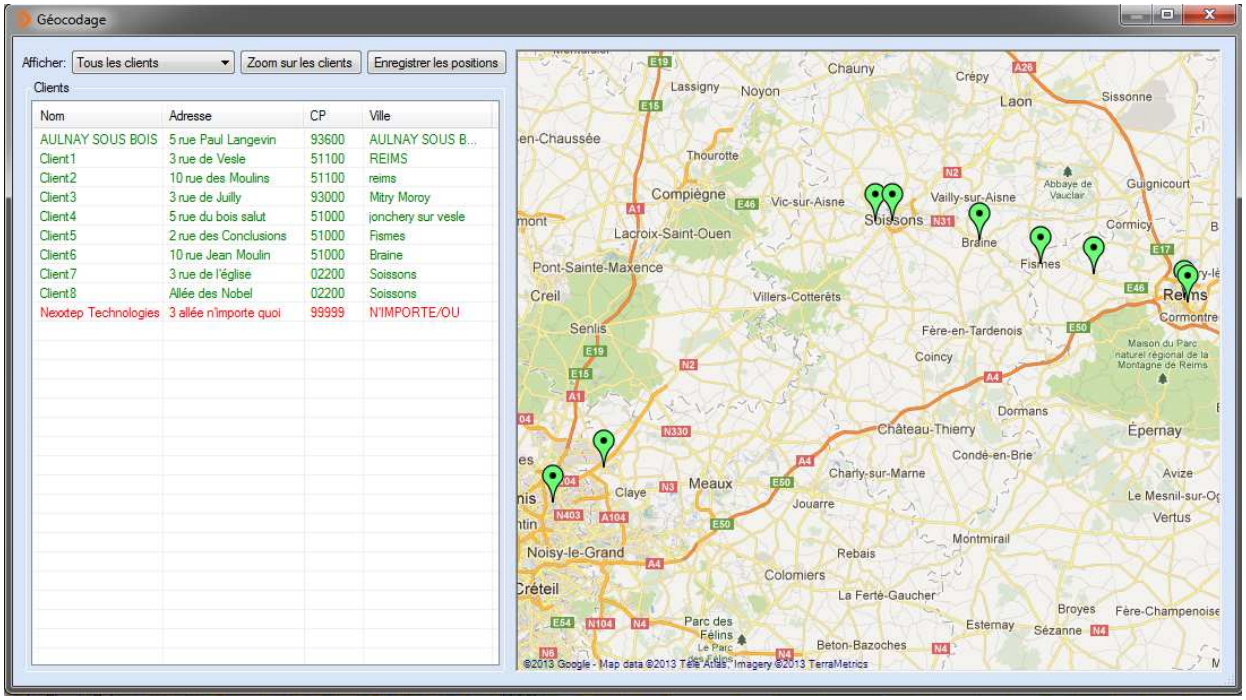


FIGURE *6.8: Géocodage des clients*

Une fois qu'un client est enregistré dans la base de données et que sa position GPS exacte est connue, il peut être affecté à une instance. Une fiche client telle que le montre la figure 6.9 se crée alors et permet de consulter et de modifier les informations qui lui sont relatives. Comme sa position est connue, un aperçu de ses alentours est obtenu à l'aide de Google Street View et permet de prendre connaissance d'éventuelles spécificités d'accès.



FIGURE *6.9: Fiche client*

La base de donnée contient également le distancier qui fournit la distance et le temps de parcours entre deux clients donnés. Pour connaitre ces informations, il faut interroger un service web. Une requête contenant les coordonnées GPS de la paire d'adresses concernée est envoyée avec différentes options (p. ex. le type de véhicule, la limitation de vitesse à considérer, etc ...) est envoyée et les informations relatives au trajet (distance, temps de parcours, points de passages, etc ...) sont renvoyées. Le distancier n'est pas calculé systématiquement pour toutes les paires de clients de la base car le temps de traitement d' une requête est important. Certains clients ne seront jamais associés dans une même instance (jours de livraison différents, succursales de rattachement différentes ...), il est inutile de calculer les informations relatives au trajet qui les sépare. Le distancier se construit au fur et à mesure des utilisations du logiciel. A chaque fois qu'une instance doit être optimisée, une vérification est faite pour savoir si le distancier correspondant est complet. Si ce n'est pas le cas, la base de données est complétée avec les distances et de temps de trajet manquants. Ces données sont obtenues par le biais de requêtes au service web Google Maps API Web Services, elles sont ensuite disponibles pour de prochaines optimisations. Cette procédure rend l'utilisation du logiciel plus agréable et plus fluide pour l'utilisateur, le temps d'attente dévolu au calcul du distancier est réduit au strict nécessaire.

6.3.3 L'interface utilisateur

L'interface utilisateur est un élément important du logiciel, elle se doit d'être simple, ergonomique et fluide à l'utilisation. Un élément essentiel de cette interface pour un logiciel d'optimisation de tournées de véhicules est la cartographie. C'est la principale interface entre le planificateur et le logiciel. Sa lecture doit être simple, la navigation au sein de celle-ci doit être intuitive et fluide. Cette cartographie peut être fournie par un logiciel tiers installée sur le poste de travail ou bien via internet. La deuxième solution qui a finalement été retenue et implémentée chez Nexxtep Technologies, nécessite une connexion internet permanente mais simplifie grandement les procédures d'installation, de maintenance et de mise à jour. En plus de la cartographie, les menus de l'application doivent être clairs et en nombre limité, assurant ainsi une bonne lisibilité à l'utilisateur en évitant l'effet "usine à gaz". La navigation au sein de ces menus doit être intuitive et rapide. La nouvelle version a été développée afin de permettre un affichage rapide des points sur la carte et une meilleure ergonomie. Une vue principale de cette version est donnée par la figure 6.10.

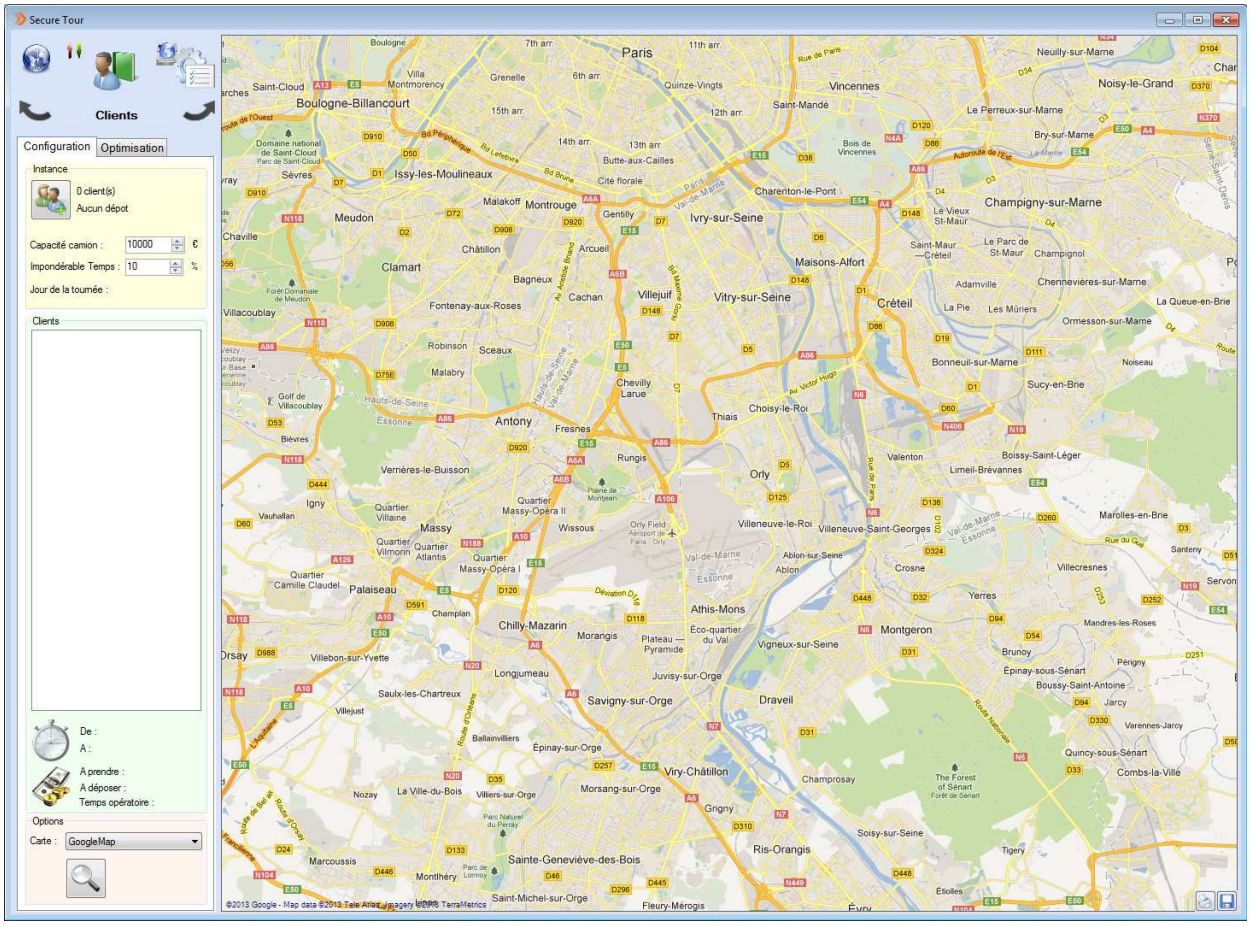


FIGURE *6.10: Vue principale du logiciel*

Une instance est construite grâce au menu de la figure 6.11. Les clients sont sélectionnés puis ajoutés ou supprimés de l'instance en construction à l'aide des flèches. Il est alors possible de définir l'espacement minimum requis entre différentes visites chez un client. Il est également possible d'enregistrer une instance et de la nommer (par exemple "instance du Lundi") pour ne pas avoir à la recréer.

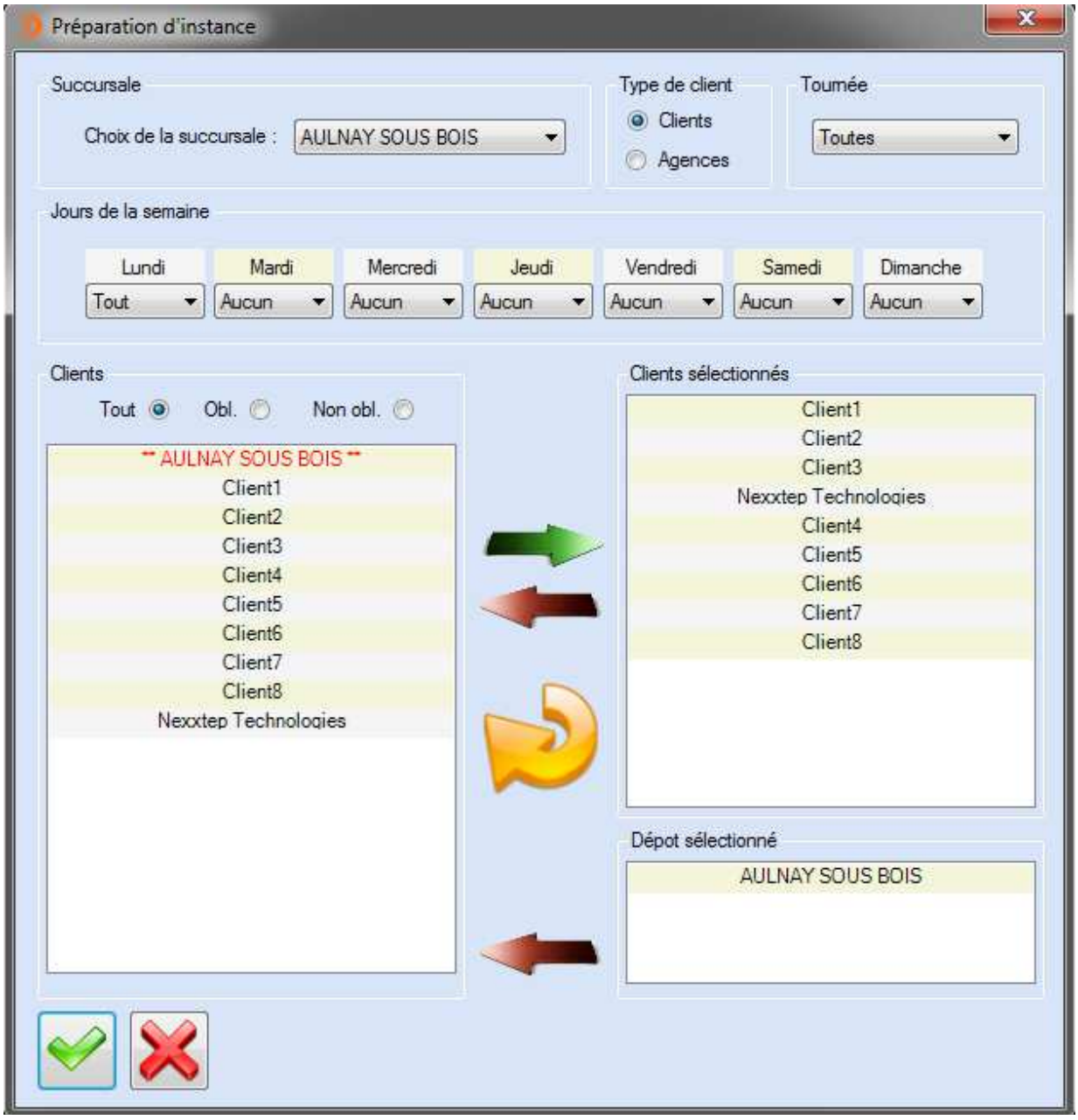


FIGURE *6.11: Création d'une instance*

Une fois l'instance créée, les clients sont affichés sur la carte comme le montre la Figure 6.12 et l'optimisation peut commencer.

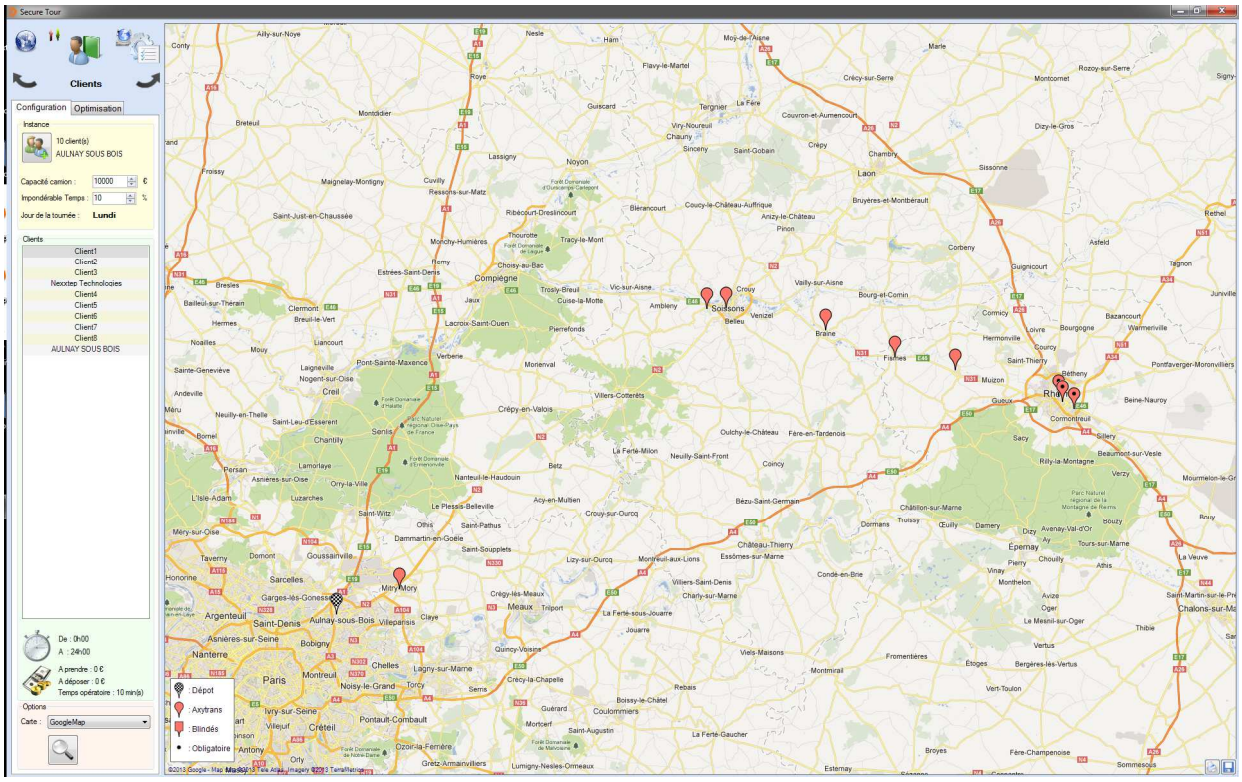


FIGURE *6.12: Instance à réaliser*

Avant le lancement des calculs d'optimisation, des vérifications sont effectuées. La capacité de la flotte de véhicule doit permettre de livrer les clients. Les clients considérés doivent tous être correctement géocodés. Le distancier qui contient la distance et le temps de trajet qui séparent chaque paire de client de l'instance doit être à jour. Enfin, dans le cadre de tournées journalières, les fenêtres horaires des clients doivent permettre de les atteindre depuis le dépôt. Si une de ces vérification fait apparaitre un problème, l'utilisateur en est informé comme le montre la figure 6.13. Sur cet exemple le distancier n'est pas à jour pour traiter l'instance considérée, il doit être complété.

Une fois l'optimisation effectuée le résultat est affiché sur la cartographie comme le montre la figure 6.14. Une feuille de route récapitulant les heures de passage, les quantités délivrées et autres informations pratiques peut être éditée au format pdf ou Excel.

Il est ensuite possible de générer un ensemble de solutions respectant un degré d'irrégularité temporel défini. Comme le montre la figure 6.15, ces solutions sont affichées simultanément sur des cartographies distinctes permettant ainsi de bien visualiser les différences de parcours.

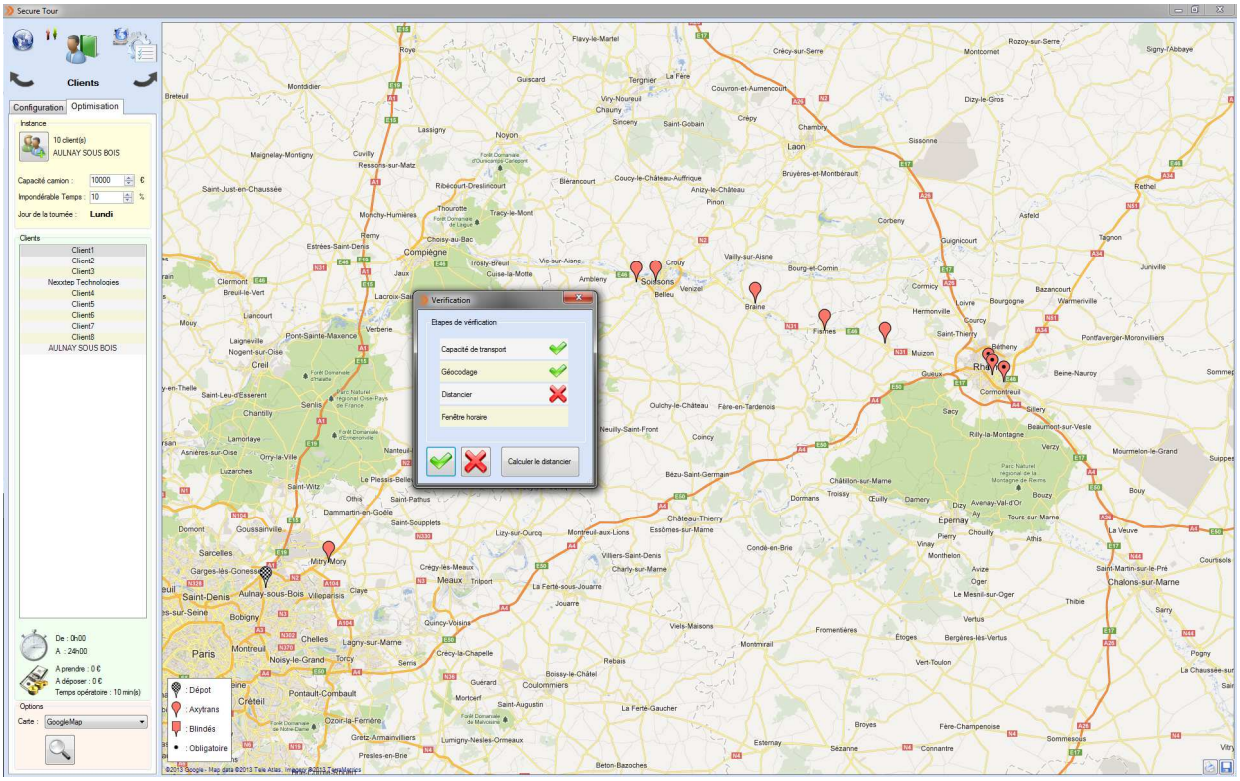


FIGURE *6.13: Vérifications préalables*

6.4 Synthèse

Dans cette section, les développements informatiques qui ont mené au logiciel que commercialise actuellement Nexxtep Technologies sont décrits. Il est expliqué comment, à partir du premier simulateur, une solution commercialisable à vue le jour. L'importance d'une architecture et d'une ergonomie adéquate y est soulignée. Aussi puissant qu'il soit, un moteur de calcul n'est pas d'une grande utilité au client final si il n'est pas servi par une interface utilisateur appropriée.

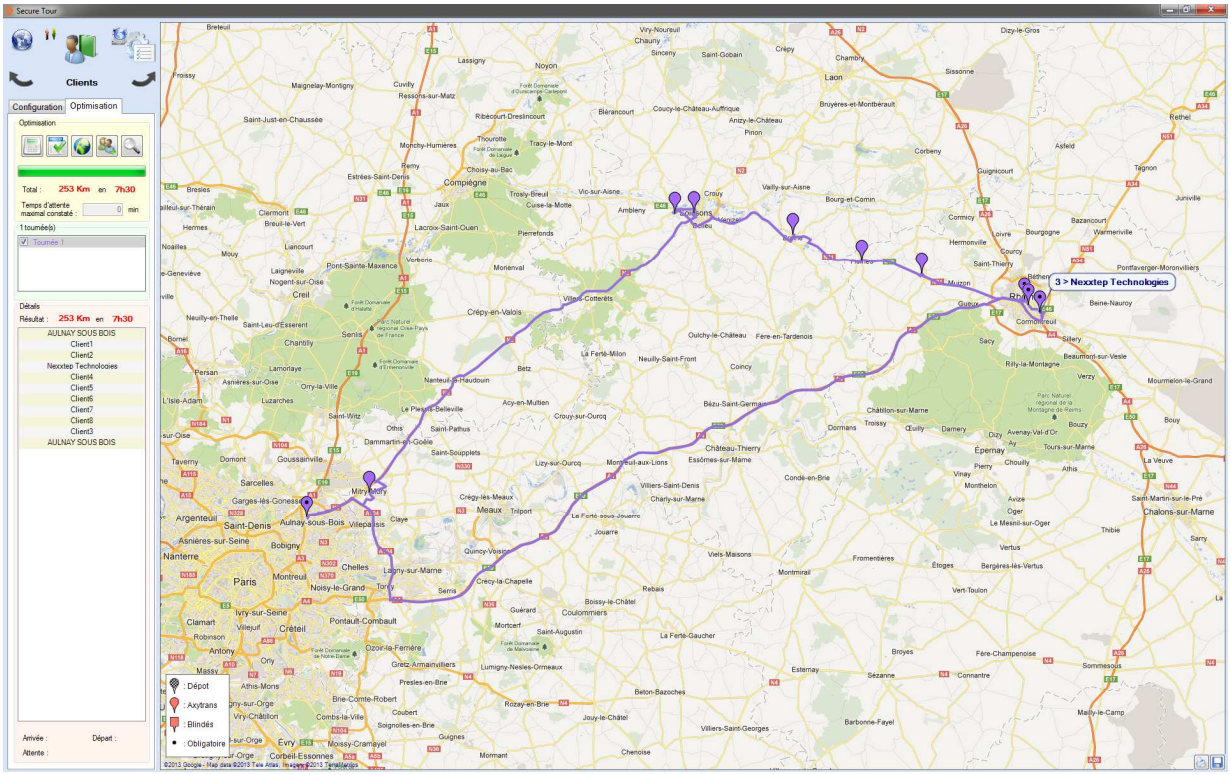


FIGURE *6.14: Tournée optimisée*

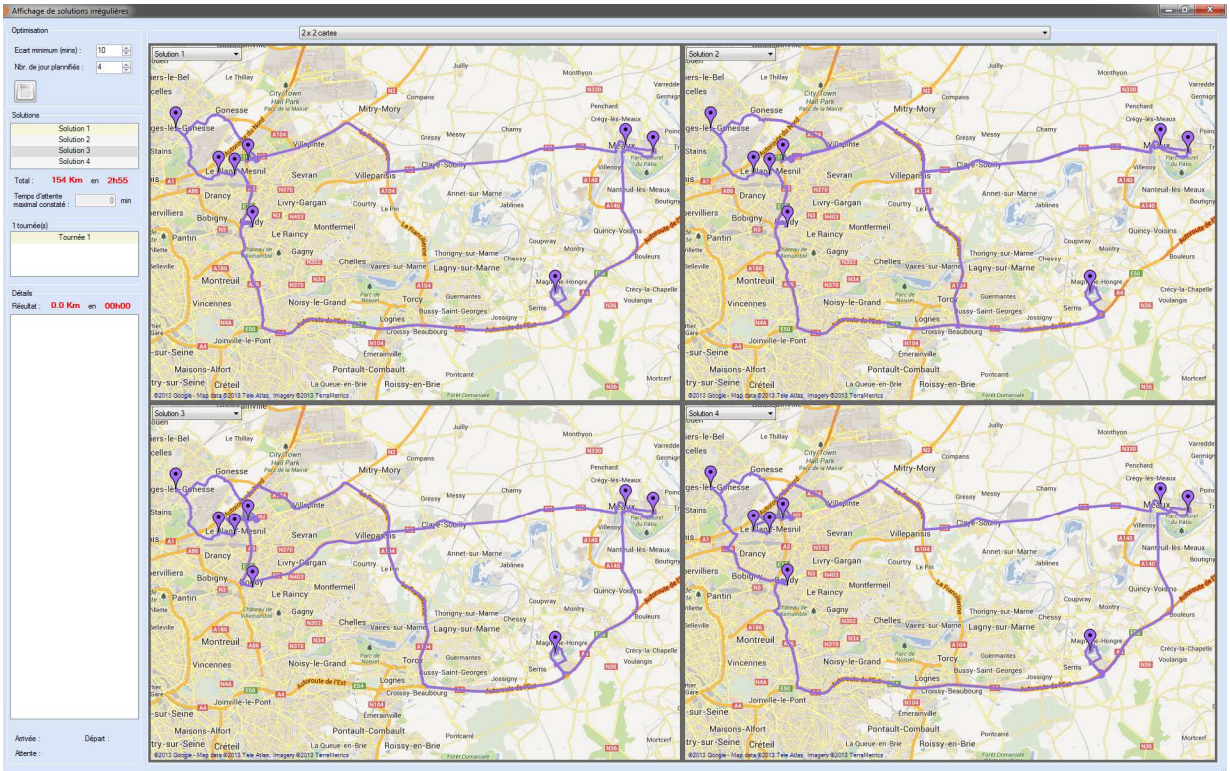


FIGURE *6.15: Jeu de tournées irrégulières*

Conclusion générale

Le premier chapitre de cette thèse présente un problème industriel de tournées de véhicules périodiques appliqué à l'activité de transport de biens de valeur. Grâce à l'état de l'art des problèmes de tournées sur nœuds dressé par la suite, une modélisation mathématique en est donnée dans le chapitre 3. Cette formulation permet de le soumettre à un solveur commercial. Les expérimentations menées avec CPLEX montrent malheureusement que seules des instances de tailles très limitées peuvent être résolues ainsi. Le problème proposé généralise en effet le VRP, avec notamment des contraintes d'espacement des visites chez les clients qui engendrent une forte interdépendance des routes. Le sous-problème de détermination des instants de départ associés se révèle être NP-Complet, augmentant encore la complexité initiale. Le traitement d'instances de tailles "pratiques", contenant plus d'une centaine de clients sur plusieurs jours, dans un contexte opérationnel, ne peut alors être envisagé que de manière approchée. Deux heuristiques constructives sont tout d'abord proposées et évaluées grâce à CPLEX sur de petites instances. Si elles ont l'avantage d'être rapides, ces heuristiques offrent des performances limitées.

Le chapitre 4 propose une métaheuristique MS-ILS efficace pour ce problème. Le cœur de la méthode est une procédure de recherche locale utilisant des fonctions de coût linéaires par morceaux. Ces fonctions permettent la prise en compte efficace de l'interdépendance des heures de services chez les clients. Les tests numériques réalisés montrent qu'elle retrouve quasi systématiquement les solutions optimales des petites instances. Elle surpasse également les heuristiques constructives sur des instances de plus grandes taille issues de la littérature et de cas pratiques. Un problème mono-période plus classique est également abordé et la méthode MS-ILS se révèle compétitive avec deux métaheuristiques de la littérature, produisant de nouvelles meilleures solutions.

Les étapes de conception et de développement de la solution logiciel proposé par Nexxtep Technologies pour l'optimisation de tournées de véhicules sont décrites dans le chapitre 6.

Un problème de tournées de véhicules avec des contraintes de régularité de service est abordé dans le chapitre 5. Dans ce problème, à l'instar du *PVRPTS*, les heures d'arrivées sont éga-

lement interdépendantes et un sous-problème de détermination des horaires de début des routes, bien que polynomial, se pose. Ici encore, la procédure de recherche locale utilisant les fonctions de coût linéaires par morceaux montre son efficacité et sa flexibilité. L'algorithme proposé s'avère en effet compétitif avec les méthodes précédemment publiées dans la littérature et permet d'aborder la version du problème avec instants de départ des routes ajustables par une simple modification de la fonction de coût associée au dépôt.

Les travaux de recherche menés dans le cadre de cette thèse ont donné lieu à deux publications dans des revues internationales, *Computers and Operations Research* et *Expert Systems with Applications*. Des présentations ont également été faites à l'occasion de cinq conférences internationales et trois conférences nationales.

De nombreuses perspectives sont ouvertes. Elles sont méthodologiques notamment, avec le développement de méthodes exactes efficaces pour le *PVRPTS* afin de pouvoir résoudre des instances de plus grande taille.

Du coté de la modélisation, un niveau d'irrégularité global prenant en compte à la fois les itinéraires empruntés et les horaires de livraison peut être défini pour le transport sécurisé. Dans la réalité, il existe plusieurs trajets entre deux points de livraison, ayant chacun leur longueur. Le *PVRPTS* peut donc aussi se modéliser dans un multigraphe contenant plus de deux arcs entre chaque paire de sommets. Un nouveau paradigme mérite également d'être exploré pour le convoi de fonds, dans lequel le client délègue entièrement la gestion de la disponibilité de ses fonds à l'entreprise de convoyage. Le problème est alors une variante de l'IRP dans laquelle une irrégularité sur les jours et les horaires de livraison peut par exemple être introduite. Les versions bi-objectif coût versus niveau de sécurité du *PVRPTS* et coût versus qualité de service du *ConVRP* sont aussi à considérer.

Du coté des applications, la méthode de résolution MS-ILS utilisée, et plus particulièrement la procédure de recherche locale avec fonctions de coût linéaires par morceaux s'avère intéressante pour des problèmes où la composante temporelle est forte et génère des interdépendances. Citons les problèmes de tournées multi-échelons que l'on retrouve en logistique urbaine et dans lesquels des contraintes de synchronisation notamment temporelle peuvent être introduites.

En plus du travail de recherche, une importante part de cette thèse a été consacrée aux activités de développement logiciel chez Nexxtep Technologies. A cette occasion, de nombreuses rencontres clients sont venues nourrir la réflexion menée avec l'indispensable "réalité terrain".

La pertinence de des travaux menés a récemment été confirmée par la parution du décret n° 2012-1109 du 1er octobre 2012 relatif à la protection des transports de fonds. Ce décret

modifie le décret n° 2000-376 du 28 avril 2000 relatif à la protection des transports de fonds en y insérant notamment l'article 2-1 dont voici un extrait :

*"**Art.2-1.- I.-** Les circuits des véhicules de transport de fonds sont préparés par les entreprises de transport de fonds de façon à assurer le départ d'un lieu sécurisé et la variation des itinéraires. Pour les transports desservant les succursales de la Banque de France, une convention conclue entre celle-ci et l'entreprise de transport de fonds précise cette obligation."*

La variation des itinéraires de livraison y est clairement posée comme une obligation. A ce jour, le logiciel développé chez Nexxtep Technologies est le seul à pouvoir répondre à cette contrainte.

Bibliographie

[1] http ://data.worldbank.org/indicator/FB.ATM.TOTL.P5.

[2] A. Ageev, A. Baburin, and E. Gimadi. A 3/4-approximation algorithm for finding two disjoint Hamiltonian cycles of maximum weight. *Journal of Applied and Industrial Mathematics*, 1(2) :142–147, 2007.

[3] A. Ageev and A. Pyatkin. A 2-Approximation Algorithm for the Metric 2-Peripatetic Salesman Problem. In C. Kaklamanis and M. Skutella, editors, *Approximation and Online Algorithms*, volume 4927 of *Lecture Notes in Computer Science*, pages 103–115. Springer Berlin / Heidelberg, 2008.

[4] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58(1-3) :295–324, 1993.

[5] R. Baldacci, E. Bartolini, A. Mingozzi, and A. Valletta. An Exact Algorithm for the Period Routing Problem. *Operations Research*, 59(1) :228–241, 2011.

[6] J. Beasley. Route First-Cluster Second Methods for Vehicle Routing. *Omega*, 11(4) :403–408, 1983.

[7] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[8] F. Belmecheri, C. Prins, F. Yalaoui, and L. Amodeo. An ant colony optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. *Information Control Problems in Manufacturing*, 13(1) :1550–1555, 2009.

[9] E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1) :65–94, 1974.

[10] N. Bianchessi and G. Righini. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34(2) :578–594, Feb. 2007.

[11] O. Bräysy. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4) :1–37, 2003.

[12] B. Bullnheimer, R. Hartl, and C. Strauss. An improved ant System algorithm for thevehicle Routing Problem. *Annals of Operations Research*, 89 :319–328, 1999.

[13] V. Cacchiani, V. Hemmelmayr, and F. Tricoire. A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics*, pages 1–12, Sept. 2012.

[14] A. Campbell, L. Clarke, A. J. Kleywegt, and M. W. P. Savelsbergh. The inventory routing problem. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 95–113. Springer, Boston, 1998.

[15] CECEI. Raport annuel Exercice 2007. Technical report, 2007.

[16] N. Christofides and S. Eilon. An Algorithm for the Vehicle-Dispatching Problem. *Operational Research Quarterly*, 20(3) :309–318, 1969.

[17] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In C. Sandi, editor, *Combinatorial optimization*, pages 315–338. Wiley, New York, 1979.

[18] G. Clarke and J. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4) :568–581, 1964.

[19] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. VRP with Time Windows. In P. Toth and D. Vigo, editors, *The vehicle routing problem*, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[20] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2) :105–119, Sept. 1997.

[21] G. Dantzig and J. Ramser. The truck dispatching problem. *Management Science*, 6(1) :80–91, 1959.

[22] J. B. J. M. De Kort. Lower bounds for symmetric K-peripatetic salesman problems. *Optimization*, 22(1) :113–122, 1991.

[23] J. B. J. M. De Kort. Bounds for the symmetric 2-peripatetic salesman problem. *Optimization*, 23(4) :357–367, 1992.

[24] J. B. J. M. De Kort. A branch and bound algorithm for symmetric 2-peripatetic salesman problems. *European Journal of Operational Research*, 70(2) :229–243, 1993.

[25] R. Dechter, I. Meiri, and J. Pearl. Dechter et al. temporal constraint networks.pdf. *Artificial Intelligence*, 49 :61–95, 1991.

[26] I. Deif and L. Bodin. Extension of the Clark and Wright Algorithm for the Vehicle Routing Problem With Backhauling. In *Proceedings of the Babson Conference on Software Uses in Transportation and Logistics Management*, pages 75–96, 1984.

[27] M. Desrochers and J. Desrosiers. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2) :342–354, 1992.

[28] J. Dethloff. Vehicle routing and reverse logistics : The vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum*, 23(1) :79–96, Feb. 2001.

[29] A. Donati, R. Montemanni, N. Casagrande, A. Rizzoli, and L. Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, 185(3) :1174–1191, Mar. 2008.

[30] M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem. *Bio Systems*, 43(2) :73–81, Jan. 1997.

[31] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system : optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1) :29–41, Jan. 1996.

[32] E. Duchenne, G. Laporte, and F. Semet. Branch-and-cut algorithms for the undirected m-Peripatetic Salesman Problem. *European Journal of Operational Research*, 162(3) :700–712, 2005.

[33] E. Duchenne, G. Laporte, and F. Semet. The Undirected m-Peripatetic Salesman Problem : Polyhedral Results and New Algorithms. *Operations Research*, 55(5) :949–965, Sept. 2007.

[34] E. Duchenne, G. Laporte, and F. Semet. Heuristiques pour le Problème du Vendeur m-Péripatétique. *RAIRO - Operations Research*, 43(01) :13–26, 2009.

[35] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints : Application to some vehicle routing problems. *Networks*, 44(3) :216–229, Oct. 2004.

[36] D. Feillet, T. Garaix, F. Lehuédé, O. Péton, and D. Quadri. Une heuristique pour le problème de tournées de véhicules régulières. In *Conférence ROADEF 2010*, pages 3–4, Toulouse, 2010.

[37] T. Feo and M. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8 :67–71, 1989.

[38] T. A. Feo and M. G. C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6(2) :109–133, 1995.

[39] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2) :109–124, 1981.

[40] P. Francis, K. Smilowitz, and M. Tzur. The Period Vehicle Routing Problem with Service Choice. *Transportation Science*, 40(4) :439–454, Nov. 2006.

[41] P. Francis, K. Smilowitz, and M. Tzur. The Period Vehicle Routing Problem and its Extensions. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem : Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 73–102. Springer US, 2008.

[42] Z. Fu, R. Eglese, and L. Y. O. Li. A unified tabu search algorithm for vehicle routing problems with soft time windows. *Journal of the Operational Research Society*, 59(5) :663–673, Feb. 2007.

[43] Y. Gajpal and P. Abad. An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, 36(12) :3215–3223, Dec. 2009.

[44] M. R. Garey and D. S. Johnson. *Computers and Intractability ; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[45] H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99*, pages 57–64, 1999.

[46] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10) :1276–1290, 1994.

[47] A. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2 :82–114, 1974.

[48] B. Gillett and L. Miller. A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations Research*, 22(2) :340–349, 1974.

[49] E. Gimadi, Y. Glazkov, and A. Glebov. Approximation algorithms for solving the 2-peripatetic salesman problem on a complete graph with edge weights 1 and 2. *Journal of Applied and Industrial Mathematics*, 3(1) :46–60, Mar. 2009.

[50] F. Glover. Tabu Search-Part I. *ORSA Journal on Computing*, 1(3) :190–206, Jan. 1989.

[51] F. Glover. Tabu Search-Part II. *ORSA Journal on computing*, 2(I) :4–32, 1990.

[52] F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65(1-3) :223–253, Mar. 1996.

[53] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, 1997.

[54] M. Goetschalckx and C. Jacob-Blecha. The Vehicle Routing Problem with Backhauls. *European Journal of Operational Research*, 42(1) :39–51, 1989.

[55] B. Golden, E. Baker, J. Alfaro, and J. Schaffer. The Vehicle Routing Problem With Backhauling : Two Approaches. 1985.

[56] B. Golden, S. Raghavan, and E. Wasil. *The vehicle routing problem, latest advances and new challenges*. Springer, New York, 2008.

[57] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5) :275–278, 1958.

[58] C. Groer, B. Golden, and E. Wasil. The Consistent Vehicle Routing Problem. *Manufacturing & Service Operations Management*, 11(4) :630–643, Dec. 2008.

[59] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32(6) :1195–1220, 1984.

[60] H. Hashimoto, M. Yagiura, and T. Ibaraki. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2) :434–456, May 2008.

[61] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3) :791–802, June 2009.

[62] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[63] T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno, and M. Yagiura. Effective Local Search Algorithms for Routing and Scheduling Problems with General Time-Window Constraints. *Transportation Science*, 39(2) :206–232, May 2005.

[64] T. Ibaraki, S. Imahori, K. Nonobe, K. Sobue, T. Uno, and M. Yagiura. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics*, 156(11) :2050–2069, 2008.

[65] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :671–80, May 1983.

[66] K. Knight and J. Hofer. Vehicle Scheduling with Timed and Connected Calls : A Case Study. *Operation Research Quarterly*, 19(3) :299–310, 1968.

[67] Y. A. Koskosidis, W. B. Powell, and M. M. Solomon. An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints. *Transportation Science*, 26(2) :69–85, May 1992.

[68] A. A. Kovacs, S. N. Parragh, and R. F. Hartl. A template based adaptive large neighborhood search for the consistent vehicle routing problem, 2011.

[69] J. Krarup. The peripatetic salesman and some related unsolved problems. In B. Roy, editor, *Combinatorial programming : methods and applications*, pages 173–178. Reidel, Dordrecht, 1975.

[70] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3) :497–520, 1960.

[71] F. Li, B. Golden, and E. Wasil. Very large-scale vehicle routing : new test problems, algorithms, and results. *Computers & Operations Research*, 32(5) :1165–1179, May 2005.

[72] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44 :2245–2269, 1965.

[73] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2) :498–516, 1973.

[74] S. Lin and B. W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2) :pp. 498–516, 1973.

[75] L. Lindner-Dutton, R. Batta, and M. Karwan. Equitable sequencing of a given set of hazardous materials shipments. *Transportation Science*, 2(25) :124–137, 1991.

[76] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated Local Search : Framework and Applications. In M. Gendreau and J.-Y. Potvin, editors, *Book Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, chapter 12, pages 363–397. Springer, 2010.

[77] R. Masson, F. Lehuédé, and O. Péton. Simple Temporal Problems in Route Scheduling for the Dial a Ride Problem with Transfers. In N. Beldiceanu, N. Jussien, and E. Pinson, editors, *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*, volume 7298, pages 275–291. Springer Berlin Heidelberg, lncs 7298 edition, 2012.

[78] R. Masson, T. Vidal, J. Michallet, P. H. V. Penna, V. Petrucci, A. Subramanian, and H. Dubedout. An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. *Expert Systems with Applications*, 40(13) :5266–5275, Mar. 2013.

[79] J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vitry. A periodic vehicle routing problem with time windows and time spread constraints on services. In *MIC 2011 : The IX Metaheuristics International Conference*, page 10p, 2011.

[80] J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vitry. Vehicle Routing Problem with Overlap constraints. In *Proceedings of International Conference on Industrial Engineering and Systems Management*, pages 1311–1320, Metz, France, 2011.

[81] J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vitry. Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Computers & Operations Research*, 41 :196–207, 2014.

[82] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7 :326–329, 1960.

[83] H. Min. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A : General*, 23(5) :377–386, 1989.

[84] L. Mingyong and C. Erbao. An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence*, 23(2) :188–195, Mar. 2010.

[85] R. H. Mole and S. R. Jameson. A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion. *Operational Research Quarterly (1970-1977)*, 27(2) :pp. 503–511, 1976.

[86] F. A. T. Montané and R. D. Galvão. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers &amp ; Operations Research*, 33(3) :595–619, 2006.

[87] G. Nagy. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162(1) :126–141, Apr. 2005.

[88] S. Ngueveu. *Problèmes de tournées de véhicules avec contraintes particulières pour la maîtrise des risques*. PhD thesis, Université de Technologies de Troyes, 2009.

[89] S. U. Ngueveu, C. Prins, and R. Wolfler-Calvo. A Hybrid Tabu Search for the m-Peripatetic Vehicle Routing Problem. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Matheuristics*, volume 10 of *Annals of Information Systems*, pages 253–266. Springer US, 2010.

[90] S. U. Ngueveu, C. Prins, and R. Wolfler-Calvo. Lower and upper bounds for the m-peripatetic vehicle routing problem. *4OR*, 8(4) :387–406, 2010.

[91] I. Or. *Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking*. PhD thesis, Northwestern University, Evanston, 1976.

[92] M. Padberg and M. Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1) :67–80, 1982.

[93] M. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, 6(1) :1–7, 1987.

[94] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1) :21–51, Mar. 2008.

[95] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2) :81–117, May 2008.

[96] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8) :2403–2435, Aug. 2007.

[97] J. Potvin and J. Rousseau. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46(12) :1433–1446, 1995.

[98] J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3) :331–340, 1993.

[99] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12) :1985–2002, Oct. 2004.

[100] C. Prins. A GRASP X Evolutionary Local Search Hybrid for the Vehicle Routing Problem. *Bio-inspired algorithms for the vehicle routing problem*, 161 :35–53, 2009.

[101] C. Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22(6) :916–928, Sept. 2009.

[102] C. Prins, N. Labadi, and M. Reghioui. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47(2) :507–535, Jan. 2009.

[103] H. Pullen and M. Webb. A computer application to a transport scheduling problem. *The Computer Journal*, 10 :10–13, 1967.

[104] C. Rego. Node-ejection chains for the vehicle routing problem : Sequential and parallel algorithms. *Parallel Computing*, 27(3) :201–222, 2001.

[105] M. Reimann, K. Doerner, and R. Hartl. Analyzing a unified Ant System for the VRP and some of its variants. *Applications of Evolutionary Computing, LNCS*, 2611 :300–310, 2003.

[106] M. Reimann, K. Doerner, and R. F. Hartl. D-Ants : Savings Based Ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4) :563–591, Apr. 2004.

[107] G. Reinelt. TSPLIB–A Traveling Salesman Problem Library. *INFORMS Journal on Computing*, 3(4) :376–384, 1991.

[108] J. Rieck and J. Zimmermann. A sampling procedure for real-life rich vehicle routing problems. In K. H. Waldmann and U. H. Stocker, editors, *Operations Research Proceedings 2006*, pages 355–360. Springer, Berlin, 2007.

[109] J. Roskind and R. E. Tarjan. A Note on Finding Minimum-Cost Edge-Disjoint Spanning Trees. *Mathematics of Operations Research*, 10(4) :701–708, 1985.

[110] G. Ross and R. Soland. A branch and bound algorithm for the generalized assignment problem. *Mathematical programming*, 8 :91–103, 1975.

[111] B. Roy and M. Dibon. L'ordonnancement par la méthode des potentiels-Le programme CONCORD. *Automatisme*, (2) :1–11, 1966.

[112] M. Savelsbergh. The vehicle routing problem with time windows : Minimizing route duration. *ORSA Journal on Computing*, 4(2) :146–154, 1991.

[113] M. M. Solomon. Algorithms for the Vehicles Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2) :254–265, 1987.

[114] A. Subramanian, L. Drummond, C. Bentes, L. Ochi, and R. Farias. A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Computers & Operations Research*, 37(11) :1899–1911, Nov. 2010.

[115] C. Tarantilis, F. Stavropoulou, and P. Repoussis. A template-based Tabu Search algorithm for the Consistent Vehicle Routing Problem. *Expert Systems with Applications*, 39(4) :4233–4239, Mar. 2012.

[116] R. E. Tarjan. Shortest-Paths. Technical report, AT&T Bell Laboratories, Murray Hill, 1981.

[117] P. Toth and D. Vigo. *The Vehicle Routing Problem*. Philadelphia, Society for Industrial and Applied Mathematics, 2002.

[118] S. Tse. *L'art de la guerre*. L'impensé Radical, Paris, 1978.

[119] T. Vidal, T. Crainic, M. Gendreau, and C. Prins. A Unifying View on Timing Problems and Algorithms. Technical report, CIRRELT, Montréal, Canada, 2011.

[120] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, 60(3) :611–624, 2012.

[121] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1) :475–489, Jan. 2013.

[122] S. Wolf and P. Merz. Evolutionary local search for the super-peer selection problem and the p-hub median problem. In T. Bartz-Beielstein, M. Blesa Aguilera, C. Blum, B. E Naujoks, A. E Roli, G. E Rudolph, and M. E Sampels, editors, *Hybrid Metaheuristics, LNCS*, volume 4771, pages 1–15. Springer Berlin Heidelberg, 2007.

[123] R. Wolfler-Calvo and R. Cordone. A heuristic approach to the overnight security service problem. *Computers & Operations Research*, 30(9) :1269–1287, Aug. 2003.

[124] B. Yu, Z.-Z. Yang, and B. Yao. An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, 196(1) :171–176, July 2009.

[125] T. Zhang, W.-x. Tian, Y.-j. Zhang, and S.-x. Liu. Improved Ant Colony System for VRPSPD with Maximum Distance Constraint. *Systems Engineering - Theory & Practice*, 28(1) :132–140, Jan. 2008.

Julien MICHALLET

Doctorat : Optimisation et Sûreté des Systèmes

Année 2013

Problèmes de tournées de véhicules périodiques avec contraintes de sécurité ou de qualité de service

Cette thèse aborde le problème de tournées de véhicules périodiques (PVRP) lorsqu'il est appliqué au transport de marchandises convoitables. Des contraintes spécifiques relatives à la sécurité du convoi doivent être définies.
Le problème de tournées de véhicules périodiques avec dispersion des instants de service (PVRPTS) est alors décrit puis modélisé mathématiquement. Le but est de servir un ensemble de clients sur plusieurs jours en respectant un degré de variation défini dans les heures de service. Le modèle obtenu est discuté et deux heuristiques constructives sont proposées et évaluées pour sa résolution.
Une recherche locale itérée avec redémarrages (MS-ILS) est proposée pour ce problème. Les résultats obtenus montrent que cette méthode surpasse les deux précédentes sur toutes les instances de test. Elle est ensuite évaluée sur un problème plus classique de la littérature : le problème de tournées de véhicules avec fenêtres horaires souples (VRPSTW) et s'avère très compétitive, produisant de nouvelles meilleures solutions.
La MS-ILS est ensuite transposée au problème de tournées de véhicules régulières (ConVRP). Contrairement au PVRPTS, il s'agit dans le ConVRP de servir régulièrement des clients aux demandes intermittentes. La méthode montre une flexibilité remarquable et produit de bons résultats.
Pour finir, les développements effectués chez Nexxtep Technologies sont présentés. Ils comprennent la conception d'un logiciel commercial pour l'optimisation de tournées de véhicules et l'implémentation des méthodes développées.

Mots clés : recherche opérationnelle - logistique (organisation) - sécurité - optimisation combinatoire - transport – qualité de service.

Periodic Vehicle Routing Problems with Security Constraints or Quality of Service Requirements

This thesis is dedicated to the periodic vehicle routing problem when applied to the transportation of valuable goods. Specifics constraints have to be defined to ensure the security of the convoy.
The periodic vehicle routing problems with time spread constraints on services (PVRPTS) is defined and a mathematical model is given. The goal is to serve a set of customers over several days such that their visit times differ by a minimum amount. The depicted model is discussed and two constructive heuristics are designed and assessed.
A Multi-start iterated local search (MS-ILS) is proposed to solve this problem. The results shows that the method outperform the two previous heuristics. For the sake of comparison with previous approaches, the MS-ILS is evaluated on a more classical problem : the vehicle routing problem with soft time windows (VRPSTW). The method proves to be very competitive, producing new best known solutions.
The MS-ILS is then adapted to solve the consistent vehicle routing problem (ConVRP). Unlike the PVRPTS, the goal of the ConVRP is to deliver customers who have intermittent demands with regularity in terms of service times and drivers. The method demonstrates his flexibility and produces good results.
Finally, the developments performed at Nexxtep Technologies are depicted. They encompass commercial-software design and implementation of the proposed methods.

Keywords: operations research - business logistics – security - combinatorial optimization – transportation - service-level agreements.

Thèse réalisée en partenariat entre :

NEXXTEP TECHNOLOGIES

utt université de technologie Troyes

Ecole Doctorale "Sciences et Technologies"