HEC MONTRĒAL

Résolution de tournées de véhicules dépendantes du temps avec fenêtre de temps (TDVRPTW) par génération de colonnes.

par

Vincent Huart

Sciences de la gestion

(Méthodes analytiques de gestion)

Mémoire présenté en vue de l'obtention

du grade de maîtrise ès sciences

(M. Sc.)

Mai, 2013

© Vincent Huart , 2013

Résumé

Le problème de tournées de véhicules est un sujet très étudié en recherche opérationnelle avec de multiples applications. Il est basé sur un graphe où les nœuds, représentant les clients, ont une demande, et où les arêtes, représentant les routes, ont un coût (ex : distance). L'objectif est de traiter tous les clients en partant et revenant à un nœud appelé dépôt tout en minimisant les coûts.
Dans ce mémoire, nous ajoutons la dépendance au temps. En effet dans la majorité des modèles, le problème a été simplifié en considérant la vitesse des véhicules comme constante. Pour aborder ce problème de tournées de véhicules dépendantes du temps nous avons fait deux choix : (i)La vitesse dépend directement du type de route emprunté. On différencie ainsi autoroutes et rues. (ii) La vitesse varie au cours de la journée. Pour prendre en compte ces deux, nous avons caractérisé chaque route par un profil de vitesse qui correspond à la variation de la vitesse en fonction des diverses périodes de la journée.
Nous résolvons le problème avec un algorithme parallélisé de génération de colonnes. Les colonnes sont générées heuristiquement à l'aide de différents mouvements et de métaheuristiques de type VND. Nos résultats sont comparés avec les instances de Solomon. Ces instances, connues dans la littérature, prennent en compte des fenêtres de temps sur les clients ainsi que des capacités sur les véhicules. Pour terminer, nous avons modifié ces instances pour les adapter à des problèmes dépendants du temps.

Mots clés : Tournées de véhicules dépendantes du temps(TDVRPTW), Métaheuristiques (VND), Génération de colonnes

Abstract

The Vehicle routing problem is very studied in operational research and have multiple applications. It's based on a graph where nodes represent customers and edges represent roads between these clients. On each edge we assigned a cost which may for instance represent distance. Each customer has a demand to satisfy. Goal is to treat all customers starting and returning to one node called deposit and minimizing the number of vehicles used.

In this master thesis add time-dependency. Indeed, most models simplified the problem considering a constant vehicle speed. To address this routing problem of time dependent vehicles we made two choices: (i) Speed directly depends on the type of road on which the vehicle is traveling. Thus we differentiate highways and streets of downtown. (ii) Speed varies during the day. To take into account both parameters in our model, we characterized each road by a velocity profile. That corresponds to a change in velocity as a function of the various periods of the day.

We solved the problem with a parallelized algorithm using column generation. Columns are generated heuristically using different movements and metaheuristics like VND (Variable Neighborhood, Descent). To test our model, we began by using a constant speed so that we compare the results obtained on the instances of Solomon. These instances, known in the literature, are typical for vehicle routing. It use time windows of the customers as well as capacity on vehicles. Finally, we modified these instances to adapt them to time-dependent problems.

Keywords : Vehicle Routing, Metaheuristic, TDVRPTW, VND, Column Generation

Table des matières

Résumé **iii**

Abstract **iv**

1 Introduction **10**

1.1 Introduction 10

1.2 Structure du mémoire 11

2 Cheminement **12**

2.1 Modélisation 12

2.2 Instances 13

2.2.1 Clients 13

2.2.2 Arcs 14

2.2.3 Scénario 16

2.3 Solution initiale 17

2.4 VND(Variable Neighbhorood Search) 18

2.5 Génération de colonnes 19

2.6 Parallélisation 21

2.7 Développement 22

3 A Column Generation Heuristic for the TDVRPTW **23**

3.1 Introduction 1

3.2 Literature review 3

3.2.1 Studies on the TDVRPTW 3

3.2.2 Studies on the TDSPPTW 5

3.2.3 Column Generation Algorithm for the VRPTW 6
3.3 Mathematical Formulation 6
3.3.1 Mixed Integer Program for the VRPTW 6
3.3.2 Time Dependent modeling 8
3.4 Proposed Solution Method 12
3.4.1 Column Generation 12
3.4.2 Variable Neighborhood Descent (VND) 15
3.4.3 Global design of our heuristic Column generation 19
3.4.4 Settings 22
3.5 Computational results 24
3.5.1 Comparison with VRPTW 24
3.5.2 Results on TDVRPTW 29
3.6 Conclusion 33

4 Résultats **34**
4.1 VRPTW 34
4.1.1 Résultats 25 35
4.1.2 Résultats 50 37
4.1.3 Résultats 100 41
4.2 TDVRPTW 45
4.2.1 Scénarios TDVPRTW 45
4.2.2 C101 avec 25 clients 46
4.2.3 C101 avec 50 clients 49

5 Conclusion **51**
5.1 Résumé 51
5.2 Recherche future 51

Liste des figures

2.1 Approximation de la vitesse réelle 12
2.2 Instance C101 de Solomon avec 25 Clients 15
2.3 Assignation de profils sur l'instance C101 15
2.4 Optimum Local vs Optimum Global 18
2.5 Algorithme de génération de colonnes 21

3.1 Two different scenarios of three profiles with three intervals 9
3.2 Speed evolution on an arc of length 1 as a step function 10
3.3 Travel time between clients i and j 11
3.4 Illustration of internal 2-opt move 16
3.5 Illustration of internal Exchange of size 3 17
3.6 Illustration of a cross with $n_1 = n_2 = 2$ 18
3.7 Global design of our parallel column generation heuristic 23
3.8 Modified instance C101 with 25 customers 31
3.9 Modified instance C101 with 50 customers 32

4.1 Résultats avec 25 clients avec objectif coût 36
4.2 Résultats avec 25 clients avec objectif tournée 38
4.3 Résultats avec 50 clients avec objectif coût 39
4.4 Résultats avec 50 clients avec objectif tournée 40
4.5 Résultats avec 100 clients avec objectif coût 41
4.6 Résultats avec 100 clients avec objectif tournée 42
4.7 Résultats avec 100 clients avec objectif coût, distance réelle 43
4.8 Résultats avec 100 clients avec objectif tournée, distance réelle 44
4.9 Profils sur l'instance C101 à 25 clients 47

4.10 Structures des résultats sur le scénario constant 47
4.11 Structures des résultats sur le scénario 1 48
4.12 Structures des résultats sur le scénario 2 48
4.13 Structures des résultats sur le scénario 3 48
4.14 Structures des résultats sur le scénario 4 49
4.15 Profils sur l'instance C101 à 50 clients 50

Liste des tableaux

2.I Début d'une instance de solomon 13
2.II Assignation des arcs et profils sur les 10 premiers clients 16
2.III Scenario 1 avec 3 profils de vitesses et 7 intervalles 17

3.I Results on Solomon instances with 25 clients 26
3.II Results on Solomon instances with 50 clients 27
3.III Results on Solomon instances with 100 clients 28
3.IV Summary on Solomon instances with 100 clients 29
3.V scenarios for the speed profiles in our TDVRPTW instances 30
3.VI Results on C101 with 25 clients 30
3.VIIResults on C101 with 50 clients 32

4.I Scénarios utilisés sur le TDVRPTW 46
4.II Résultats TDVRPTW C101 avec 25 clients 47
4.III Résultats TDVRPTW C101 avec 50 clients 49

Chapitre 1

Introduction

1.1 Introduction

Comme dans beaucoup de problèmes issus du quotidien, on souhaite minimiser les coûts. Sans parler nécessairement de profit financier, on peut vouloir minimiser les coûts pour avoir une plus grande rapidité de traitement, une meilleure répartition des charges. La recherche opérationnelle et l'optimisation combinatoire nous permettent de répondre à cette problématique.
Le problème classique de tournées de véhicules a été présenté il y a une cinquantaine d'années comme une extension du problème du voyageur de commerce. Dans ce problème initial, il s'agissait de trouver la route de distance minimale reliant un ensemble de villes. Imaginez par exemple que vous partez pour une semaine de vacances en France et que vous souhaitez visiter Paris, Nîmes, Nantes, Clermont-Ferrand ... comment planifier votre trajet de manière à perdre le moins de temps dans les transports ?

Le premier problème de tournée de véhicules a été proposé par Dantzig et al. en 1959 [9] pour traiter les problèmes de livraison d'essence par des camions. Il s'agit de déterminer les tournées, c'est à dire l'affectation des clients, d'une flotte de véhicules de manière à minimiser le coût. Diverses applications existent : opération de maintenance, collecte de déchets, livraison ... Dans le problème de base, on dispose d'une carte représentant les sommets à visiter ainsi que du coût lié à la distance à parcourir, on souhaite

alors minimiser le nombre de véhicules utilisés tout en minimisant la distance parcourue. De nombreuses variantes du problème de tournées ont ensuite vues le jour, ajout de contraintes de capacités sur les véhicules (poids, volume, quantité ...), de contraintes liées aux clients (fenêtre de temps représentant les heures d'ouverture et de fermeture, traitement spécialisé ...), des contraintes affectant la tournée d'un véhicule (ordonnancement des clients, clients de collecte et/ou de livraison) ... Nous allons dans ce projet nous intéresser à une version beaucoup plus réaliste qui prend en compte le fait que la vitesse sur une route n'est pas toujours constante et surtout que les routes ne sont pas toutes identiques. On rajoute aussi le fait qu'un client possède des horaires d'ouverture et de fermeture ; cela impose donc que son traitement soit placé à une certaine position dans la tournée ainsi qu'une capacité sur nos véhicules ce qui les oblige à rentrer au dépôt s'ils sont pleins (ou vides selon le type de problème). Ce problème ainsi modifié est connu dans la littérature sous le nom de TDVRPTW (Time Dependent Vehicule Routing Problem with Time Windows).

1.2 Structure du mémoire

Ce mémoire est un mémoire par article. Il est basé autour de l'article présenté au chapitre 3. Les autres chapitres sont présents pour expliquer la démarche entreprise et apporter une introduction pour une personne étrangère au sujet. Ce rapide premier chapitre introduit le sujet, dans le deuxième chapitre nous présenterons le cheminement et le travail qui a été réalisé dans ce mémoire. Le chapitre 3 contiendra l'article et même si les résultats y sont déjà présentés nous en détaillerons quelques-uns dans le chapitre 4. Enfin, nous conclurons dans le cinquième chapitre.

Chapitre 2

Cheminement

Ce chapitre explique globalement le travail qui a été réalisé dans ce mémoire, l'origine des idées et la mise en application. Nous avons essayé de vulgariser au maximum afin qu'une personne étrangère à la recherche opérationnelle puisse comprendre de quoi il relève. Pour plus de précisions et de détails techniques sur les différents points abordés, se reporter à l'article présenté au chapitre 3.

2.1 Modélisation

Nous avons ajouté des profils de vitesse afin d'imiter la variation de la vitesse réelle. On va approximer la fonction représentant la vitesse réelle par une fonction en escalier avec des intervalles qui correspondront à divers créneaux horaires de la journée :

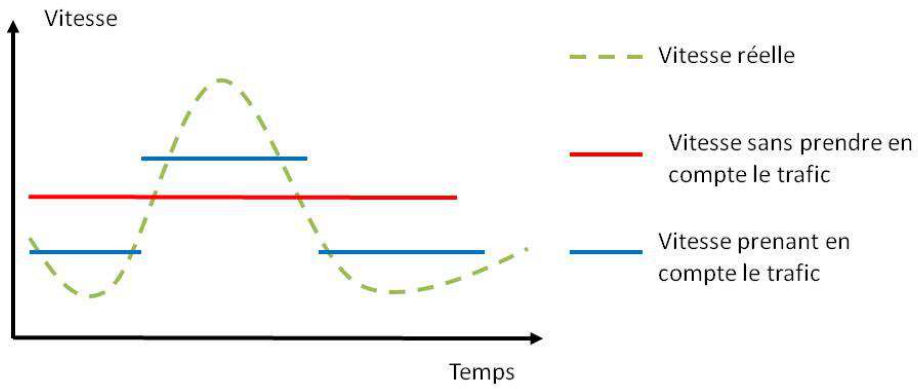

FIGURE 2.1 – Approximation de la vitesse réelle

2.2 Instances

2.2.1 Clients

Nous ne travaillons pas avec des données réelles, nous utilisons les instances de Solomon qui modélisent le VRPTW(tournée de véhicules avec fenêtre de temps). Ces instances générées sont devenues des références dans ce domaine de recherche. Elles sont disponibles publiquement sur internet sous la forme de fichiers textes. Ces fichiers permettent de construire le graphe sur lequel les algorithmes feront les analyses. Le tableau 2.2.1 montre un morceau de la syntaxe d'une instance de Solomon.

C101						
VEHICLE						
NUMBER	CAPACITY					
25	200					
CUSTOMER						
CUST NO.	XCOORD.	YCOORD.	DEMAND	READYTIME	DUEDATE	SERVICE
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90

Table 2.I – Début d'une instance de solomon

La première ligne fait référence au nom de l'instance, c'est-à-dire le nom du fichier. Ensuite, le fichier indique la taille de la flotte de véhicules disponibles pour traiter les requêtes des clients, ainsi que leur capacité. La troisième partie du fichier donne des informations par rapport aux clients à traiter.

- Cust No. : Identificateur du client.
- XCoord. YCoord. : Coordonnées géographiques du client.
- Demand : Requête à traiter. Elle peut indiquer aussi bien la quantité que le client demande ou la quantité qu'il veut livrer (quantité que l'on devrait alors collecter).
- ReadyTime : Aussi appelé le Earliest Starting Time EST, correspond à la date au plus tôt à laquelle un client peut démarrer le traitement de sa requête.
- DueTime : Aussi appelé le Latest Starting Time LST, correspond à la date au plus tard à laquelle un client peut démarrer le traitement de sa requête.

– Service : Temps de traitement du client. Le temps que dépense le véhicule avant de pouvoir repartir pour traiter un autre client.

Par exemple la ligne 2 de la troisième partie du fichier représentera un client avec l'identificateur 1, localisé au pont (45, 68), qui demande 10 unités, et qui a comme fenêtre de temps l'intervalle [912, 967].

Les instances de Solomon sont divisées en trois catégories :

1. Cxxx : instance générée de manière à avoir des clients répartis sous forme de clique.
2. Rxxx : instance générée de manière à avoir des clients placés répartis aléatoirement.
3. RCxxx : instance mêlant clients répartis en clique et aléatoirement.

Pour chacun de ces trois types, on dispose d'instances avec 25,50 et 100 clients. Plus on augmente le nombre de clients plus la complexité devient élevé et donc plus la résolution devient compliquée.

On présente sur la figure 2.2.1 l'instance C101 de Solomon avec 25 clients.

2.2.2 Arcs

Comme on peut le voir sur la figure 2.2.1, le réseau est complet : on peut aller de tout client vers tout client. Ceci n'est pas très réaliste c'est pourquoi nous allons modifier ce réseau en supprimant des arcs et en assignant des profils aux routes. On obtient alors la figure 2.2.2.

Dans cet exemple nous sommes partis de l'instance C101 de Solomon, les clients étaient donc répartis en clique. On pourrait voir leur placement comme trois villes différentes. La numérotation des profils des voies a été choisie pour représenter les petites rues dans le centre-ville (avec le code 1), les boulevards de la ville (avec le code 2) et les périphériques ou autoroutes (avec le code 3). Dans un premier temps, il est possible de considérer que la vitesse est croissante selon le type de voie.

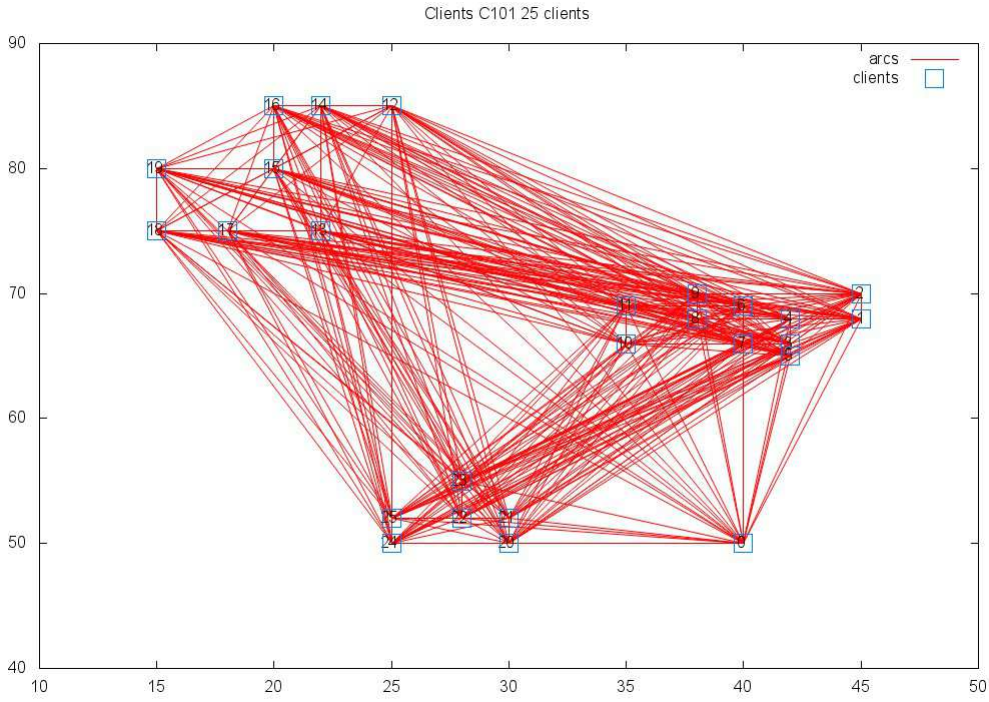

FIGURE 2.2 – Instance C101 de Solomon avec 25 Clients

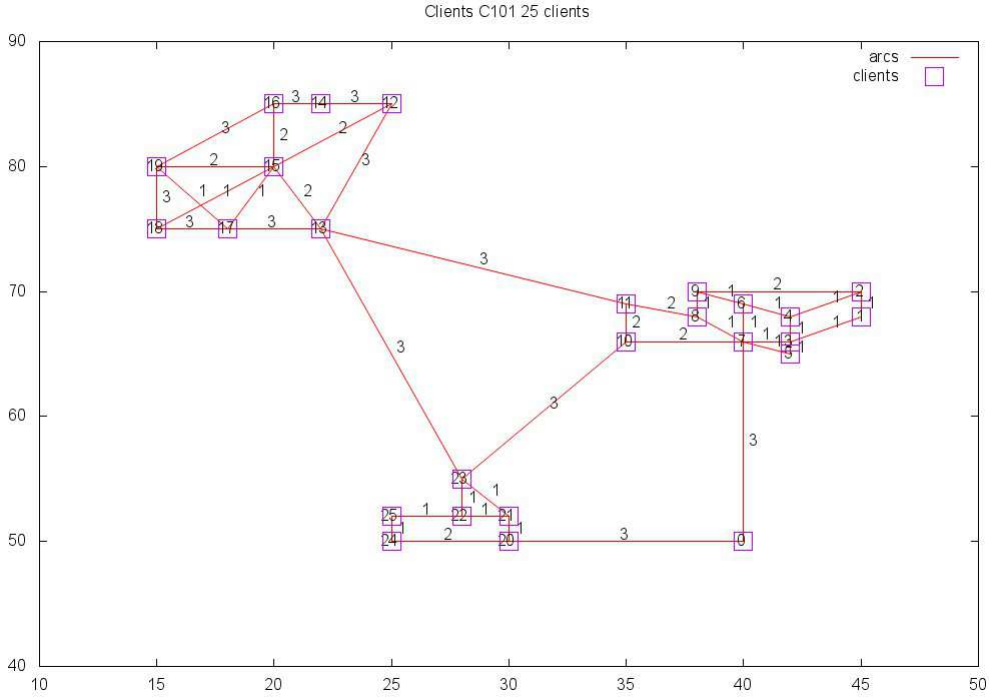

FIGURE 2.3 – Assignation de profils sur l'instance C101

On fournit à l'entrée de notre programme un deuxième fichier indiquant les arcs présents dans le modèle et le profil assigné (voir tableau 2.2.2).

	0	1	2	3	4	5	6	7	8	9	10
0	0										
1	0	0									
2	0	1	0								
3	0	1	0	0							
4	0	0	1	1	0						
5	0	0	0	1	0	0					
6	0	0	0	0	1	0	0				
7	3	0	0	1	0	1	1	0			
8	0	0	0	0	0	0	0	1	0		
9	0	0	2	0	0	0	1	0	1	0	
10	0	0	0	0	0	0	0	2	0	0	0

TABLE 2.II – Assignation des arcs et profils sur les 10 premiers clients

Sur cet exemple(tableau 2.2.2) on peut lire que l'arc entre les sommets 2 et 1 est un arc de profil 1.

2.2.3 Scénario

Le dernier fichier nécessaire pour notre programme est la définition du scénario, c'est-à-dire la définition de la vitesse sur chaque type de route. On le définit à l'aide du tableau 2.2.3. Il y a 3 types des voies dans ce scénario. Pour le premier type de voie, il existe 7 intervalles de temps. Le premier intervalle en minutes est [0, 420] et son facteur de vitesse est de 1. Si on regarde plus bas dans le fichier, pour la voie de type 2, dans le même intervalle, le facteur de vitesse est de 1.5, ce qui met en évidence que le transit pour la voie 2 est plus rapide que pour la voie 1.

3	
1	7
420.0	1.0
540.0	0.54
...	
1440.0	1.0
2	7
420.0	1.5
540.0	0.81
...	
1440.0	1.5
3	7
...	

Table 2.III – Scenario 1 avec 3 profils de vitesses et 7 intervalles

2.3 Solution initiale

Quelque soit le choix de la méthode de résolution, avoir une bonne solution de départ obtenue rapidement, permet toujours d'accélérer l'algorithme. C'est le rôle des heuristiques. La première idée a été d'utiliser une heuristique de type glouton c'est-à-dire qu'on choisit un véhicule et on remplit sa tournée client par client en choisissant à chaque fois le meilleur client potentiel parmi les clients non traités. Dans notre cas, le meilleur client est le client le plus proche puisqu'on souhaite minimiser la durée de trajet de chaque véhicule.

Appliquer ce type d'heuristique nous amène rapidement vers un optimum local et il faut faire appel aux métaheuristiques pour avoir une chance de trouver une solution globale. Pour comprendre ce phénomène, il suffit d'avoir l'image en deux dimensions d'une région montagneuse (voir figure 2.3).

L'objectif est d'aller le plus bas possible, mais en utilisant une heuristique on ne se permet pas de trouver la plus basse position possible, car une fois arrivé sur un minimum local on ne trouve pas de mouvement permettant de descendre plus bas. Pour avoir une chance de trouver plus bas il faut se permettre de détériorer la solution actuelle c'est-à-dire sur cet exemple de remonter.

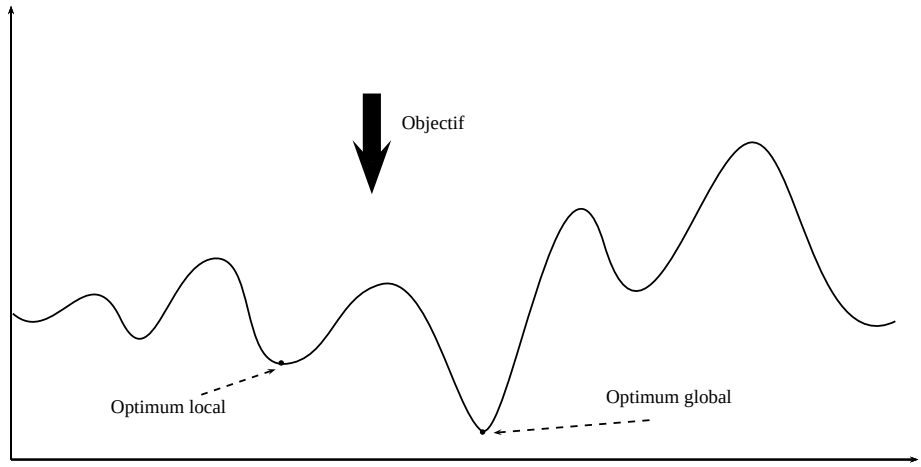

FIGURE 2.4 – Optimum Local vs Optimum Global

2.4 VND(Variable Neighbhorood Search)

Pour notre problème nous avons choisi d'utiliser une métaheuristique de type VND (Descente à Voisinage Variable) introduite par Pierre Hansen il y a une quinzaine d'années. Cette métaheuristique ne choisit pas une unique façon d'améliorer la solution : on parle alors de voisinages. Pour notre problème de tournées de véhicules les voisinages peuvent être :

- inverser l'ordre de visite des clients pour le même véhicule.
- échanger des clients entre les véhicules.
- croiser le début d'une tournée d'un véhicule avec la fin d'une autre tournée.
- ...

Pour plus de détails sur les voisinages, se référer à l'article présenté en chapitre 3. Nous utilisons dans notre VND pas moins de 12 voisinages différents !

L'algorithme nécessite une solution initiale (celle-ci nous est donnée par l'heuristique présentée à la question précédente). Le but est ensuite de faire évoluer la solution courante. On commence par visiter le premier voisinage afin de trouver le meilleur mouvement améliorant. Si aucun mouvement n'améliore le résultat, on teste le voisinage suivant, si au contraire on trouve une amélioration, on revient sur le premier voisinage. L'algorithme s'arrête lorsque tous les voisinages ont été traversés sans amélioration. L'ordre des voisinages possède une grande importance, car les premiers mouvements vont être explorés plus souvent que les derniers. C'est pourquoi on commence par classer

nos voisinages par complexité : les voisinages les plus simples, c'est à dire les plus rapides à exécuter, en premiers.

Dans ce mémoire nous avons essayé un classement des voisinages de façon dynamique. Les voisinages étaient initialement classés selon leur complexité puis reclassés au cours de l'algorithme suivant le ratio

$$r = \frac{vi}{tc + \epsilon} * \frac{ni}{nc}$$

avec

- tc : temps passé cumulativement dans ce voisinage.
- vi : gain total obtenu grâce à ce voisinage.
- ni : nombre total de fois où ce voisinage a permis d'apporter une amélioration.
- nc : nombre total d'appels du voisinage.

Mais finalement nous sommes repassé à la version standard avec un classement fixe selon la compléxité. L'avantage de cette métaheuristique est le fait qu'elle soit très modulaire, on peut facilement rajouter ou enlever des voisinages sans modifier le reste de l'algorithme.

2.5 Génération de colonnes

La génération de Colonnes est un outil de recherche opérationnelle qui permet de résoudre des problèmes de grosses tailles en divisant le problème en deux parties. Elle est basée sur le principe de décomposition mathématique de Dantzig Wolfe introduite en 1960 [10]. Pour le problème de tournées, chaque colonne va représenter la tournée d'un véhicule. On va avoir une colonne binaire de la taille du nombre de clients, chaque 1 dans cette colonne correspond à la présence du client dans la tournée. Par exemple, avec 7 clients, un véhicule représenté par la colonne 0100110 traiterait les clients 2, 5 et 6. À chaque colonne on associe un coût représentant la durée de trajet. On remarque que cette modélisation ne garde pas l'ordre de visite des clients, on ne sait pas si l'ordre de visite est 0->2->5->6->0, 0->6->5->5->0, etc. il faudra donc une autre structure permettant de garder l'ordre des tournées. La génération de colonnes fonctionne alors comme suit :

- Problème maître : c'est un problème de couverture il faut trouver un ensemble de colonnes tel que tous les clients soient traités et tel que la somme des coûts des colonnes soit minimale.
- Sous-Problème : il s'agit de générer des colonnes valides, réalisables. C'est-à-dire construire des tournées respectant les fenêtres de temps des clients, la capacité des véhicules et utilisant la vitesse réelle.

Le problème maître est un problème linéaire, il est résolu par CPLEX®, un des meilleurs logiciels pour résoudre les problèmes linéaires. Le sous-problème est non linéaire en raison de la dépendance au temps, car la vitesse sur une route dépend d'une autre variable qui est l'heure de départ. Donc, pour résumer, nous souhaitons minimiser le temps de trajet entre chaque client et la durée de trajet va dépendre de l'heure d'engagement sur chacun des arcs. Ce problème non linéaire est résolu par notre métaheuristique. Notre VND fonctionne donc comme un générateur de colonnes. On lui fournit une solution initiale au problème de TDVRPTW, celle-ci la fait évoluer et nous renvoie la solution améliorée. Au cours de sa recherche, toutes les colonnes susceptibles d'améliorer la solution (repérées par leur coût réduit) sont ajoutées dans notre problème maître. L'algorithme s'arrête lorsque le sous-problème n'est plus capable de trouver de colonnes améliorantes. Lorsqu'il est possible de traiter le sous-problème de manière exacte, cet algorithme nous assure qu'il n'existe aucune autre colonne améliorante et donc que l'optimum trouvé est bien l'optimum global. Malheureusement dans notre cas, étant donné que les colonnes sont générées de manière heuristique, nous perdons cette propriété ... L'algorithme de génération de colonnes est illustré sur la figure 2.5.

Encore une fois, le détail est indiqué dans l'article, cette partie a juste pour but de présenter le fonctionnement général.

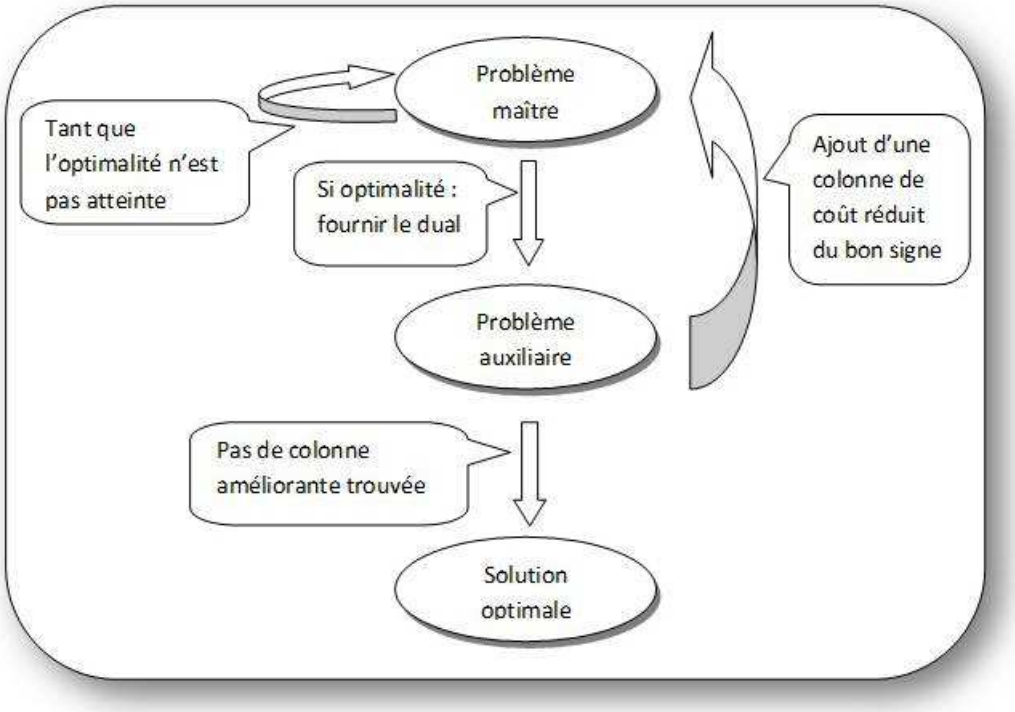

FIGURE 2.5 – Algorithme de génération de colonnes

2.6 Parallélisation

En regardant la formulation de la génération de colonnes on s'aperçoit que le sous-problème est en fait juste un générateur de colonnes, pourquoi ne pas ajouter d'autres générateurs de colonnes fonctionnant en parallèle ?
La VND fonctionne sur une solution du problème de TDVRPTW. Mais il est intéressant d'avoir des générateurs de colonnes qui vont générer des colonnes variées, non reliées entre elles. Le rôle du problème maître sera alors de sélectionner celles qui formeront notre solution finale. Nous avons donc rajouté des mouvements fonctionnant en parallèle de la VND.

On remarque aussi que notre problème maître optimise le coût total de notre solution, mais que, pour une entreprise, l'optimisation du nombre de véhicules est aussi très importante. Pourquoi ne pas rajouter alors un problème maître fonctionnant sur le nombre de véhicules ?

Au final, nous avons décidé d'avoir deux problèmes maîtres, un travaillant sur le coût et un sur le nombre de tournées, les solutions qu'ils trouvent sont fournies à deux VND différentes. On ajoute aussi des voisinages supplémentaires qui vont apporter de la variété à nos colonnes.

Au bout du compte, notre algorithme est modulaire à plusieurs niveaux : choix dans les voisinages des VND et dans les voisinages additionnels. C'est aussi un algorithme multiobjectifs.

2.7 Développement

Tout ce projet a été développé en langage C++, il fait appel au logiciel CPLEX®. Le programme compte à ce jour plus de 15 000 lignes de codes ! La génération de colonnes est très étudiée au GERAD®et la parallélisation apporte une vision nouvelle à cet algorithme.

Chapitre 3

A Column Generation Heuristic for the TDVRPTW

Vincent Huart, Sylvain Perron, Christophe Duhamel and Gilles Caporossi

Abstract :

This paper studies the Time Dependent Vehicle Routing Problem with Time Windows (TDVRPTW), an extension of the Vehicle Routing Problem with Time Windows (TRPTW) where the travel time between two clients can vary over the time. The main interest of the TDVRPTW is to better handle the dynamic nature of the travel time, especially in urban contexts where there is often a strong variation in terms of traffic. We propose a parallel heuristic for solving the TDVRPTW. It is based on the column generation techniques of linear programming as well as on Variable Neighborhood Descent(VND). The robustness and efficiency of the method is validated by a detailed comparison of benchmark results on the TRPTW. Results on Solomon instances adapted to the case of time-dependency are also presented showing the impact of time-depency on the solution obtained.
Keywords: Vehicle Routing, Time-Dependency, Column Generation, Variable Neighborhood Descent, Heuristic

3.1 Introduction

Routing problems often are the very core of transportation systems in logistics. The Vehicle Routing Problem (VRP), introduced by Dantzig and Ramser in 1959 [9], is the most prominent as it consists in building a set of routes of minimal cost to visit a set of clients with known demands. The VRP has been extensively studied both with exact and heuristic methods (see, for instance, the extensive surveys [16, 39]) and many extensions have been proposed such as adding time windows on clients (VRPTW), distinguish pickup and delivery clients (VRPPD), allowing split delivery (SDVRP), adding a capacity on vehicles (CVRP), etc. This paper focuses on the Time Dependent Vehicle Routing Problem with Time Windows (TDVRPTW), an extension of the VRP considering time-windows for each client as well as time-dependency.

In most studies on vehicle routing, the travel time between clients is supposed constant over time. It corresponds usually to the shortest travel time (without congestion) or to the average travel time. In the time-dependent Vehicle Routing Problem - TDVRP), the travel time between two clients can vary over the time (see, e.g., [28]). Its main interest is to better handle the dynamic nature of the travel time, especially in

urban contexts where there is often a strong variation in terms of traffic. By doing so, one can expect vehicles routing solutions that avoid the congestion nodes when traffic jam happens. Thus, real-life phenomena as periodic congestion (daily commuting for instance) can be better handled. This can strongly improve the efficiency for companies specialized in fast deliveries and guaranteeing deliveries with quality of service. It is important to notice that the time-dependent extension of the VRP is different to the real-time extension. In the first case the information about speed is known beforehand. It usually correspond to a set of scenario based on collected statistics about speed. Therefore, the routes are designed as the beginning and they are not adapted during the day. At contrary, congestion due to accidents or other punctual and random incidents can be considered in the real-time extension and therefore, the routes may be modified over the time. The TDVRP cannot be seen just as a superposition of graphs with different speeds since the shortest path – in terms of duration – between two clients will change during the day. Besides, travel time along this path might often change. Combinatorial on the number of graphs would be too large.

This paper presents a parallel heuristic based on column generation and on Variable Neighborhood Descent to solve the TDVRPTW. The method uses several parallel neighborhood structures to explore the solution space, leading to efficient diversification. Moreover, the method considers simultaneously (in parallel) the two important objectives in vehicles routing problem: the minimum number of vehicles on one hand and the minimum travel cost on the other hand. The robustness of the method is validated by comparing our results for the VRPTW with benchmark results on the classical Solomon instances. Moreover, the use of the method allows us to clearly illustrate the impact of the time-dependency structure on the design of the routes obtained.

The paper is organized as follows: Section 2 reviews the most important works on the TDVRP as well as classical solution methods for the VRPTW. Section 3 proposes a mathematical formulation of the TDVRPTW. Section 4 describes the proposed column generation heuristic for solving the TDVRPTW. Numerical experiments are presented in Section 5. The paper concludes with a final discussion in Section 6.

3.2 Literature review

In this section, the most important studies on the Time-Dependent Shortest Path problem with Time Windows (TDSPPTW) and on the TDVRPTW are first reviewed. Then, we present works on the VRPTW focusing on papers applying Column generation techniques to this problem.

3.2.1 Studies on the TDVRPTW

To the best of our knowledge, the first study on the use of time-dependency for the VRP has been done by Malandraki [26] and Malandraki and Daskin [27]. They propose a mixed integer formulation and a greedy heuristic for solving the TDVRP without time-windows. The authors also provide a branch-and-cut algorithm for solving small instances. At the same period, Hill and Benton [19] introduce a time-dependent problem focusing on the way to represent and store data. This task is of practical importance as many different shortest paths have to be stored for every pair of nodes since the duration for a path between two nodes changes over time. The next important studies on the TDVRP appeared in the 2000s when considering only time-dependency and not real-time. The first paper on the TDVRPTW is [36] in which the authors propose a tabu search. They also introduce a statement, the First In First Out (FIFO) property, about the way to model the vehicles behavior in the time-dependency : vehicles can not overtake in the traffic. For instance, let k_1 and k_2 be two vehicles going from node i to node j and starting at $r_i^{k_1}$ and $r_i^{k_2}$. If $r_i^{k_1} \leq r_i^{k_2}$, then vehicle k_1 must necessarily arrive at j before k_2. This assumption is kept in our problem statement and modeling. Fleischmann et al. [15] propose the first real-life application of the TDVRPTW. Their results on Berlin data, based on a construction algorithm, show that time-dependency has a significant impact. This work raised the interest of the community on this topic. At the same time, Osvald [2] consider the TDVRPTW using a travel speed discretization. Unlike most studies, they do not use a complete graph with euclidean distances. Considering paths rather than direct arcs between nodes is especially interesting in the time-dependent problem because it will impact all the solutions. Indeed, let's imagine that a highway is blocked, we need to find another path to go to our destination, and this can impact

the entire route. In 2007, Hashimoto [18] propose an iterated local search using 2-opt* and Or-opt neighborhoods and travel costs evaluated by dynamic programming. Our algorithm shares some features with this method since Variable Neighborhood Descent (VND) also uses several neighborhoods. In 2009, Soler [11] propose an exact algorithm for solving small instances of the TDVRPTW. Finally, one of the most recent paper on the TDVRPTW presents an iterative route construction and an improvement algorithm [3]. Some benchmark instances based on Solomon instances with different speed profiles and keeping the same network structure are reported.

Thus, several meta-heuristics have been proposed and tested for the TDVRPTW, the most notable being:

- Tabu search: Ichoua et al. [36] proposed an extension of the parallel tabu search developed by Taillard [38]. It relies on a adaptive memory of routes first filled with solutions generated by a randomized insertion heuristic and improved by tabu search. Then, solution are built using routes in the memory and improved by tabu search. This meta-heuristic uses CROSS exchanges neighborhood structure. Both the time-dependent and the real-time contexts are considered;
- Genetic algorithm: Haghani et al. [1] considered the time-dependent and the real-time VRPTW with an heterogeneous fleet. They first propose a mathematical formulation for the problem. It is use to provide lower bounds to the problem. Then they use a genetic algorithm to compute upper bounds;
- Ant colony algorithm: Donati et al. [14] propose a Multi Ant Colony System (MACS). The first ACS aims at minimizing the number of routes while the second one minimizes the total route length. Information is exchanged between the two procedures by transferring some pheromone values. Each ant build a solution in a way similar to the route extension constructive heuristic. A local search is also developed to improve the solution built by the ants. It uses several neighborhood structures such as client relocation, client exchange, bloc reversal, bloc relocation and bloc exchange. Balseiro et al. [5] use a similar idea, with an improved local search;
- Genetic algorithm: Zhao et al. [40] are also considering both time-dependent and real-time settings. They propose a simulator in which the genetic algorithm runs in

the background while a event handler manages the real-time events. Five criteria are considered. They are aggregated in a single objective function;

– Simulated Annealing: Kuo et al. [25] consider both a fuel consumption model and a transportation time model. A nearest neighbour heuristic is used to generate initial solutions. An improved cooling procedure is used.

Publications focusing on real-time VRP are not mentioned since we focus on time-dependent VRP.

3.2.2 Studies on the TDSPPTW

Computing the travel time between two nodes is a key subproblem of the TDVRPTW since it has to be called a large number of time, whatever the approach. Formally, the Time Dependent Shortest Path Problem with Time Windows TDSPPTW consists in computing the fastest path from an origin node to a destination node, at a given starting time and respecting the time windows. Either all the time windows or the time windows from only the origin and the destination nodes are considered. Travel duration evolves along the time period and solving it reduces to the computation of earliest arrival time for each node. Thanks to the FIFO property, waiting at some node does not allow to reduce the travel time and the problems keeps a low complexity. Dynamic programming is a classic way to efficiently solve this subproblem. We use a modified Dijkstra's label-setting algorithm. Our approach is similar to the method presented in [37]. Several improvements have been published to increase the computational efficiency of this algorithm. Kholer [23] gives some techniques to accelerate time dependent shortest path computation using bidirectional algorithms. Sanders [34, 35] presents the concept of "highway hierarchies" and compares the efficiency with Dijkstra's algorithm. More recently, Nannicini [32, 12, 31, 30] has presented improved dynamic programming methods to solve the problem. In this work we will use the simple algorithm. It is also worth noting Kok [24] has also used dynamic programming to compute time-dependent shortest path in order to provide analyze on congestion and on the way to avoid it.

3.2.3 Column Generation Algorithm for the VRPTW

Several methods have been proposed for solving the VRPTW (see, e.g., [16, 39] for references). Since our method is based on column generation (CG) techniques, we will focus on applications of the CG to the VRPTW. Desrochers et al. [13] present a set partitioning formulation of the VRPTW and develop an exact CG algorithm where the subproblem is formulated as a shortest path with time windows and it is solved by dynamic programming. The same master problem formulation may be used for the TDVRPTW but the subproblem is much more difficult to solve since it involves time-dependency. Kohl et al. [22] extend the work presented in [13] by adding cuts reducing the gap between the linear relaxation solution and the integer solution of the master problem. Prescott [33] presents a CG-based heuristic for the VRPTW using tabu search. Duhamel [7] presents an application of CG for solving real-life application of the VRPTW for oil wells maintenance.Christophe: je vais donner d'autres references

3.3 Mathematical Formulation

In this section, we present the mathematical modeling of the TDVRPTW. We first give a mixed integer programming formulation of the VRPTW similar to the one proposed in [22] and then present the way time-dependency is modeled in our work.

3.3.1 Mixed Integer Program for the VRPTW

We consider a set $I = \{1, 2, \ldots, n\}$ of n clients to be served. Each client $i \in I$ has a demand d_i, a service time s_i and a time windows $[a_i, b_i]$ representing the interval within which the service must begin. Then, we define $G(V, E)$ a complete graph where $V = \{0\} \cup I$ is the set of nodes representing the union of the depot, labeled 0, and the set of clients; E defines the set of arcs between those nodes such that $E = \{(i, j) | i, j \in V, i \neq j\}$. The set of vehicles is represented by M. Each vehicle has a capacity of q. A cost $c_{i,j}$ and a travel time $t_{i,j}$ are associated to arc $(i, j) \in E$. The parameters d_0, s_0, and a_0 are set to 0 and b_0 to a large value representing the fact that there no service time, no demand and no time window assigned to the depot. Note that b_0 is usually

set to 0 in VRPTW studies, imposing that all routes must leave the depot at time 0. However, we do not impose this condition as in case of time-dependency, it might be interesting to wait and leave the depot after the rush hour of the morning.

We use two types of decisions variables:

(1) for each arc $(i, j) \in E$ and for each vehicle $k \in M$, we define a binary variable $x^k_{i,j}$ which is equal to 1 if vehicle k directly travels from node i to node j and 0 otherwise;

(2) for each node $i \in V$ and for each vehicle $k \in M$, we define a continuous variable u^k_i representing the starting service time of vehicle k for client i.

The VRPTW can now be formulated as:

$$\text{minimize} \sum_{k \in M} \sum_{(i,j) \in E} x^k_{i,j} * c_{i,j} \tag{3.1}$$

subject to

$$\sum_{k \in M} \sum_{\{j:(i,j) \in E\}} x^k_{i,j} = 1, \forall i \in I \tag{3.2}$$

$$\sum_{\{i:(i,\ell) \in E\}} x^k_{i,\ell} - \sum_{\{j:(\ell,j) \in E\}} x^k_{\ell,j} = 0, \forall k \in M, \forall \ell \in V \tag{3.3}$$

$$\sum_{k \in M} \sum_{j=1}^{n} x^k_{0,j} = 1, \forall k \in M \tag{3.4}$$

$$\sum_{k \in M} \sum_{i=1}^{n} x^k_{i,0} = 1, \forall k \in M \tag{3.5}$$

$$\sum_{(i,j) \in E} d_i * x^k_{i,j} \leq q, \forall k \in M \tag{3.6}$$

$$(u^k_i + s_i + t_{i,j} - u^k_j) * x^k_{i,j} \leq 0, \forall k \in M, \forall (i, j) \in E \tag{3.7}$$

$$a_i \leq u^k_i \leq b_i, \forall k \in M, \forall i \in V \tag{3.8}$$

$$x^k_{i,j} \in \{0, 1\}, \forall k \in M, \forall (i, j) \in E \tag{3.9}$$

The objective (1) is to minimize the total cost. Constraint (2) forces to visit and serve each client only once. Constraint (3) is the classical flow conservation conservation. Constraints (4) and (5) ensure that each vehicle will start and finish his route at the depot. Constraint (6) states the respect of capacity vehicles. Constraints (7) and (8) deal with the time concordance. Constraint (7) ensures that if client i precedes

immediately client j on route k, then the starting service time of client j must be greater than the finish service time of client i plus the travel time between clients i and j. Constraint (8) imposes time windows on the starting service time for all clients. Constraint (9) is the binary constraint.
In this formulation the constraint (7) is not linear but it can be linearized using a large constant β:

$$(u_i^k + s_i + t_{i,j} - u_j^k) \leq \beta * (1 - x_{i,j}^k), \forall k \in M, \forall (i,j) \in E \tag{3.10}$$

The resulting model is a mixed integer linear program, called MIP-VRPTW in the next sections.

3.3.2 Time Dependent modeling

The model MIP-VRPTW must be adapted to the case of time-dependency. First, one can replace coefficient $c_{i,j}$ in (1) by $t_{i,j}$ expressing the fact that we want to minimize the sum of duration of all routes. Furthermore, $t_{i,j}$ now depends on the starting time from node i. Let r_i^k be the starting service time of client i by vehicle k, so we have $r_i^k \geq u_i^k + s_i$, meaning that departure from node i is at least the arrival time at node i plus the service time. The value $t_{i,j}^k$ is equal to the shortest time to go from node i to node j into the real network, not in $G(V,E)$ which is the complete graph with all connections. Indeed, $t_{i,j}^k = \sum_{n=i}^{n-1} \frac{d_{n,n+1}}{f_{n,n+1}(r_n^k)}$ where $f_{i,j}$ is a function describing speed on road (i,j). To obtain travel duration between node i and j we have to compute the duration of each road of the path between i and j respecting the departure of each encountered node. As a result, the constraint (10) and the objective become non linear. Since they cannot be linearized, the resulting mathematical model becomes non linear and too difficult to be solved by an exact solution method.

Speed profile on arcs

We will use the same representation for travel speed on arcs as in [40]. For each arc, a profile representing the arc type, must be assigned. In realife problem, these

types can represent different kinds of roads: highways, national routes, streets, etc. For each profile, we define a series of intervals within which speed travel differs from normal speed. For instance, we could define an interval $[7am - 9am]$ and put a speed coefficient inferior to 1 representing congestion. Outside these intervals, the speed coefficient is set to 1, meaning that the speed is normal. We can create different scenarios for each arc but only one can be active at a time. In our implementation, each profile of road can have a different number of intervals. Figure 3.1 shows an example with three profiles, three intervals for each profile and two scenarios can occur. For instance, if scenario 1 occurs, than for the roads having a profile 1, the speed is always less than or equal to the normal speed and also that the traffic is high during the interval T1 where the speed is 46 % under the normal speed.

	Profile	T1	T2	T3
Scenario 1	1	0.54	0.81	0.54
	2	0.81	1.22	0.81
	3	1.22	1.82	1.22
Scenario 2	1	0.33	0.67	0.33
	2	0.67	1.33	0.67
	3	1.33	2.67	1.33

Figure 3.1: Two different scenarios of three profiles with three intervals

Values as those reported in 3.1 are obtained by approximating the speed evolution function for each arc profile. This kind of function is continuous, as shown on the left part of Figure 3.2. To approximate such a continuous function, we discretize the time horizon in several steps and then take the average speed coefficient on each interval as shown on the right part of Figure 3.2. For instance, on interval T1 ([0-2]), the speed coefficient will be set to 2, corresponding to the mean speed coefficient of the continuous function within this interval.

Duration computation between two nodes

Modifying the speed profile on an arc will automatically change the travel time on each path using this arc. As a consequence, the travel time between two clients may differ depending on the departure time from the first client. Figure 3.3 shows

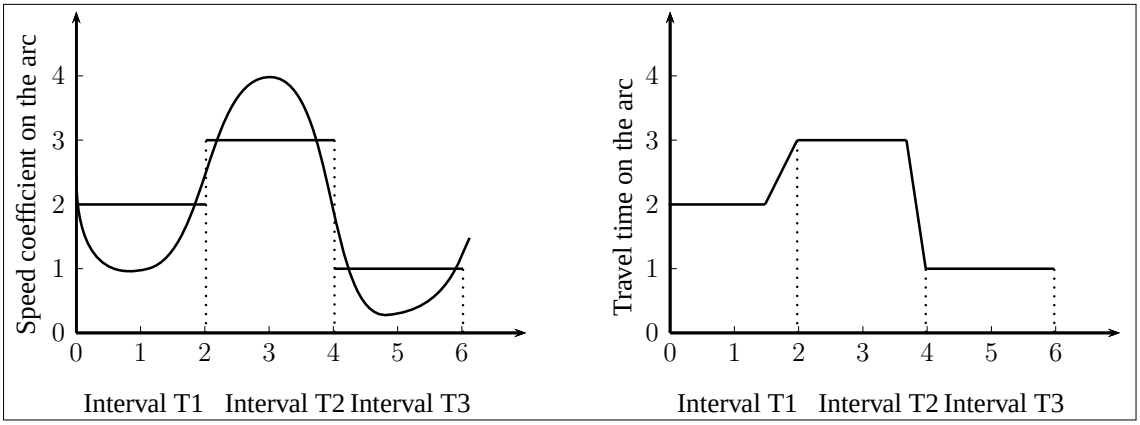

Figure 3.2: Speed evolution on an arc of length 1 as a step function

a representation of the travel time between two clients. Note that a change of speed profile on arc does not coincide with a change on the path. For computing the shortest paths, we use an adaptation of the well-known Dijkstra's algorithm. The paths are then stored in a matrix, MC where each element $mc_{i,j}$ represents the shortest path from i to j satisfying the duration and time windows, meaning that $mc_{i,j}$ is nonempty if and only if it exists a feasible path between i and j. Since the shortest path between i and j may change depending on the moment of the day, the element $mc_{i,j}$ will contain a list of paths, and the corresponding intervals of validity. Figure 3.3 shows an example with four intervals where the shortest path 1 is the valid one in the first interval, the shortest path 2 in the second interval, and so on. We have tested different implementations of the Dijkstra's algorithm and decided to keep the min binary heap implementation with a worst case complexity of $\mathcal{O}((|E| + |V|)log(|V|))$. To accelerate the process, we stop the algorithm as soon as the client is reached. In practice we have to choose a time step and we iterate between each pair of clients as long as the time windows are compatible. Indeed as we only need the path in which the vehicle travels directly from i to j, it is useless to take into account paths leaving i before the time windows $[a_i, b_i]$.

To compute the time travel on a path we use the Algorithm 1 and Algorithm 2:

– Algorithm 1 computes the travel time on an arc. This algorithm needs the current date and the distance to travel. It is a recursive function because the speed can change on the arc. So we first compute the next step of speed modification, if we can travel the entire arc with the current speed without exceeding the next step,

the duration is just $\frac{distancetotravel}{currentspeed}$; otherwise we have to compute the remaining distance to travel and then re-apply the function with the new speed.

– Algorithm 2 computes the duration on all paths, i.e., for each path, we apply Algorithm 1 for each arc of the path.

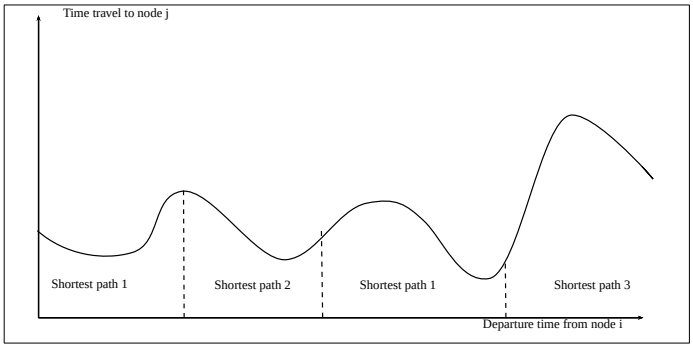

Figure 3.3: Travel time between clients i and j

Algorithm 1 Function to compute the travel time on an arc

Require: r_i is the starting time from i and $d_{i,j}$ the remaining distance between i and j

```
function computeDurationArc(r_i, d_{i,j})
    nSC ← lower bound interval current arc          ▷ Next date of Speed Change
    s ← speed of current arc on the interval
    if s*dBC ≥ d_{i,j} then                          ▷ We can do the entire arc
        return d_{i,j}/s;
    else
        d_{i,j} ← d_{i,j}-s*dBC                      ▷ Remaining distance
        cT ← dBC+r_i                                 ▷ Current time
        return dBC+computeDurationArc(cT,d_{i,j});
```

Algorithm 2 Function to compute the travel time on a path

Require: r_i is the starting time from i and $p_{i,j}$ is the path between i and j

```
function computeDurationPath(r_i, p_{i,j})
    dT ← 0                                          ▷ Total Duration
    cD ← u_i                                        ▷ Current date
    for each arc a ∈ p_{i,j} do
        dT ← dT + computeDurationArc(a,currentDate)
        cD ← cD + dT
    return cD
```

3.4 Proposed Solution Method

We propose a parallel heuristic based on column generation and on Variable Neighborhood Descent to solve the TDVRPTW. As mentioned in Section 3.2, column generation (CG) has already been proposed for solving extensions of the VRP but to the best of our knowledge, parallel heuristic based on column generation and on Variable Neighborhood Descent has never been used before for any extensions of the VRP.

In this section, we explain in details our algorithm. We first discuss the main principles of column generation techniques and its application to our problem. Then, we describe the Variable Neighborhood Descent and the types of neighborhoods considered in our algorithm. We conclude the section by presenting the global design of the proposed parallel heuristic method.

3.4.1 Column Generation

Following the principles of the Dantzig-Wolfe decomposition, column generation techniques applied to the TDVRPTW consists to split the problem in two parts:

1. The Master Problem (MP): an optimization problem which selects routes among a pool of feasible routes in order to visit all clients only once.
2. The Subproblem (SP): an optimization problem whose role is to generate routes to be added to the pool of the MP. In order to simplify the formulation and the solution of the MP, it is possible to design the SP in such a way that the columns generated will satisfy hard constraints such as the vehicle capacity, the time-

windows for the clients, the duration computed according to time-dependency, etc.

In classical CG algorithms, a column generated by the SP is added to the MP only if its reduced cost is negative, meaning that the column might improve the objective function value of the continuous relaxation of the MP or, in other words, that the current solution of relaxation does not satisfy the optimality conditions.

Master Problem

As in classical CG algorithms for the VRP, the MP is formulated as the following set partitioning program:

$$\text{minimize} \sum_{i \in P} x_i \cdot c_i \tag{3.11}$$

subject to

$$\sum_{i \in P} x_i \cdot v_{i,j} = 1, \forall j \in V \tag{3.12}$$

$$x_i, \in \{0, 1\}, \forall i \in P \tag{3.13}$$

where P denotes the pool of columns (routes patterns) already added to the MP; x_i is a Boolean decision variable indicating if pattern i is selected or not; $v_{i,j}$ is a Boolean parameter indicating if vehicle route $i \in P$ visits client $j \in V$ or not; c_i is the cost of column i. The cost of a column depends on the objective function under consideration. For instance, c_i may be set: (i) to the total duration of the route described by pattern $i \in P$ if the objective is to minimize the total duration of all the selected routes, (ii) to 1 if the objective is to minimize the total number of vehicles; (iii) to a weighted sum of the two objectives for considering both objectives simultaneously, etc.

Partitioning constraints (3.12) may be relaxed:

$$\sum_{i \in P} x_i \cdot v_{i,j} \geq 1, \forall j \in V. \tag{3.14}$$

The solution of the resulting covering problem may be modified easily in a post-optimization phase to insure covering each client only once.

Finally, in order to get pricing (dual) information for the SP, integer constraints have to be relaxed and therefore constraints (13) are replaced by:

$$0 \leq x_i \leq 1, \forall i \in P. \tag{3.15}$$

In the remaining sections, we will denote by MIP the model with objective (3.11) and constraints (3.14), (3.13) while the term *relaxed* MP will refer to model (3.11), (3.14), (3.15).

Subproblem

As mentioned before, the role of the subproblem is to generate new columns (routes patterns) to be added to the MP. A column is added if its actual reduced cost is negative. Using the notation introduced earlier, the reduced cost of column i of the *relaxed* MP may be computed by:

$$c_i - \sum_{j \in V} v_{i,j} \cdot \pi_j \tag{3.16}$$

where π_j is the dual variable associated to the covering constraint (3.14) of client j. Replacing each parameter $v_{i,j}$ by a binary decision variable y_j equal to 1 if client j is part of the route and 0 otherwise, we obtain the following optimization problem:

$$\min_{y \in Y} f(y) - \sum_{j \in V} y_j \cdot \pi_j \tag{3.17}$$

where Y is the set of routes satisfying constraints on the vehicle capacity, the time-windows for the clients, and the duration computed according to time-dependency and $f(y)$ is the cost function of the MP for the column associated to y. There are two major difficulties when solving this optimization problem: the domain Y is hard to express explicitly in context of time-dependency and the value of $f(y)$ depends not only the clients visited but also on the order in which the clients are visited. Indeed, a column of our MP is represented by a binary vector with a size equal to the number of clients. However, the cost of this column depends also on the order these clients are visited and

also on the starting time of the route. For instance, suppose a problem with only 5 clients and a route visiting clients 2, 4 and 5. This route may be represented by the binary vector [0 1 0 1 1] in the MP but there are 3! possible permutations of these three clients: some of them may be unfeasible according to time-windows, some of them may lead to a different cost value. In our implementation, we use, outside the LP associated to the MP a more complete representation of a column than a simple binary vector to store the different routes to allow us getting the information about the visiting order associated to each route and the departure time of each client.

Due to its difficult resolution, we propose to solve the SP heuristically using the idea being the Variable Neighborhood Descent meta-heuristic (VND).

3.4.2 Variable Neighborhood Descent (VND)

To generate columns to be added to the MP, we use the metaheuristic Variable Neighborhood Descent(VND) introduced Brimberg [6]. VND is a metaheuristic, or framework for building heuristics, aimed at solving combinatorial and global optimization problems. VND is a variant of the Variable Neighborhood Search (VNS) metaheuristic which undergone many developments and has been applied in numerous fields (see [17] for a recent survey). Algorithm 3 presents the steps of a basic version of VND.

Algorithm 3 VND algorithm

```
Data

N^k ← {set of Neighborhoods}
x ← initial solution
k ← 1                                          ▷ First Neighborhood

while k ≤ k_max do
    x' ← best neighborhood of x, x' ∈ N^k(x)
    if x' is better than x then
        x ← x'
        k ← 1                                  ▷ Return at first Neighborhood
    else
        k ← k+1                                ▷ Next neighborhood
```

The efficiency of VND depends on the choice of the neighborhoods and also on the order in which they are used. The use of several neighborhoods allow to explore different areas of the solution space in order to escape from local optima and perhaps find new improving solutions. We will explain in details the neighborhoods used in our algorithm.

Neighborhoods

Internal 2-opt

Classical move used in VRP aims at avoiding crossing in the route. The move may be viewed as a switch either on arcs or on nodes. In Figure 3.4, we illustrate this neighborhood on a small example where the arcs (i, j) and $(i+1, j+1)$ are removed and the route is reconnected by adding arcs $(i, i+1)$ and $(j, j+1)$. Another way to see this move is that nodes $i+1$ and j are swapped in the route $0 \to i \to j \to i+1 \to j+1 \to 0$ to obtain the route $0 \to i \to i+1 \to j \to j+1 \to 0$.

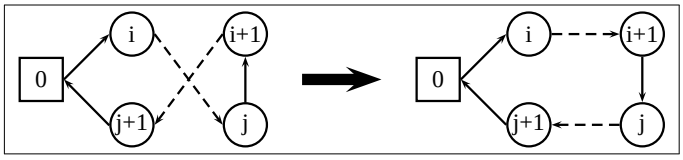

Figure 3.4: Illustration of internal 2-opt move

Internal exchange

This move is a generalization of the previous one in which λ clients of a route are swapped in such a way that each of these clients will change their position on the resulting route. Figure 3.5 illustrates this move with a swap of size 2 (top-right) or 3 clients (bottom-right). The complexity of this move increases rapidly with λ. Indeed, there is only one way to swap two clients but there are 3 ways to swap three clients and nine ways to swap 4 clients. In fact a disarranging is defined by

$$D_\lambda = \lambda! * \sum_{i=0}^{\lambda} \frac{(-1)^i}{i!} \approx_{\lambda \to \infty} \frac{\lambda!}{e}.$$

For $\lambda = 5$ we already have $D_5 = 44$, $D_6 = 265$. It is therefore too time consuming for $\lambda \geq 5$ as we must perform this move for each different combination of the selected

clients. We have many symmetries so the number of combination is $A_k^\lambda = \frac{\lambda!}{(\lambda-k)!}$. Finally for $\lambda = 10$ clients in the route and if we use a swap of size 4 we have $A_4^{10} * D_4 = 5040 * 9 = 45360$ moves to evaluate. To limit the complexity of applying this move, we propose to swap no more than 4 clients at a time in our algorithm.

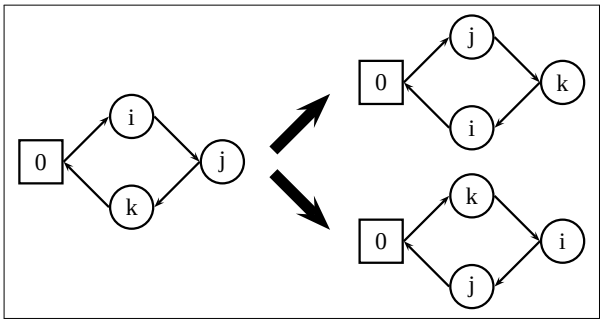

Figure 3.5: Illustration of internal Exchange of size 3

External exchange

The external exchange, sometimes called "Swap", is more time consuming since we have to choose clients in two different routes. Let λ_1, λ_2 be respectively the number of clients to be swapped from route 1 and route 2, then the complexity of an External exchange of size 1 is $\lambda_1 \times \lambda_2$. For an external exchange of size 2, it is $A_2^{\lambda_1} * A^{\lambda_2}$. Therefore, if $\lambda_1 = \lambda_2 = 10$ an External exchange of size 2 will need $\frac{10!}{8!} * \frac{10!}{8!} = 8100$ moves in which each time two routes must be updated.

Cross

This move aims at crossing routes. It takes two routes and splits each of them in two parts, λ_1 and λ_2 being the positions where the first and the second route will be splitted respectively. Then, the first part of each route is merged with the second part of the other route. Figure 3.6 illustrates this move with a value of 2 for λ_1 and λ_2. The complexity of the move is $\lambda_1 \times \lambda_2$.

Reorganization

This move tries to optimize the ordering of a route knowing all clients visited in the route. It consists in destroying the route and trying to build it again with the same clients. We start by ranking all the clients of the route by ascending order of their

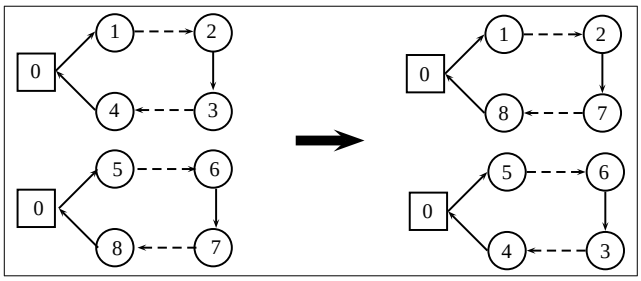

Figure 3.6: Illustration of a cross with $n_1 = n_2 = 2$

time-windows length and then we try to re-insert them one by one. The idea behind this order is that a client with a short time-windows is harder to insert in a route. Since we want to reinsert all the clients, if at any moment one client cannot be inserted, we rank him at the top and we start again from scratch the reinsertion. Many options may be considered for the reinsertion:

- insert at the first feasible position;
- insert at the first improving position according to the objective;
- insert at the first improving position according to the reduced cost;
- insert at the best position according to the objective;
- insert at the best position according to the reduced cost.

Internal Relocate

This move is nearly the same than the previous one except that we remove only a subset of clients instead of all the clients of the route. This neighborhood was used up to a size of two, meaning a reinsertion of at most two clients. Reinserting more clients will lead to a neighborhood similar to the previous one and will raise the complexity of the selection of clients to be reinserted. The strategies for the reinsertion of the clients are the same as for the previous move.

External Relocate

It removes a subset of clients from a route and tries to reinsert them in another route. We do not use this neighborhood with a subset bigger than two, because of its complexity growing too rapidly. Once again we use the same strategies for the reinsertion.

3.4.3 Global design of our heuristic Column generation

In order to explore several areas of the solution space and also to take into account both objectives (travel cost and number of vehicles) simultaneously, we have decided to embed our column generation and VND into a parallel framework. Using parallelization allows to diversify the search by considering several subproblems at the same time where each subproblem uses a single neighborhood and tries to add negative reduced cost columns at the same time in a shared pool. Moreover, using parallelization it is possible to consider more than one master problem at a time in order to take into account several objectives. For the TDVRPTW, we can consider two master problems: (i) one MP optimizing the number of vehicles; (ii) one MP optimizing the travel cost of the routes.

Figure 3.7 shows the global design of our parallel column generation heuristic where the horizontal axis represents the threads working in parallel during the process. Here are the main components of our algorithm:

Dual variables

During the process, the dual variables are continuously updated by the relaxed master problems for both objectives and stored in a shared memory storage (see the box on the top of Figure 3.7). Each VND uses these dual variables to evaluate the reduced cost of a column.

Columns pool

Each MP uses the same pool of columns which is filled by the heuristics. We have three heuristic kind: (1) A greedy heuristic which open a new route and fill it with the nearest client in term of duration until none of the remaining clients can be added. (2) A trivial heuristic which use a vehicle by client. (3) And a heuristic which randomly fill the routes.

Relaxed master problems

Solving the relaxed MPs is done by a single thread which calls Cplex in turn for getting the optimal solution of each relaxed MP. Since it is not time consuming (each call takes

less than 0.01 second using Cplex), this task is repeated all the time during the process to update frequently the dual variables values associated to each objective. The two MPs are the following:

- MP on travel cost: LP relaxation of a set covering problem optimizing the sum of all routes travel costs;
- MP on the number of vehicles: LP relaxation of a set covering problem optimizing the sum of the vehicles used, i.e., the sum of the number of columns used.

Integer master problems (MIP)

Two threads (one for each objective) are dedicated to the solution of the MIP, i.e., the MP with the integer constraints. Each thread works in the same manner solving in sequence the MIP by Cplex using the actual set of columns and then trying to improve the best known integer solution using VND.

As we relaxed the partitioning constraint in the MIP, Cplex may obtain an integer solution with a multiple covering for some clients. Since our VND needs a feasible initial solution, multiple coverage must be removed. To do so, we remove each multiple covered clients in all columns where he belongs, add all the resulting columns and solve again the MIP by Cplex. The obtained feasible integer solution is used to initialize the VND.

The MIP optimizing the travel cost is accelerated in Cplex by giving the best known solution in term of travel cost (WarmUp) and by adding a bound on the cost (Cuttof). The MIP optimizing the number of vehicles works differently. Minimizing the number of vehicles may lead to a solution with a very large travel cost. Also, we already have an upper bound on the number of vehicles by using the incumbent solution on the travel cost. The goal of the MIP on the number of vehicles is to find a solution reducing the number of vehicles. To do so, we impose a constraint on the number of vehicles to be strictly lower to the known upper bound minus one and we minimize the sum of the travel cost of the routes plus a big fixed cost corresponding to the cost of using a new vehicle. Note that we can no more use a warmup with Cplex as no feasible solution with such a number of vehicles is known up to now.

To accelerate the process, for both MIPs, we limit the branch-and-bound process of Cplex by imposing either a time limit or limit on the tree size. Furthermore, before calling a MIP, we compare the best integer solution known with the relaxed one and we ask Cplex to solve the MIP only if the gap is significant. For instance, on the optimization of the number of vehicles, it is useless to solve the MIP if the actual relaxation is not strictly lower than the known upper bound minus one.

Each VND works on a global solution of the problem, i.e., a set of valid routes covering each client once. Algorithm 3 is applied taking this integer solution as an initial solution. Then, the VND heuristic tries to improve the integer solution by applying the neighborhoods described in Section 3.4.2. It therefore generates new columns to be added to the pool regarding their actual negative reduced cost (for both sets of dual variables) but also columns leading to a gain on the cost when calculating the effect of a move even if their actual reduced cost is positive. After each neighborhood studied in our VND, we compare the best integer solution with the relaxed one, if there is an interesting gap we run the MIP and update the best solution. If the best solution is updated and different of the actual solution of the VND, we restart the VND with this new solution. Otherwise, we continue the VND.

Note that in each VND, the dual variables values are updated between each neighborhood. Also, VND are communicating during the process, i.e. if the VND working on the best integer solution in term of vehicles find a better cost, the other VND will stop and restart from this solution and inversely.

Subproblems

The remaining threads are used to diversify the search and to generate new columns. Each of this thread repeats in turn the same procedure: obtain the actual dual variables values, take a valid column(s) and then apply a single neighborhood in order to find new columns to be added to the pool, i.e., columns having a current negative reduced cost for at least one of the objectives. This heuristic may be viewed as a VND algorithm with a neighborhoods set of size 1. Note that depending on the neighborhood considered, the initial solution is either a single column or a set of two columns. In the case of two

columns we can either randomly pick them from the pool or take only one from the pool and generate the second with random clients different.

At the end, the more we have threads the more we can add new neighborhoods and the more we add diversification. On the other hands, some threads may decelerate the overall process. Even if we can launch as many threads as we want on a processor we need that each thread run alone on a single core to ensure that the neighborhood is not get run over by another function.

Stopping condition

We need a good stopping criterion, with all those subproblems and two master problems we cannot stop when no column are found because if we have a degenerated problem we will find a lot of columns even if optimum is reached. We chose to end when both VND have finished and MIPs could not find a new one solution, i.e. the both VND cannot restart with a new solution.

3.4.4 Settings

The VND tries to improve the MIP solutions by using the same set of 12 neighborhoods which are, in order: (1) 2-Opt; (2) Reorganization; (3) Internal relocate of size 2; (4) Internal exchange of size 2; (5) External relocate of size 1; (6) Internal relocate of size 1; (7) Cross; (8) Internal exchange of size 3; (9) Internal exchange of size 4; (10) External exchange of size 2; (11) External relocate of size 2; 12 External exchange of size 1. The order is really important, we chose to apply a rank based on the complexity, simplest neighborhoods at the top.

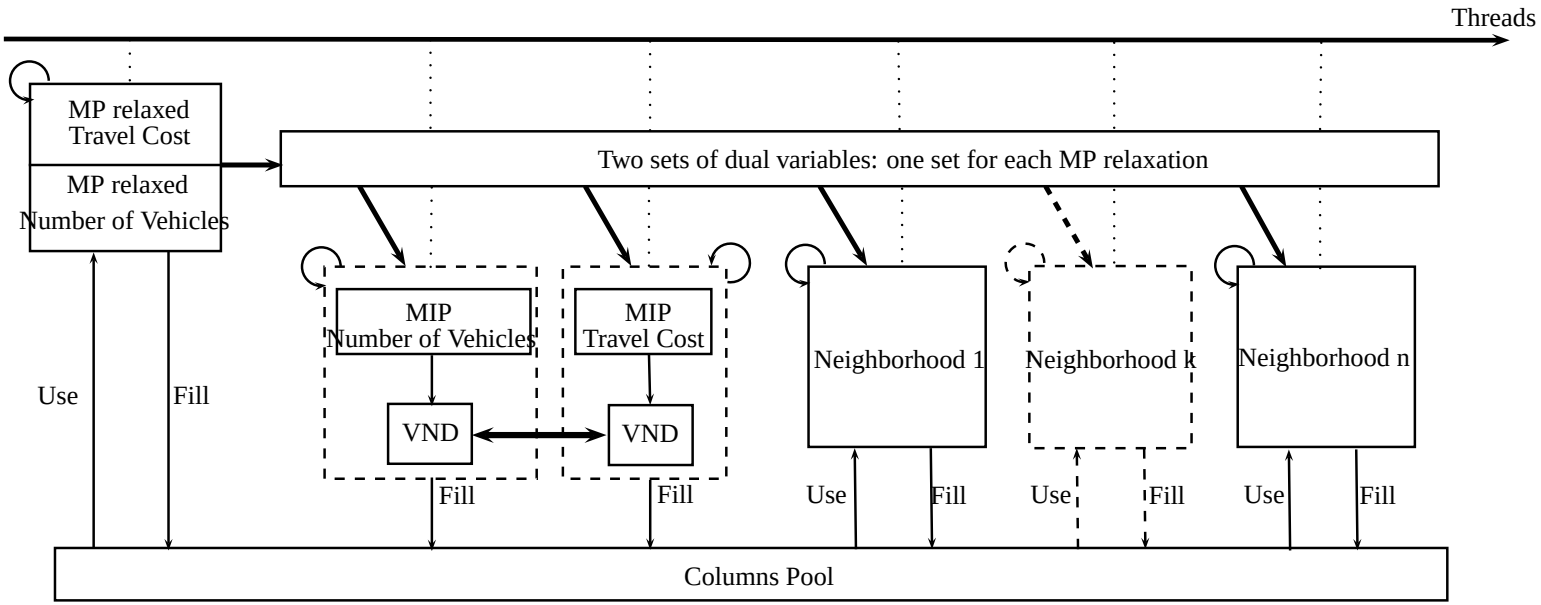

Figure 3.7: Global design of our parallel column generation heuristic

3.5 Computational results

The numerical experiments are done on an Intel(R) Xeon(R) E5-2665 processor running at 2.40 GHz with cores and hyperthreading. This means 32 threads can be considered without any major slowdowns. In practice, 15 threads used, which means we add 12 neighborhoods to provide diversity. The more neighborhoods we add the better it is. Besides, the same neighborhoods can also be added several times.

To the best of our knowledge, no instances are publicly available for the TDVRPTW. Most authors used modified Solomon instances but do not provide their instances. Thus, our algorithm is first validated on unmodified VRPTW Solomon instances, without any time dependency. Then, some time-dependent instances will be defined and used.

3.5.1 Comparison with VRPTW

Solomon's instances for the VRPTW consists of 6 subsets: R1, C1, RC1, R2, C2 and RC2, depending on the the clients distributions (random, clustered and mixed) and the time windows (narrow, wide). We only consider instances C1, RC1 and R1 in this experiment. Those instances have narrow time windows and clients are respectively located in clusters, randomly and in a combination of randomly and in clusters. Recent articles about VRPTW also use new larger instances. Besides they do not report detailed results and provide instead a summary. We will compare here the quality of our results with the exact values in [22]. Then, we will compare our computation times with other articles. Note that Solomon's instances contains 100 clients. In the next section, the 25 and the 50 first clients are considered.

Results on 25 and 50 customers instances

The algorithm from [22] optimizes the route cost and use a branch & price to minimize the number of vehicles. Its results are used as reference (column KDMSS). Our algorithm provides two solutions: a first one optimized on the route cost and another one optimized on the number of vehicles. Table 3.5.1 compares the results on the 25 customers instances. Five runs of our algorithm have be done on each instances and the

best result is reported in the Table. Better results are displayed with a plus ("+") sign and worse results with a minus ("−") sign. For those instances, the standard deviation of our approach is around 0.04 % with a maximum of 0.74 %. We obtained all the best solutions when considering the route cost. Some optimal solutions in terms of number of vehicles could not be found. Optimizing on the number of vehicles enables us to improve 4 instances. We next presents the results for 50 and 100 customers.

Table 3.5.1 reports the results for 50 clients. Our approach reaches the same values on the C instances for both the best solution on route cost and the best solution on the number of vehicles. Our two solutions are also identical for the RC instances. They are quite similar to the solutions of [22]. The main difference occurs for the R instances. As for 25 clients, our solutions for R106 instance are very different, with 2 more vehicles. The solution optimizing the number of routes is better for 6 out of 12 instances, with a increased route cost. It is worse for only 1 instance (R106).

Results on 100 customers instances

As in the previous section, our results are compared with [22]. They are reported in the Table 3.5.1. The solutions for the C instances are identical. Differences, in terms of number of vehicles, increase between the solution from [22] and our solutions.

A summary of our results is then compared with the following articles:

- BRM11 : Baldacci, Mingozzi and Roberti [4]. Exact method based on a reduced set partitioning and a branch & cut algorithm.
- JSP08 : Jepsen et al. [21]. Exact method based on branch-and-cut-and-price algorithm.
- CM12 : Cordeau and Maischberger [8]. Metaheuristic based on an iterated parallel tabu search. We took the best results on 10^6 runs.
- NBD10 : Nagata, Braysy and dullaert [29]. Metaheuristic based on a memetic algorithm using an edge assembly crossover.

As for the 25 and 50 instances our algorithm provides two solutions, one is optimized on the cost and another one on the number of vehicles. Once again five runs are done and the better one is kept in the Table 3.5.1. For each method, the mean on the number

	KDMSS		route cost sol.		nb. vehicles sol.	
Instances	Cost	Nb. Veh.	Cost	Nb. Veh.	Cost	Nb. Veh.
C101	191.3	3	191.3	3	191.3	3
C102	190.3	3	190.3	3	190.3	3
C103	190.3	3	190.3	3	190.3	3
C104	186.9	3	186.9	3	186.9	3
C105	191.3	3	191.3	3	191.3	3
C106	191.3	3	191.3	3	191.3	3
C107	191.3	3	191.3	3	191.3	3
C108	191.3	3	191.3	3	191.3	3
C109	191.3	3	191.3	3	191.3	3
avg.	190.59	3.00	190.59	3.00	190.59	3.00
R101	617.1	8	617.1	8	617.1	8
R102	547.1	7	547.1	7	547.1	7
R103	454.6	5	454.6	5	(+)472.3	(-)4
R104	416.9	4	416.9	4	416.9	4
R105	530.5	6	530.5	6	(+)555.6	(-)5
R106	465.4	3	465.4	(+)5	465.4	(+)5
R107	424.3	4	424.3	4	424.3	4
R108	397.3	4	397.3	4	397.3	4
R109	441.3	5	441.3	5	(+)463.4	(-)4
R110	444.1	4	444.1	(+)5	(+)444.7	4
R111	428.8	5	428.8	(-)4	428.8	(-)4
R112	393.0	4	393	4	393	4
avg.	463.37	4.92	463.37	5.08	468.83	4.75
RC101	461.1	4	461.1	4	461.1	4
RC102	351.8	3	(-)351.7	3	(-)351.7	3
RC103	332.8	3	332.8	3	332.8	3
RC104	306.6	3	306.6	3	306.6	3
RC105	411.3	4	(-)411.2	4	(-)411.2	4
RC106	345.5	3	345.5	3	345.5	3
RC107	298.3	3	298.3	3	298.3	3
RC108	294.5	3	294.5	3	294.5	3
avg.	350.24	3.25	350.21	3.25	350.21	3.25

Table 3.I: Results on Solomon instances with 25 clients

	KDMSS		route cost sol.		nb. vehicles sol.	
Instances	Cost	Nb. Veh.	Cost	Nb. Veh.	Cost	Nb. Veh.
C101	362.4	5	362.4	5	362.4	5
C102	361.4	5	361.4	5	361.4	5
C103	361.4	5	361.4	5	361.4	5
C104	358.0	5	358.0	5	358.0	5
C105	362.4	5	362.4	5	362.4	5
C106	362.4	5	362.4	5	362.4	5
C107	362.4	5	362.4	5	362.4	5
C108	362.4	5	362.4	5	362.4	5
C109	362.4	5	362.4	5	362.4	5
avg.	361.69	5.00	361.69	5.00	361.69	5.00
R101	1044	12	(-)1043.8	12	(+)1097.7	(-)11
R102	909.0	11	909.0	11	(+)921.4	(-)10
R103	772.9	9	772.9	9	(+)781.7	(-)8
R104	625.4	6	(+)629.5	6	(+)629.5	6
R105	899.3	9	(-)899.2	9	(-)899.2	9
R106	793.0	5	793.0	(+)8	(+)854.0	(+)7
R107	711.1	7	711.1	7	(+)736.5	(-)6
R108	617.7	6	617.7	6	617.7	6
R109	786.8	8	(-)786.7	8	(+)799.6	(-)7
R110	697.0	7	697.0	7	697.0	7
R111	707.2	7	(+)707.6	7	(+)707.6	7
R112	630.2	6	(+)641.6	6	(+)641.6	(-)6
avg.	766.13	7.75	767.43	8.00	781.96	7.50
RC101	944.0	8	944.0	8	944	8
RC102	822.5	7	(-)822.4	7	(-)822.4	7
RC103	710.9	6	(+)712.6	6	(+)712.6	6
RC104	545.8	5	545.8	5	545.8	5
RC105	855.3	8	855.3	8	855.3	8
RC106	723.2	6	723.2	6	723.2	6
RC107	642.7	6	(+)644.4	6	(+)644.4	6
RC108	598.1	6	598.1	6	598.1	6
avg.	730.31	6.50	730.73	6.50	730.73	6.50

Table 3.II: Results on Solomon instances with 50 clients

	KDMSS		route cost sol.		nb. vehicles sol.	
Instances	Cost	Nb. Veh.	Cost	Nb. Veh.	Cost	Nb. Veh.
C101	828.94	10	828.937	10	828.937	10
C102	828.94	10	828.937	10	828.937	10
C103	828.06	10	828.065	10	828.065	10
C104	824.78	10	824.777	10	824.777	10
C105	828.94	10	828.937	10	828.937	10
C106	828.94	10	828.937	10	828.937	10
C107	828.94	10	828.937	10	828.937	10
C108	828.94	10	828.937	10	828.937	10
C109	828.94	10	828.937	10	828.937	10
avg.	828.38	10.00	828.38	10.00	828.38	10.00
R101	1645.79	19	(-)1643.34	(+)20	(+)1655.23	19
R102	1486.12	17	(-)1472.81	(+)18	(+)1487.31	17
R103	1292.68	13	(-)1213.62	(+)14	(-)1213.62	(+)14
R104	1007.24	9	(-)997.843	(+)11	(+)1030.93	(+)10
R105	1377.11	14	(-)1368.47	(+)15	(-)1371.15	(+)15
R106	1251.98	12	(-)1243.62	(+)13	(+)1268.03	12
R107	1104.66	10	(-)1074.92	(+)11	(-)1074.92	(+)11
R108	960.88	9	(-)954.195	(+)10	(-)954.195	(+)10
R109	1194.73	11	(-)1154.55	(+)12	(-)1154.55	(+)12
R110	1118.59	10	(-)1100.83	(+)12	(-)1100.83	(+)12
R111	1096.72	10	(-)1059.85	(+)12	(-)1070.8	(+)11
R112	982.14	9	(-)972.693	(+)10	(-)972.693	(+)10
avg.	1209.89	11.92	1188.06	13.17	(-)1196.19	12.75
RC101	1696.94	14	(-)1637.04	(+)15	(-)1637.04	(+)15
RC102	1554.75	12	(-)1484.88	(+)14	(-)1485.2 (+)	14
RC103	1261.67	11	(+)1279.14	11	(+)1279.14	11
RC104	1135.48	10	(+)1137.25	10	(+)1137.25	10
RC105	1629.44	13	(-)1518.58	(+)16	(-)1552.05	(+)14
RC106	1424.73	11	(-)1397.05	(+)13	(-)1397.05	(+)13
RC107	1230.48	11	(-)1213.78	(+)12	(+)1258.75	11
RC108	1139.82	10	(+)1141.74	(+)11	(+)1147.88	(+)11
avg.	1384.16	11.50	1351.18	12.75	1361.80	12.38

Table 3.III: Results on Solomon instances with 100 clients

of vehicles and the mean on the route cost are reported. The last line gives the average CPU time to solve one instance. Our solutions are competitive when considering the cost but they use more vehicles. This clearly is an issue which has to be addressed in the future. For the moment, having a better route cost is important because the route duration is the objective function in the TDVRPTW. Thus solutions with more vehicles can be kept, as long as they provide better route costs. Indeed one can imagine a solution using one vehicle in the morning, keeping it at the depot at the middle of the day and then using it again at the evening.

instances	BMR11	JPS08	CM12	NBD10	route cost sol.	nb. vehicles sol.
C1	10.00 828.38	10.00 828.38	10.00 828.38	10.00 828.38	10.00 828.38	10.00 828.38
R1	11.92 1209.89	11.92 1209.89	12 1209.19	11.92 1210.34	13.17 1188.06	12.75 1196.19
RC1	11.5 1384.16	11.5 1384.16	11.5 1385.9	11.5 1384.16	12.75 1351.18	12.38 1361.8
Avg. time (s)	184	12805	2357	300	363.96	363.96

Table 3.IV: Summary on Solomon instances with 100 clients

3.5.2 Results on TDVRPTW

Results for two TDVRPTW instances are reported here. Those instances have been created from the C101 Solomon's instance, respectively with 25 clients 50 clients . Some edges have been removed and a speed profile has been assigned to each remaining arc. The aim is to get a network which has the ability to represent a real network. The five different following scenarios are then used on those networks:

- scenario 0 (Constant scenario) uses the constant speed profile 1, like in classical Solomon instances.
- scenario 1 provides edges differentiation: speed profile 1 is used for the slowest edges (downtown streets), speed profile 2 is for moderate speeds (boulevards, beltways ...) and profile 3 is for the fastest edges (motorways).
- scenario 2 uses speed profile 1 and also considers slowdowns. Speed was considered constant during the day in scenario 1. In scenario 2, slowdowns are set on rush hours. for instance, the interval [421-540] corresponds to the early morning (7 h - 9 h) time period and speed is reduced by about 60 %.

- scenario 3 keeps slowdowns on profile 3 edges and adds hold-ups on edges of profile 3. Hold ups are traduced by a very slow speed, here we have chosen a 0.5 speed ratio. the 0 speed ratio cannot be used because edges from the depot can be on profile 2.
- scenario 4 sets slowdowns on profile 2 and hold-ups on profile 3.

Table 3.5.2 reports the speed ratio for each time interval, depending on the scenario. The time intervals correspond to time in minutes from 0:00am.

		Speed inside time intervals in minutes						
Scenario	Profiles	[0-420]	[421-540]	[541-720]	[721-840]	[841-1020]	[1021-1140]	[1141-1440]
Constant scenario	1	1						
	2							
	3							
Scenario 1 : different profiles	1	1						
	2	1.8						
	3	2.5						
Scenario 2 : slowdown profiles 2 and 3	1	1	1	1	1	1	1	1
	2	1.8	1	1.5	1.3	1.5	1	1.8
	3	2.5	1.5	2	2.2	2	1.5	2.5
Scenario 3 : hold-up profile 2 slowdown profile 3	1	1	1	1	1	1	1	1
	2	1.8	0.5	1.5	0.5	1.5	0.5	1.8
	3	2.5	1.5	2	2.2	2	1.5	2.5
Scenario 4 : slowdown profile 2 hold-up profile 3	1	1	1	1	1	1	1	1
	2	1.8	1	1.5	1.3	1.5	1	1.8
	3	2.5	0.5	2	1	2	0.5	2.5

Table 3.V: scenarios for the speed profiles in our TDVRPTW instances

We will start explaining the C101 instance with 25 customers, speed profiles associated to edges are presented on Figure 3.5.2. Results are presented on Table 3.5.2.

	Constant Scenario	Scenario 1	Scenario 2	Scenario 3	Scenario 4
	nb. vehicles solution				
duration	2793.9	2410.2	2502.3	2559.8	2904.2
distance	294.6	318.6	295.5	383.2	404.2
waiting time	249.3	0	82.9	90.7	310.2
nb. vehicles	3	3	3	3	3
	route duration solution				
duration	2496.2	2383	2396.15	2447.4	2904.2
distance	246.2	250.4	248.3	294.5	404.2
waiting time	0	0	0	0.8	310.2
nb. vehicles	4	4	4	5	3

Table 3.VI: Results on C101 with 25 clients

As can be expected, the travel duration decreases when the speeds increases. When considering the route duration optimization, it is preferable to decrease the waiting

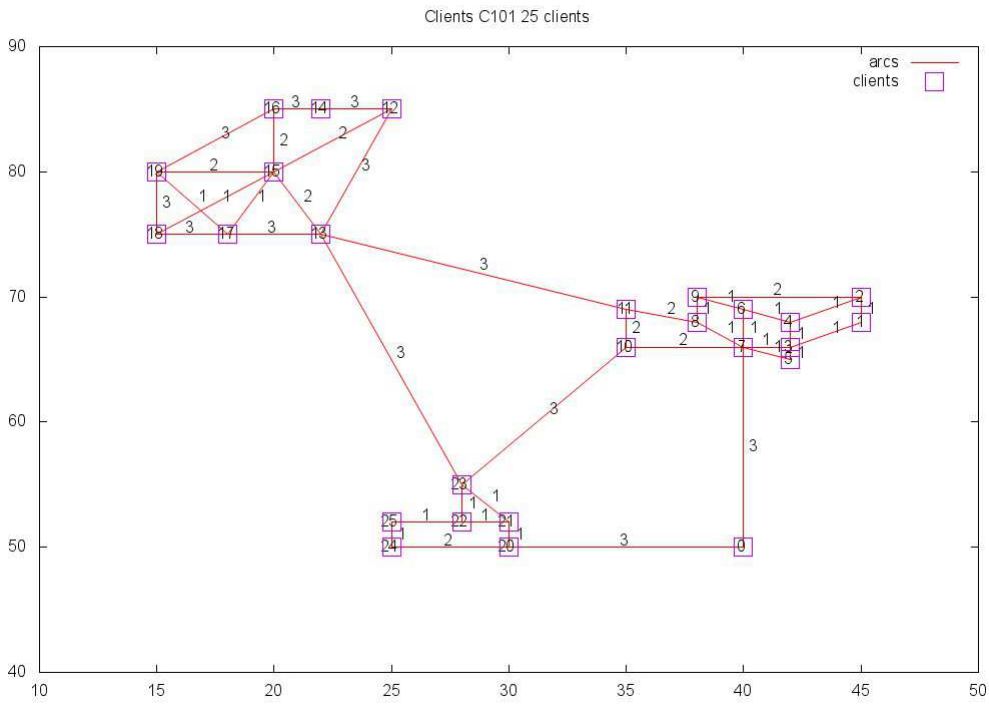

Figure 3.8: Modified instance C101 with 25 customers

time. This explains the intuitive result that some vehicles will be used twice a day but staying at the depot in the middle of the day. Thus those vehicles will not be jammed in the traffic during the periods of high congestion. For instance, the waiting time is 90 for some solutions with scenario 2 and we prefer split the day in two in order to have a vehicle working in the morning and another one in the evening. In the scenario 4 we cannot perform such a split because all edges going out from the depot belong to profile 3, so we have to pay the price anyway.

The second instance contains 50 clients and it has also been built from instance C101. Associated profiles are described on Figure 3.5.2 and results are shown in Table 3.5.2. Once again, the results correspond to what can be expected: when optimizing the travel duration, it is better to cut the waiting time and to use more vehicles.

The results on the five different scenarios are reported on Table 3.5.2.

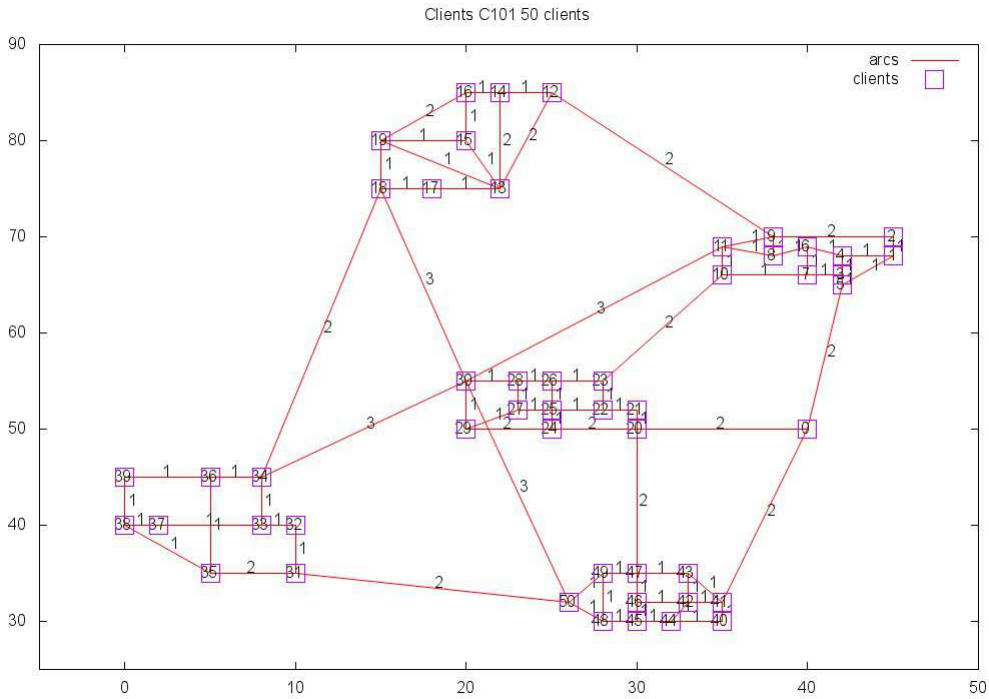

Figure 3.9: Modified instance C101 with 50 customers

	Constant Scenario	Scenario 1	Scenario 2	Scenario 3	Scenario 4
	Nb. Vehicles Objective				
duration	5166.3	4897.1	4944.4	4964.2	4918
distance	647.5	569.9	533.1	547.2	537.1
waiting time	18.8	6.1	46.1	43.3	0
nb. vehicles	6	5	5	5	6
	Duration Objective				
duration	5166.3	4880.4	4905.6	4948.3	4918
distance	647.5	539.9	543.9	546.5	537.1
waiting time	18.8	6.1	0.6	0	0
nb. vehicles	6	6	6	6	6

Table 3.VII: Results on C101 with 50 clients

3.6 Conclusion

In this paper, we have considered an extension of the VRPTW in which the travel time is not constant but changes over the day. This problem, known as the time-dependent VRP (TDVRPTW), as a higher complexity than the original one and a route which is valid at some period of time can become infeasible at other moments. The algorithm we proposed is an approach using a metaheuristic (with VND) and a column generation. Besides, parallelization is used to speed up the process. We first compared our approach with existing results on the VRPTW to measure to quality of the approach. Then, our first results on two instances show our approach is promising. Using time-dependency to model real network is highly important as it corresponds to real-life situation. The last results prove that there is something to earn with this on duration time. Finally, in a near future, we want to apply this algorithm to other problem, the use of parallelized column generation may be a key for next heuristic column generation developpements.

Chapitre 4

Résultats

Dans ce chapitre nous allons brièvement présenter les résultats obtenus. Ceux-ci vont être présentés en deux parties, une première, comparant l'efficacité de notre algorithme sur les problèmes connus de tournées de véhicules avec fenêtres de temps, c'est à dire sans prendre en compte la dépendance au temps. Dans ce premier cas, le coefficient de vitesse est une constance égale à 1, donc la durée de trajet est égale à la distance de trajet. Dans la seconde partie, nous interpréterons quelques résultats obtenus sur nos instances modifiées pour prendre en compte une vitesse variable au cours de la journée.

4.1 VRPTW

Pour valider notre algorithme, nous nous sommes comparés aux résultats de références connus sur les instances de Solomon. Pour les instances de taille 25 et 50 nous nous sommes comparés à la méthode exacte présentée par Kohl et al. [22] que l'on notera KDMSS en référence aux auteurs et co-auteurs. Ces premiers résultats ont été calculés avec une distance mesurée en tronquant au premier chiffre derrière la virgule. La distance entre deux clients est donnée par

$$d = \frac{\left\lfloor 10 * \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 +} \right\rfloor}{10}$$

ou (x_1, y_1) et (x_2, y_2) sont les coordonnées des clients 1 et 2. Cette méthode ne permettait pas de résoudre tous les problèmes de taille 100 en un temps raisonnable, c'est

pourquoi nous ajouterons la comparaison à d'autres algorithmes pour les instances de taille 100. À noter aussi que cette première méthode optimise sur le coût, mais pas sur le nombre de véhicules.
Comme précisé au chapitre 2, dans la section "Génération de Colonnes", notre algorithme est multiobjectif, nous présenterons donc à chaque fois deux tableaux : un premier, montrant nos solutions optimales sur le coût et un second, sur le nombre de tournées.

4.1.1 Résultats 25

Pour réaliser nos résultats, nous avons lancé cinq fois l'algorithme et pris le meilleur résultat obtenu parmi ces cinq lancements.

Les informations dans ce tableau sont à lire de la manière suivante :

- Colonne n°1 : nom de l'instance.
- Colonne n°2 : Meilleur objectif trouvé par KDMSS.
- Colonne n°3 : Meilleur nombre de tournées trouvé par KDMSS.
- Colonne n°4 : Meilleur objectif trouvé par notre méthode.
- Colonne n°5 : Meilleur nombre de tournées par notre méthode.
- Colonne n°6 : Temps de calcul moyen en second pour résoudre une fois l'instance.
- Colonne n°7 : Ecart en pourcentage du coût entre notre méthode et celle de KDMSS.
- Colonne n°8 : Ecart en pourcentage du nombre de tournées entre notre méthode et celle de KDMSS.
- Colonne n°9 : Ecart en pourcentage du nombre de tournées entre notre méthode et le meilleur résultat rencontré au cours de toutes nos expérimentations.
- Colonne n°10 : Ecart en pourcentage du nombre de tournées entre notre méthode et le meilleur résultat rencontré au cours de toutes nos expérimentations.
- Colonne n°11 : Meilleur résultat en terme de coût rencontré au cours de nos expérimentations.
- Colonne n°12 : Meilleur résultat en terme de tournées rencontré au cours de nos expérimentations.

	KDMSS		Notre algorithme			Web		Meilleure Connue		Meilleure Connue			
	Coût	Nb. Veh.	Coût	Nb. Veh.	Temps (s)	Ecart Coût	Ecart Véhicules	Ecart Coût	Ecart Véhicules	Coût	Nb. Veh.		
C101	191.3	3	191.3	3	0.53	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C102	190.3	3	190.3	3	1.01	0.00%	0.00%	0.00%	0.00%	190.300	3	1	
C103	190.3	3	190.3	3	4.87	0.00%	0.00%	0.00%	0.00%	190.300	3	1	
C104	186.9	3	186.9	3	4.19	0.00%	0.00%	0.00%	0.00%	186.900	3	1	
C105	191.3	3	191.3	3	0.55	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C106	191.3	3	191.3	3	0.41	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C107	191.3	3	191.3	3	0.76	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C108	191.3	3	191.3	3	2.97	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C109	191.3	3	191.3	3	3.38	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
MOY	190.59	3.00	190.59	3.00	2.07						0		
											0		
R101	617.1	8	617.1	8	0.36	0.00%	0.00%	0.00%	0.00%	617.100	8	1	
R102	547.1	7	547.1	7	2.78	0.00%	0.00%	0.00%	0.00%	547.100	7	1	
R103	454.6	5	454.6	5	3.48	0.00%	0.00%	0.00%	0.00%	454.600	5	1	
R104	416.9	4	416.9	4	3.24	0.00%	0.00%	0.00%	0.00%	416.900	4	1	
R105	530.5	6	530.5	6	2.76	0.00%	0.00%	0.00%	0.00%	530.500	6	1	
R106	465.4	3	465.4	5	4.83	0.00%	66.67%	0.00%	0.00%	465.400	5	1	
R107	424.3	4	424.3	4	4.31	0.00%	0.00%	0.00%	0.00%	424.300	4	1	
R108	397.3	4	397.3	4	3.31	0.00%	0.00%	0.00%	0.00%	397.300	4	1	
R109	441.3	5	441.3	5	2.27	0.00%	0.00%	0.00%	0.00%	441.300	5	1	
R110	444.1	4	444.1	5	30.40	0.00%	25.00%	0.00%	0.00%	444.100	5	1	
R111	428.8	5	428.8	4	5.57	0.00%	-20.00%	0.00%	0.00%	428.800	4	1	
R112	393	4	393	4	4.00	0.00%	0.00%	0.00%	0.00%	393.000	4	1	
MOY	463.37	4.92	463.37	5.08	5.61								
RC101	461.1	4	461.1	4	10.31	0.00%	0.00%	0.00%	0.00%	461.100	4	1	
RC102	351.8	3	351.7	3	0.88	-0.03%	0.00%	0.00%	0.00%	351.700	3	1	
RC103	332.8	3	332.8	3	1.48	0.00%	0.00%	0.00%	0.00%	332.800	3	1	
RC104	306.6	3	306.6	3	2.07	0.00%	0.00%	0.00%	0.00%	306.600	3	1	
RC105	411.3	4	411.2	4	1.22	-0.02%	0.00%	0.00%	0.00%	411.200	4	1	
RC106	345.5	3	345.5	3	0.82	0.00%	0.00%	0.00%	0.00%	345.500	3	1	
RC107	298.3	3	298.3	3	0.89	0.00%	0.00%	0.00%	0.00%	298.300	3	1	
RC108	294.5	3	294.5	3	1.60	0.00%	0.00%	0.00%	0.00%	294.500	3	1	
MOY	350.24	3.25	350.21	3.25	2.41							29	/29
Somme	10077.60	112.00	10077.40	114.00	105.25								

FIGURE 4.1 – Résultats avec 25 clients avec objectif coût

La dernière ligne fait la somme sur les colonnes et nous permet donc de comparer l'efficacité de notre algorithme. Sur ces premières instances, notre algorithme a fait deux erreurs et une amélioration :

- instance R106 nous obtenons une solution de même coût avec 2 véhicules de plus.
- instance R110 nous obtenons une solution de même coût avec 1 véhicule de plus.
- instance R111 nous obtenons une solution de même coût avec 1 véhicule de moins.

Nous pensons que ces erreurs sont des erreurs de typographie dans l'article KDMSS. Car les solutions sont les mêmes en terme de coût. Peut-être une inversion sur le nombre de véhicules entre l'instance R110 et R111.

Notre algorithme nous fournit aussi des solutions optimisées selon le nombre de tournées :

On obtient une amélioration du nombre de tournées sur trois instances, R103, R105 et R109 mais on perd bien sûr sur le coût. Le but de la recherche opérationnelle, aussi appelée, aide à la décision, est de fournir une solution optimisée au client, mais ce sera lui qui prendra la décision finale. On parle ici d'optimums Pareto, il y a deux objectifs et chacune des solutions proposées est optimale sur un des critères (coût ou nombre de véhicules).

Ces premiers tests sont un succès, si l'on regarde la robustesse de notre algorithme, sur les cinq itérations réalisées pour obtenir ces résultats la moyenne de l'écart type moyen est de 0.04% avec un maximum de 0.74%. Ces résultats sont tout de même à nuancer, car les instances à 25 clients sont les plus simples ...

4.1.2 Résultats 50

La lecture des colonnes est la même qu'à la section précédente. Les résultats en violets dans les deux premières colonnes sont des résultats qui ne sont pas obtenus à partir de l'article KDMSS [22] mais qui viennent de l'article de Irnich et Villeneuve [20]. Cet article présente une méthode exacte utilisant une nouvelle façon d'éliminer les cycles et donc une amélioration du calcul des bornes. On obtient les résultats suivants :

	KDMSS		Notre algorithme			Web		Meilleure Connue		Meilleure Connue			
	Coût	Nb. Veh.	Coût	Nb. Veh.	Temps (s)	Ecart Coût	Ecart Véhicules	Ecart Coût	Ecart Véhicules	Coût	Nb. Veh.		
C101	191.3	3	191.3	3	0.53	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C102	190.3	3	190.3	3	1.01	0.00%	0.00%	0.00%	0.00%	190.300	3	1	
C103	190.3	3	190.3	3	4.87	0.00%	0.00%	0.00%	0.00%	190.300	3	1	
C104	186.9	3	186.9	3	4.20	0.00%	0.00%	0.00%	0.00%	186.900	3	1	
C105	191.3	3	191.3	3	0.55	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C106	191.3	3	191.3	3	0.41	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C107	191.3	3	191.3	3	0.76	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C108	191.3	3	191.3	3	2.97	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
C109	191.3	3	191.3	3	3.38	0.00%	0.00%	0.00%	0.00%	191.300	3	1	
MOY	190.59	3.00	190.59	3.00	2.07								
R101	617.1	8	617.1	8	0.36	0.00%	0.00%	0.00%	0.00%	617.100	8	1	
R102	547.1	7	547.1	7	2.78	0.00%	0.00%	0.00%	0.00%	547.100	7	1	
R103	454.6	5	472.3	4	3.48	3.89%	-20.00%	0.00%	0.00%	472.300	4	1	
R104	416.9	4	416.9	4	3.24	0.00%	0.00%	0.00%	0.00%	416.900	4	1	
R105	530.5	6	555.6	5	2.76	4.73%	-16.67%	0.00%	0.00%	555.600	5	1	
R106	465.4	3	465.4	5	4.83	0.00%	66.67%	0.00%	0.00%	465.400	5	1	
R107	424.3	4	424.3	4	4.31	0.00%	0.00%	0.00%	0.00%	424.300	4	1	
R108	397.3	4	397.3	4	3.31	0.00%	0.00%	0.00%	0.00%	397.300	4	1	
R109	441.3	5	463.4	4	2.27	5.01%	-20.00%	0.00%	0.00%	463.400	4	1	
R110	444.1	4	444.7	4	30.40	0.14%	0.00%	0.00%	0.00%	444.700	4	1	
R111	428.8	5	428.8	4	5.57	0.00%	-20.00%	0.00%	0.00%	428.800	4	1	
R112	393	4	393	4	4.00	0.00%	0.00%	0.00%	0.00%	393.000	4	1	
MOY	463.37	4.92	468.83	4.75	5.61								
RC101	461.1	4	461.1	4	10.31	0.00%	0.00%	0.00%	0.00%	461.100	4	1	
RC102	351.8	3	351.7	3	0.88	-0.03%	0.00%	0.00%	0.00%	351.700	3	1	
RC103	332.8	3	332.8	3	1.48	0.00%	0.00%	0.00%	0.00%	332.800	3	1	
RC104	306.6	3	306.6	3	2.07	0.00%	0.00%	0.00%	0.00%	306.600	3	1	
RC105	411.3	4	411.2	4	1.22	-0.02%	0.00%	0.00%	0.00%	411.200	4	1	
RC106	345.5	3	345.5	3	0.82	0.00%	0.00%	0.00%	0.00%	345.500	3	1	
RC107	298.3	3	298.3	3	0.89	0.00%	0.00%	0.00%	0.00%	298.300	3	1	
RC108	294.5	3	294.5	3	1.60	0.00%	0.00%	0.00%	0.00%	294.500	3	1	
MOY	350.24	3.25	350.21	3.25	2.41							29	/29
	10077.60	112.00	10142.90	110.00	105.25								

FIGURE 4.2 – Résultats avec 25 clients avec objectif tournée

	KDMSS		Notre algorithme			Web		Meilleure Connue		Meilleure Connue			
	Coût	Nb. Veh.	Coût	Nb. Veh.	Temps (s)	Ecart Coût	Ecart Véhicules	Ecart Coût	Ecart Véhicules	Coût	Nb. Veh.		
C101	362.4	5	362.4	5	6.68	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C102	361.4	5	361.4	5	23.03	0.00%	0.00%	0.00%	0.00%	361.4	5	1	
C103	361.4	5	361.4	5	51.57	0.00%	0.00%	0.00%	0.00%	361.4	5	1	
C104	358	5	358	5	75.05	0.00%	0.00%	0.00%	0.00%	358.0	5	1	
C105	362.4	5	362.4	5	4.20	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C106	362.4	5	362.4	5	2.88	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C107	362.4	5	362.4	5	9.98	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C108	362.4	5	362.4	5	15.71	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C109	362.4	5	362.4	5	28.65	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
MOY	361.69	5.00	361.69	5.00	24.20								
R101	1044	12	1043.8	12	7.45	-0.02%	0.00%	0.00%	0.00%	1043.8	12	1	
R102	909	11	909	11	11.34	0.00%	0.00%	0.00%	0.00%	909.0	11	1	
R103	772.9	9	772.9	9	36.33	0.00%	0.00%	0.00%	0.00%	772.9	9	1	
R104	625.4	6	629.5	6	78.56	0.66%	0.00%	0.66%	0.00%	625.4	6	0	
R105	899.3	9	899.2	9	122.52	-0.01%	0.00%	0.00%	0.00%	899.2	9	1	
R106	793	5	793	8	41.63	0.00%	60.00%	0.00%	0.00%	793.0	8	1	
R107	711.1	7	711.1	7	52.80	0.00%	0.00%	0.00%	0.00%	711.1	7	1	
R108	617.7	6	617.7	6	72.57	0.00%	0.00%	0.00%	0.00%	617.7	6	1	
R109	786.8	8	786.7	8	100.92	-0.01%	0.00%	0.00%	0.00%	786.7	8	1	
R110	697	7	697	7	58.64	0.00%	0.00%	0.00%	0.00%	697.0	7	1	
R111	707.2	7	707.6	7	80.02	0.06%	0.00%	0.06%	0.00%	707.2	7	0	
R112	630.2	6	641.6	6	72.69	1.81%	0.00%	1.81%	0.00%	630.2	6	0	
MOY	766.13	7.75	767.43	8.00	61.29								
RC101	944	8	944	8	48.90	0.00%	0.00%	0.00%	0.00%	944.0	8	1	
RC102	822.5	7	822.4	7	204.14	-0.01%	0.00%	0.00%	0.00%	822.4	7	1	
RC103	710.9	6	712.6	6	53.85	0.24%	0.00%	0.24%	0.00%	710.9	6	0	
RC104	545.8	5	545.8	5	17.29	0.00%	0.00%	0.00%	0.00%	545.8	5	1	
RC105	855.3	8	855.3	8	315.61	0.00%	0.00%	0.00%	0.00%	855.3	8	1	
RC106	723.2	6	723.2	6	49.04	0.00%	0.00%	0.00%	0.00%	723.2	6	1	
RC107	642.7	6	644.4	6	56.27	0.26%	0.00%	0.26%	0.00%	642.7	6	0	
RC108	598.1	6	598.1	6	31.72	0.00%	0.00%	0.00%	0.00%	598.1	6	1	
MOY	730.31	6.50	730.73	6.50	97.10							24	/29
	18291.3	190	18310.1	193	1730.07								

FIGURE 4.3 – Résultats avec 50 clients avec objectif coût

On note quatre légers écarts sur l'objectif, les résultats ont déjà été rencontrés dans nos expérimentations, mais n'ont pas été trouvés sur cette batterie de cinq lancements. Sur le nombre de tournées, l'instance R106 est encore problématique avec un écart de trois véhicules.

Les résultats obtenus sur l'optimisation du nombre de tournées sont :

	KDMSS		Notre algorithme			Web		Meilleure Connue		Meilleure Connue			
	Coût	Nb. Veh.	Coût	Nb. Veh.	Temps (s)	Ecart Coût	Ecart Véhicules	Ecart Coût	Ecart Véhicules	Coût	Nb. Veh.		
C101	362.4	5	362.4	5	6.68	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C102	361.4	5	361.4	5	23.03	0.00%	0.00%	0.00%	0.00%	361.4	5	1	
C103	361.4	5	361.4	5	51.57	0.00%	0.00%	0.00%	0.00%	361.4	5	1	
C104	358	5	358	5	75.05	0.00%	0.00%	0.00%	0.00%	358.0	5	1	
C105	362.4	5	362.4	5	4.20	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C106	362.4	5	362.4	5	2.88	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C107	362.4	5	362.4	5	9.98	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C108	362.4	5	362.4	5	15.71	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
C109	362.4	5	362.4	5	28.65	0.00%	0.00%	0.00%	0.00%	362.4	5	1	
MOY	361.69	5.00	361.69	5.00	24.20								
R101	1044	12	1097.7	11	7.45	5.14%	-8.33%	0.00%	0.00%	1097.7	11	1	
R102	909	11	921.4	10	11.34	1.36%	-9.09%	0.00%	0.00%	921.4	10	1	
R103	772.9	9	781.7	8	36.33	1.14%	-11.11%	0.00%	0.00%	781.7	8	1	
R104	625.4	6	629.5	6	78.56	0.66%	0.00%	0.08%	0.00%	629.0	6	0	
R105	899.3	9	899.2	9	122.52	-0.01%	0.00%	0.00%	0.00%	899.2	9	1	
R106	793	5	854	7	41.63	7.69%	40.00%	0.00%	0.00%	854.0	7	1	
R107	711.1	7	736.5	6	52.80	3.57%	-14.29%	0.00%	0.00%	736.5	6	1	
R108	617.7	6	617.7	6	72.57	0.00%	0.00%	0.00%	0.00%	617.7	6	1	
R109	786.8	8	799.6	7	100.92	1.63%	-12.50%	0.00%	0.00%	799.6	7	1	
R110	697	7	697	7	58.64	0.00%	0.00%	0.00%	0.00%	697.0	7	1	
R111	707.2	7	707.6	7	80.02	0.06%	0.00%	0.00%	0.00%	707.6	7	1	
R112	630.2	6	641.6	6	72.69	1.81%	0.00%	0.17%	0.00%	640.5	6	0	
MOY	766.13	7.75	781.96	7.50	61.29								
RC101	944	8	944	8	48.90	0.00%	0.00%	0.00%	0.00%	944.0	8	1	
RC102	822.5	7	822.4	7	204.14	-0.01%	0.00%	0.00%	0.00%	822.4	7	1	
RC103	710.9	6	712.6	6	53.85	0.24%	0.00%	0.24%	0.00%	710.9	6	0	
RC104	545.8	5	545.8	5	17.29	0.00%	0.00%	0.00%	0.00%	545.8	5	1	
RC105	855.3	8	855.3	8	315.61	0.00%	0.00%	0.00%	0.00%	855.3	8	1	
RC106	723.2	6	723.2	6	49.04	0.00%	0.00%	0.00%	0.00%	723.2	6	1	
RC107	642.7	6	644.4	6	56.27	0.26%	0.00%	0.00%	0.00%	644.4	6	1	
RC108	598.1	6	598.1	6	31.72	0.00%	0.00%	0.00%	0.00%	598.1	6	1	
MOY	730.31	6.50	730.73	6.50	97.10							26	/29
	18291.3	190	18484.5	187	60.91								

FIGURE 4.4 – Résultats avec 50 clients avec objectif tournée

On a une amélioration du nombre de véhicules sur cinq instances et l'écart par rapport à la solution optimale connue sur l'instance R106 passe à deux.
Sur les cinq itérations réalisées pour obtenir ces résultats la moyenne de l'écart type moyen est de 0.23% avec un maximum de 1.22%.
Ces résultats sur 50 clients restent très satisfaisants et continuent de valider notre algorithme.

4.1.3 Résultats 100

Résultats 100 : KDMSS & IV

Les résultats de référence ne sont pas complets en raison de la longueur des anciennes méthodes.

	KDMSS		Notre algorithme			Web		Meilleure Connue		Meilleure Connue			
	Coût	Nb. Veh.	Coût	Nb. Veh.	Temps (s)	Ecart Coût	Ecart Véhicules	Ecart Coût	Ecart Véhicules	Coût	Nb. Veh.		
C101	827.3	10	827.3	10	20.79	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C102	827.3	10	827.3	10	93.47	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C103	826.3	10	826.3	10	205.05	0.00%	0.00%	0.00%	0.00%	826.3	10	1	
C104	822.9	10	822.9	10	315.99	0.00%	0.00%	0.00%	0.00%	822.9	10	1	
C105	827.3	10	827.3	10	21.56	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C106	827.3	10	827.3	10	30.44	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C107	827.3	10	827.3	10	36.55	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C108	827.3	10	827.3	10	64.89	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C109	827.3	10	827.3	10	114.53	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
MOY	826.70	10.00	826.70	10.00	100.36								
R101	1637.7	20	1636.7	20	88.83	-0.06%	0.00%	0.00%	0.00%	1636.7	20	1	
R102	1466.6	18	1467.2	18	393.31	0.04%	0.00%	0.00%	0.00%	1467.2	18	1	
R103	1208.7	14	1212.4	14	240.32	0.31%	0.00%	0.00%	0.00%	1212.4	14	1	
R104	971.5	11	976.3	10	596.33	0.49%	-9.09%	0.00%	0.00%	976.3	10	1	
R105	1355.3	15	1360.2	15	162.22	0.36%	0.00%	0.00%	0.00%	1360.2	15	1	
R106	1234.6	13	1236.6	13	313.56	0.16%	0.00%	0.00%	0.00%	1236.6	13	1	
R107	1064.6	11	1072.3	11	318.02	0.72%	0.00%	0.00%	0.00%	1072.3	11	1	
R108			956.2	10	524.85			0.00%	0.00%	956.2	10	1	
R109	1146.9	13	1153.3	13	314.43	0.56%	0.00%	0.00%	0.00%	1153.3	13	1	
R110	1068	12	1084.7	12	387.40	1.56%	0.00%	0.00%	0.00%	1084.7	12	1	
R111	1048.7	12	1050.6	12	411.54	0.18%	0.00%	0.00%	0.00%	1050.6	12	1	
R112			975.8	11	474.41			0.03%	0.00%	975.5	11	0	
MOY	1220.26	13.90	1181.86	13.25	352.10								
RC101	1619.8	15	1638.7	16	279.53	1.17%	6.67%	0.00%	0.00%	1638.7	16	1	
RC102	1457.4	14	1481	15	364.82	1.62%	7.14%	0.37%	7.14%	1475.5	14	0	
RC103	1258	11	1283.7	12	410.21	2.04%	9.09%	0.00%	0.00%	1283.7	12	1	
RC104			1132.4	10	403.69			0.00%	0.00%	1132.4	10	1	
RC105	1513.7	15	1514.1	16	317.29	0.03%	6.67%	0.00%	0.00%	1514.1	16	1	
RC106			1376.5	13	351.61			0.00%	0.00%	1376.5	13	1	
RC107	1207.8	12	1210.2	12	528.94	0.20%	0.00%	0.00%	0.00%	1210.2	12	1	
RC108	1114.2	11	1132.5	11	571.63	1.64%	0.00%	0.00%	0.00%	1132.5	11	1	
MOY	1361.82	13.00	1346.14	13.13	403.46					1345.45	13	27	/29
	27813.8	307	32391.7	354	291.9885854								

FIGURE 4.5 – Résultats avec 100 clients avec objectif coût

Les résultats commencent à descendre un peu en qualité, le coût n'est pas très éloigné, mais quatre des instances RCxxx n'ont pas le bon nombre de véhicules.

Au niveau des résultats obtenus en minimisant sur le nombre de véhicules on obtient des résultats intéressants, neuf instances ont été améliorées sur le nombre de véhicules ! Sur les cinq itérations réalisées pour obtenir ces résultats la moyenne de l'écart type

	KDMSS		Notre algorithme			Web		Meilleure Connue		Meilleure Connue			
	Coût	Nb. Veh.	Coût	Nb. Veh.	Temps (s)	Ecart Coût	Ecart Véhicules	Ecart Coût	Ecart Véhicules	Coût	Nb. Veh.		
C101	827.3	10	827.3	10	20.79	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C102	827.3	10	827.3	10	93.47	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C103	826.3	10	826.3	10	205.05	0.00%	0.00%	0.00%	0.00%	826.3	10	1	
C104	822.9	10	822.9	10	315.99	0.00%	0.00%	0.00%	0.00%	822.9	10	1	
C105	827.3	10	827.3	10	21.56	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C106	827.3	10	827.3	10	30.44	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C107	827.3	10	827.3	10	36.55	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C108	827.3	10	827.3	10	64.89	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
C109	827.3	10	827.3	10	114.53	0.00%	0.00%	0.00%	0.00%	827.3	10	1	
MOY	826.70	10.00	826.70	10.00	100.36								
R101	1637.7	20	1644.7	19	88.83	0.43%	-5.00%	0.00%	0.00%	1644.7	19	1	
R102	1466.6	18	1481.3	17	393.31	1.00%	-5.56%	0.00%	0.00%	1481.3	17	1	
R103	1208.7	14	1212.4	14	240.32	0.31%	0.00%	0.00%	0.00%	1212.4	14	1	
R104	971.5	11	976.3	10	596.33	0.49%	-9.09%	0.00%	0.00%	976.3	10	1	
R105	1355.3	15	1409	14	162.22	3.96%	-6.67%	0.00%	0.00%	1409.0	14	1	
R106	1234.6	13	1258.4	12	313.56	1.93%	-7.69%	0.00%	0.00%	1258.4	12	1	
R107	1064.6	11	1072.3	11	318.02	0.72%	0.00%	0.00%	0.00%	1072.3	11	1	
R108			956.2	10	524.85			0.00%	0.00%	956.2	10	1	
R109	1146.9	13	1168.9	12	314.43	1.92%	-7.69%	0.00%	0.00%	1168.9	12	1	
R110	1068	12	1085.3	11	387.40	1.62%	-8.33%	0.00%	0.00%	1085.3	11	1	
R111	1048.7	12	1065.3	11	411.54	1.58%	-8.33%	0.00%	0.00%	1065.3	11	1	
R112			975.8	11	474.41			0.03%	0.00%	975.5	11	0	
MOY	1220.26	13.90	1192.16	12.67	352.10								
RC101	1619.8	15	1640.2	15	279.53	1.26%	0.00%	0.00%	0.00%	1640.2	15	1	
RC102	1457.4	14	1489	14	364.82	2.17%	0.00%	0.91%	0.00%	1475.5	14	0	
RC103	1258	11	1291.1	11	410.21	2.63%	0.00%	0.00%	0.00%	1291.1	11	1	
RC104			1132.4	10	403.69			0.00%	0.00%	1132.4	10	1	
RC105	1513.7	15	1544.6	14	317.29	2.04%	-6.67%	0.00%	0.00%	1544.6	14	1	
RC106			1393.3	12	351.61			0.00%	0.00%	1393.3	12	1	
RC107	1207.8	12	1210.2	12	528.94	0.20%	0.00%	0.00%	0.00%	1210.2	12	1	
RC108	1114.2	11	1132.5	11	571.63	1.64%	0.00%	0.00%	0.00%	1132.5	11	1	
MOY	1361.82	13.00	1354.16	12.38	403.46							27	/29
	27813.8	307	32579.5	341	291.9885854								

FIGURE 4.6 – Résultats avec 100 clients avec objectif tournée

moyen est de 0.53% avec un maximum de 1.69%.

Les résultats restent convenables.

Résultats 100 : Web

Le site web `http://www.sintef.no/Projectweb/TOP/VRPTW/Solomon-benchmark/` regroupe les meilleurs résultats connus sur les instances de taille 100. Les résultats viennent d'une vingtaine d'articles différents. La distance indiquée semble être la distance réelle et non la distance tronquée, mais il est difficile de savoir si tous ces résultats sont exacts. Nous obtenons les résultats suivants :

	Web		Notre algorithme			Web		Meilleure Connue		Meilleure Connue			
	Coût	Nb. Veh.	Coût	Nb. Veh.	Temps (s)	Ecart Coût	Ecart Véhicules	Ecart Coût	Ecart Véhicules	Coût	Nb. Veh.		
C101	828.94	10	828.937	10	11.74	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C102	828.94	10	828.937	10	114.02	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C103	828.06	10	828.065	10	209.35	0.00%	0.00%	0.00%	0.00%	828.1	10	1	
C104	824.78	10	824.777	10	429.03	0.00%	0.00%	0.00%	0.00%	824.8	10	1	
C105	828.94	10	828.937	10	21.83	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C106	828.94	10	828.937	10	32.86	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C107	828.94	10	828.937	10	50.69	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C108	828.94	10	828.937	10	72.57	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C109	828.94	10	828.937	10	107.72	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
MOY	828.38	10.00	828.38	10.00	116.65	0.00%	0.00%						
R101	1645.79	19	1643.34	20	90.95	-0.15%	5.26%	0.03%	0.00%	1642.9	20	0	
R102	1486.12	17	1472.81	18	123.85	-0.90%	5.88%	0.00%	0.00%	1472.8	18	1	
R103	1292.68	13	1213.62	14	241.42	-6.12%	7.69%	0.00%	0.00%	1213.6	14	1	
R104	1007.24	9	997.843	11	1160.89	-0.93%	22.22%	2.17%	0.00%	976.6	11	0	
R105	1377.11	14	1368.47	15	217.07	-0.63%	7.14%	0.57%	0.00%	1360.8	15	0	
R106	1251.98	12	1243.62	13	476.58	-0.67%	8.33%	0.22%	0.00%	1240.9	13	0	
R107	1104.66	10	1074.92	11	836.27	-2.69%	10.00%	0.00%	0.00%	1074.9	11	1	
R108	960.88	9	954.195	10	466.36	-0.70%	11.11%	0.37%	0.00%	950.7	10	0	
R109	1194.73	11	1154.55	12	285.13	-3.36%	9.09%	4.32%	9.09%	1106.8	11	0	
R110	1118.59	10	1100.83	12	521.62	-1.59%	20.00%	3.77%	0.00%	1060.8	12	0	
R111	1096.72	10	1059.85	12	361.13	-3.36%	20.00%	8.49%	9.09%	976.9	11	0	
R112	982.14	9	972.693	10	552.00	-0.96%	11.11%	0.00%	0.00%	972.7	10	1	
MOY	1209.89	11.92	1188.06	13.17	444.44	-1.80%	10.49%						
RC101	1696.94	14	1637.04	15	250.74	-3.53%	7.14%	0.37%	-6.25%	1631.1	16	0	
RC102	1554.75	12	1484.88	14	345.71	-4.49%	16.67%	0.16%	0.00%	1482.5	14	0	
RC103	1261.67	11	1279.14	11	420.82	1.38%	0.00%	0.09%	-8.33%	1278.0	12	0	
RC104	1135.48	10	1137.25	10	638.18	0.16%	0.00%	0.00%	0.00%	1137.3	10	1	
RC105	1629.44	13	1518.58	16	523.36	-6.80%	23.08%	0.00%	0.00%	1518.6	16	1	
RC106	1424.73	11	1397.05	13	266.16	-1.94%	18.18%	0.00%	0.00%	1397.1	13	1	
RC107	1230.48	11	1213.78	12	408.34	-1.36%	9.09%	0.00%	0.00%	1213.8	12	1	
RC108	1139.82	10	1141.74	11	349.52	0.17%	10.00%	0.02%	0.00%	1141.5	11	0	
MOY	1384.16	11.50	1351.18	12.75	400.35					1349.955	13	17	/29
	28875.70	282.00	28412.29	305.00	9232.47								

FIGURE 4.7 – Résultats avec 100 clients avec objectif coût, distance réelle

	Web		Notre algorithme			Web		Meilleure Connue		Meilleure Connue			
	Coût	Nb. Veh.	Coût	Nb. Veh.	Temps (s)	Ecart Coût	Ecart Véhicules	Ecart Coût	Ecart Véhicules	Coût	Nb. Veh.		
C101	828.94	10	828.937	10	11.74	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C102	828.94	10	828.937	10	114.02	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C103	828.06	10	828.065	10	209.35	0.00%	0.00%	0.00%	0.00%	828.1	10	1	
C104	824.78	10	824.777	10	429.03	0.00%	0.00%	0.00%	0.00%	824.8	10	1	
C105	828.94	10	828.937	10	21.83	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C106	828.94	10	828.937	10	32.86	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C107	828.94	10	828.937	10	50.69	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C108	828.94	10	828.937	10	72.57	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
C109	828.94	10	828.937	10	107.72	0.00%	0.00%	0.00%	0.00%	828.9	10	1	
MOY	828.38	10.00	828.38	10.00	116.65								
R101	1645.79	19	1655.23	19	90.95	0.57%	0.00%	0.27%	0.00%	1650.8	19	0	
R102	1486.12	17	1487.31	17	123.85	0.08%	0.00%	0.08%	0.00%	1486.1	17	0	
R103	1292.68	13	1213.62	14	241.42	-6.12%	7.69%	0.00%	0.00%	1213.6	14	1	
R104	1007.24	9	1030.93	10	1160.90	2.35%	11.11%	0.00%	0.00%	1030.9	10	1	
R105	1377.11	14	1371.15	15	217.07	-0.43%	7.14%	-4.20%	7.14%	1431.2	14	0	
R106	1251.98	12	1268.03	12	476.58	1.28%	0.00%	0.00%	0.00%	1268.0	12	1	
R107	1104.66	10	1074.92	11	836.27	-2.69%	10.00%	0.00%	0.00%	1074.9	11	1	
R108	960.88	9	954.195	10	466.36	-0.70%	11.11%	0.00%	0.00%	954.2	10	1	
R109	1194.73	11	1154.55	12	285.13	-3.36%	9.09%	4.32%	9.09%	1106.8	11	0	
R110	1118.59	10	1100.83	12	521.62	-1.59%	20.00%	-1.62%	9.09%	1118.9	11	0	
R111	1096.72	10	1070.8	11	361.13	-2.36%	10.00%	9.61%	0.00%	976.9	11	0	
R112	982.14	9	972.693	10	552.01	-0.96%	11.11%	0.00%	0.00%	972.7	10	1	
MOY	1209.89	11.92	1196.19	12.75	444.44								
RC101	1696.94	14	1637.04	15	250.74	-3.53%	7.14%	0.00%	0.00%	1637.0	15	1	
RC102	1554.75	12	1485.2	14	345.71	-4.47%	16.67%	0.18%	0.00%	1482.5	14	0	
RC103	1261.67	11	1279.14	11	420.82	1.38%	0.00%	0.00%	0.00%	1279.1	11	1	
RC104	1135.48	10	1137.25	10	638.18	0.16%	0.00%	0.00%	0.00%	1137.3	10	1	
RC105	1629.44	13	1552.05	14	523.36	-4.75%	7.69%	0.00%	0.00%	1552.1	14	1	
RC106	1424.73	11	1397.05	13	266.16	-1.94%	18.18%	0.00%	0.00%	1397.1	13	1	
RC107	1230.48	11	1258.75	11	408.34	2.30%	0.00%	0.00%	0.00%	1258.8	11	1	
RC108	1139.82	10	1147.88	11	349.52	0.71%	10.00%	0.56%	0.00%	1141.5	11	0	
MOY	1384.16	11.50	1361.80	12.38	400.35					1360.65625	12.375	21	/29
	28875.70	282.00	28582.81	298.00	9232.50								

FIGURE 4.8 – Résultats avec 100 clients avec objectif tournée, distance réelle

Que ce soit avec l'objectif tournée ou l'objectif de coût nos résultats restent éloignés sur nombre de véhicules pour les instances Rxxx et RCxxx ... Notre algorithme améliore cependant sur le coût dans la plupart des cas. Notre avis est que nos métaheuristiques manquent de mouvements travaillant à la minimisation du nombre de véhicules.
Sur les cinq itérations réalisées pour obtenir ces résultats la moyenne de l'écart type moyen est de 0.55% avec un maximum de 1.78%.
Nos résultats sont tout de même satisfaisants et valident notre algorithme. Nous allons maintenant présenter quelques résultats sur la version dépendante du temps.

4.2 TDVRPTW

Comme précisé précédemment, il n'existe pas d'instances pour les problèmes TDVRPTW. Nous en avons donc créé à partir des instances de Solomon utilisées pour le VPRTW. Ces instances ont été créées en supprimant des arcs du graphe originel et en assignant aux arcs restants des profils entre 1 et 3. Dans ce mémoire nous présenterons deux instances TDVRPTW, la C101 avec 25 clients et la C101 avec 50 clients. Dans ces deux instances, les clients sont organisés en clusters, il est donc plus facile d'assigner des arcs de manière à imiter un cas réel. Nous allons présenter les résultats sur ces deux instances au travers de 5 scénarios afin de voir l'impact de la dépendance au temps.

4.2.1 Scénarios TDVPRTW

Voici un tableau résumant les scénarios utilisés :

Explications des différents scénarios :

- Le premier scénario (Scénario Constant) utilise une vitesse constante de 1 comme dans les instances classiques de Solomon.
- Le scénario 1 apporte une différenciation des arcs : les arcs de profil 1 sont les plus lents (imitant des rues de centre-ville), ceux de profils 2 sont de vitesse modérée (boulevard, périphérique ...) et ceux de profils 3 sont les plus rapides (autoroute).
- Le scénario 2 reprend les vitesses du scénario 1 en rajoutant des ralentissements. Dans le scénario 1 la vitesse était considérée constante au cours de la journée, dans

		Vitesse par créneau horaire en minutes						
Scénarios	Profils	[0-420]	[421-540]	[541-720]	[721-840]	[841-1020]	[1021-1140]	[1141-1440]
Scénario constant	1	1						
	2							
	3							
Scénario 1 : Différents profils	1	1						
	2	1.8						
	3	2.5						
Scénario 2 : Ralentissements profil 2 et 3	1	1	1	1	1	1	1	1
	2	1.8	1	1.5	1.3	1.5	1	1.8
	3	2.5	1.5	2	2.2	2	1.5	2.5
Scénario 3 : Congestion profil 2 Ralentissement profil 3	1	1	1	1	1	1	1	1
	2	1.8	0.5	1.5	0.5	1.5	0.5	1.8
	3	2.5	1.5	2	2.2	2	1.5	2.5
Scénario 4 : Ralentissement profil 2 Congestion profil 3	1	1	1	1	1	1	1	1
	2	1.8	1	1.5	1.3	1.5	1	1.8
	3	2.5	0.5	2	1	2	0.5	2.5

TABLE 4.I – Scénarios utilisés sur le TDVRPTW

le 2 on provoque des ralentissements aux heures de pointe. Par exemple dans le créneau [421-540] qui correspond à 7h-9h on diminue la vitesse d'environ 60%.

- Le scénario 3 garde les ralentissements sur les arcs de profil 3 mais rajoute de la congestion sur les arcs de type 2. La congestion se traduit par un très faible coefficient de vitesse, ici 0.5. Nous aurions souhaité mettre 0 mais dans la structure de nos instances certains arcs sont nécessaires et mettre une vitesse de 0 empêche toute résolution.
- Le scénario 4 met des ralentissements sur les arcs de type 2 et de la congestion sur les arcs de type 3.

4.2.2 C101 avec 25 clients

On utilise l'instance C101 de Solomon avec 25 clients, les profils associés aux arcs sont décrits sur la Figure 4.2.2.

Voici les résultats obtenus suivants les différents scénarios :

La structure des tournées est décrite sur les Figures 4.2.2 4.2.2 4.2.2 4.2.2et 4.2.2.

Les résultats sont assez intuitifs, lorsque l'on augmente la vitesse, on diminue la durée. Si on n'optimise pas sur le nombre de véhicules, notre algorithme préfère diminuer l'attente en utilisant plus de véhicules. Ceci se traduit en fait par des véhicules ne travaillant qu'une partie de la journée et réalisant de petites tournées. En effet si on

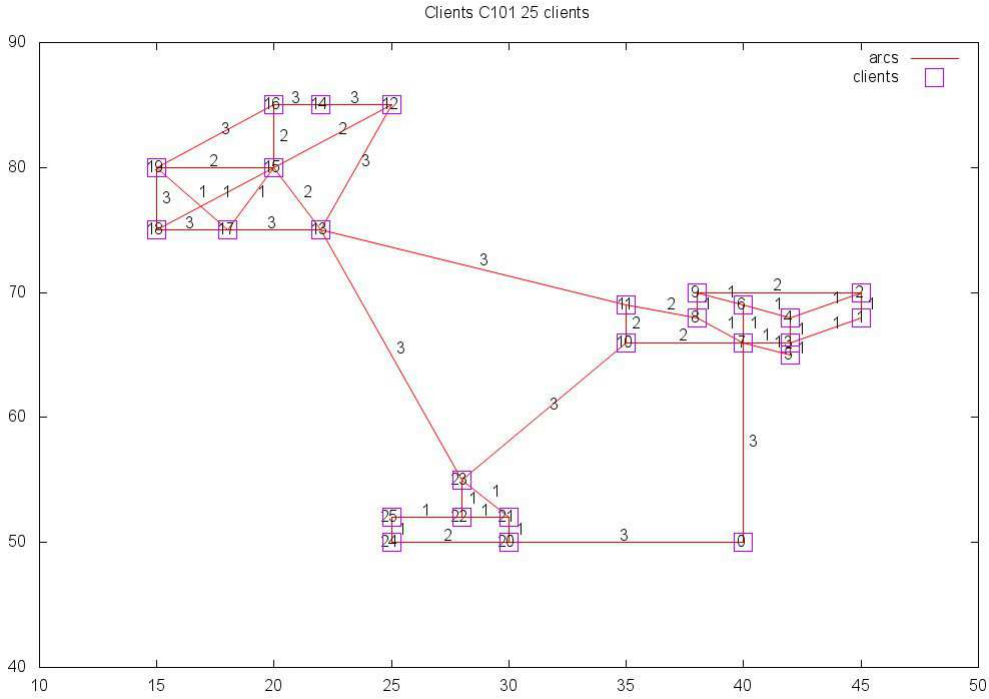

FIGURE 4.9 – Profils sur l'instance C101 à 25 clients

Scénario Constant	Scénario 1	Scénario 2	Scénario 3	Scénario 4
Objectif tournée				
Durée : 2793.9 Distance : 294.6 Attente : 249.3 Nombre de tournées : 3	Durée : 2410.17 Distance : 318.6 Attente : 0 Nombre de tournées : 3	Durée : 2502.26 Distance : 295.5 Attente : 82.8911 Nombre de tournées : 3	Durée : 2559.82 Distance : 383.2 Attente : 90.6622 Nombre de tournées : 3	Durée : 2904.22 Distance : 404.2 Attente : 310.226 Nombre de tournées : 3
Objectif Coût				
Durée : 2496.2 Distance : 246.2 Attente : 0 Nombre de tournées : 4	Durée : 2382.98 Distance : 250.4 Attente : 0 Nombre de tournées : 4	Durée : 2396.15 Distance : 248.3 Attente : 0 Nombre de tournées : 4	Durée : 2447.4 Distance : 294.5 Attente : 0.81 Nombre de tournées : 5	Durée : 2904.22 Distance : 404.2 Attente : 310.226 Nombre de tournées : 3

TABLE 4.II – Résultats TDVRPTW C101 avec 25 clients

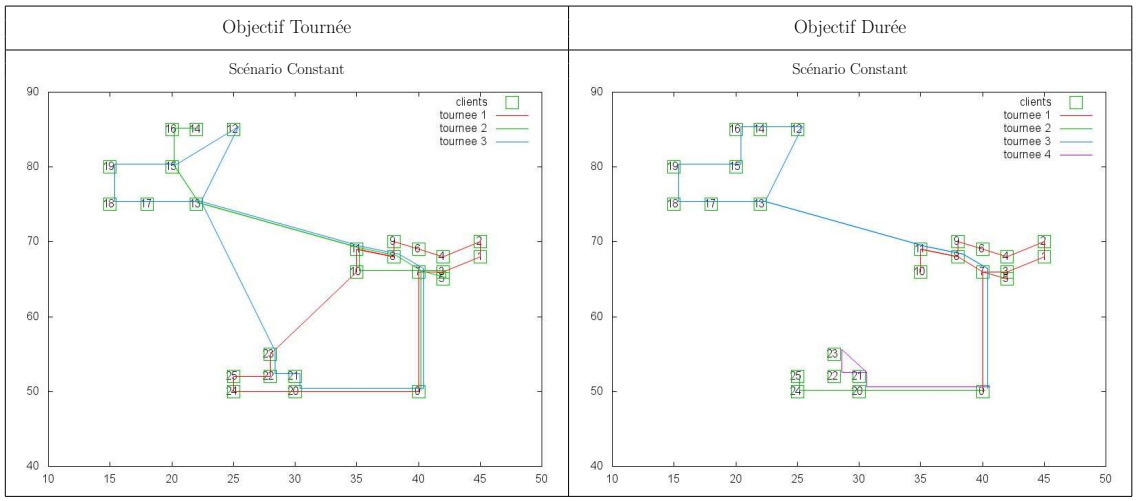

FIGURE 4.10 – Structures des résultats sur le scénario constant

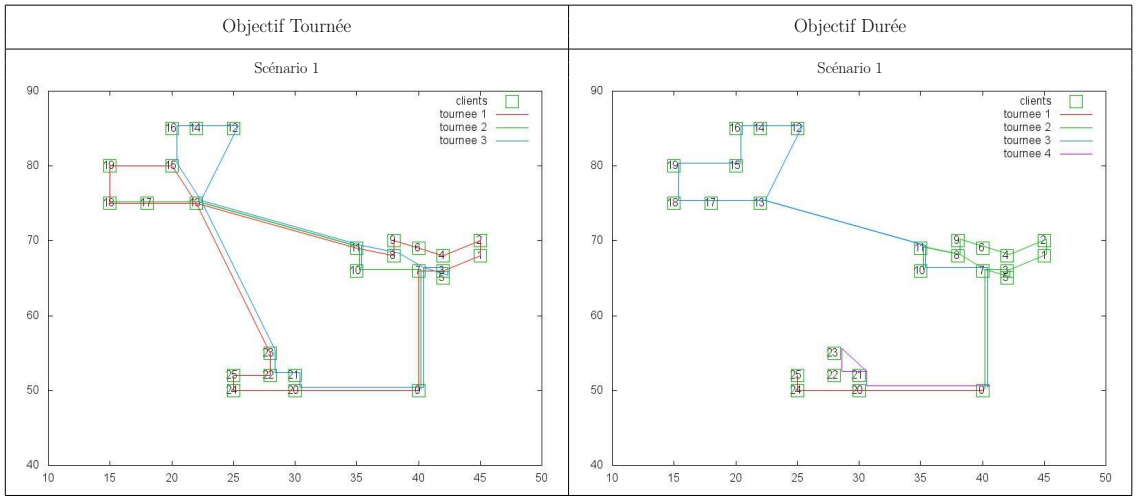

FIGURE 4.11 – Structures des résultats sur le scénario 1

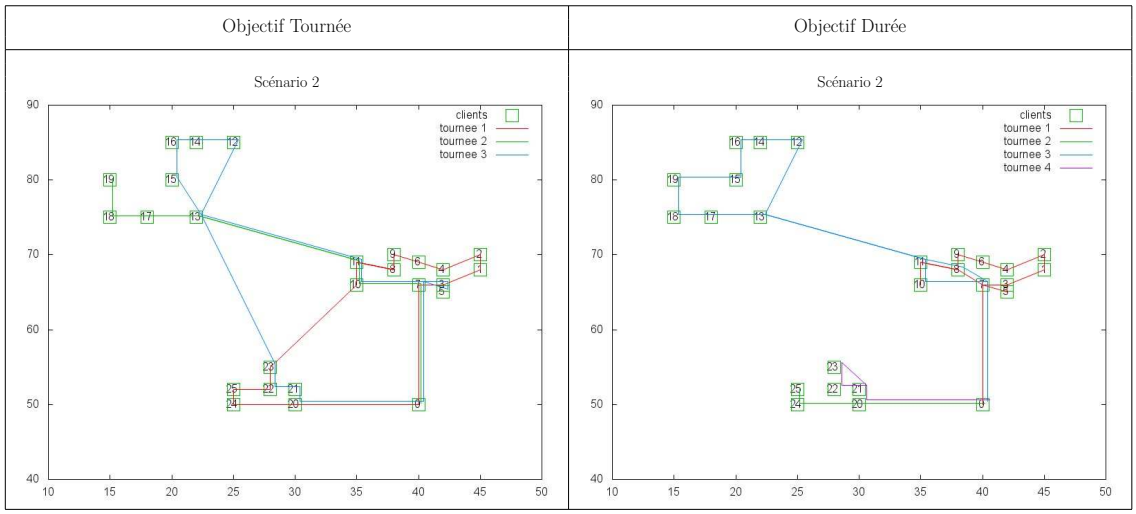

FIGURE 4.12 – Structures des résultats sur le scénario 2

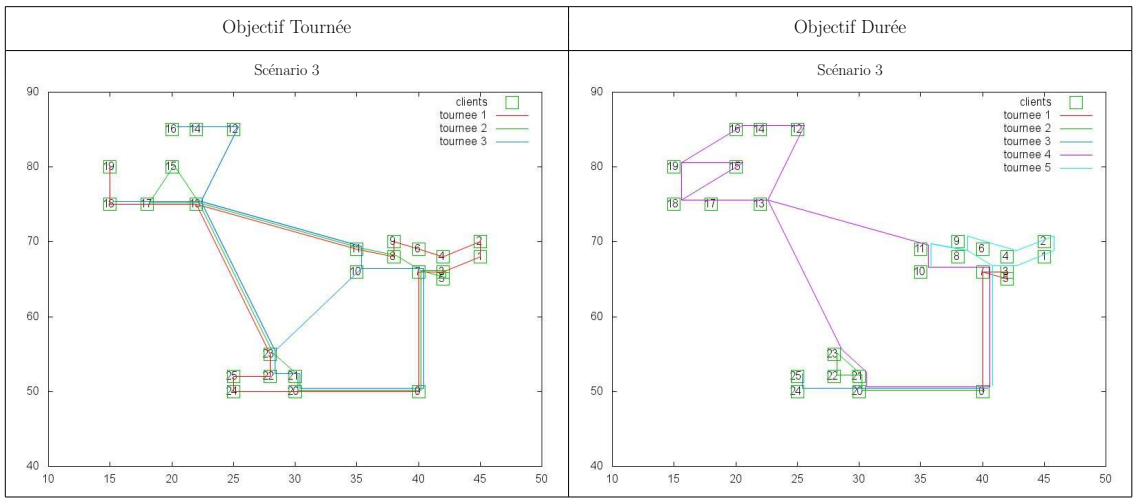

FIGURE 4.13 – Structures des résultats sur le scénario 3

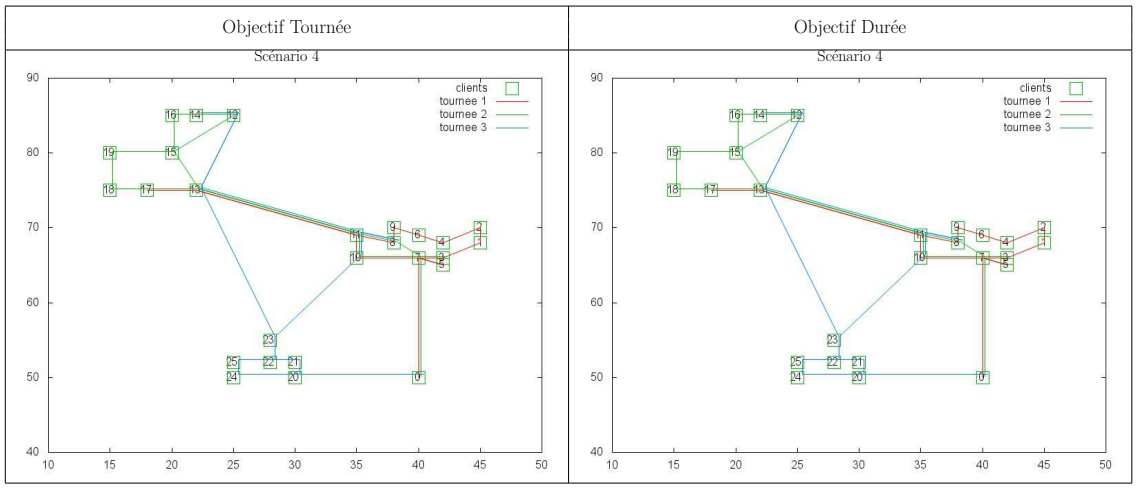

FIGURE 4.14 – Structures des résultats sur le scénario 4

regarde les résultats sur le scénario 2, l'attente est de 90, il est préférable de couper la journée en deux parties afin d'avoir un véhicule fonctionnant le matin et un le soir et d'éviter ainsi le milieu de journée. Dans le scénario 4 on ne peut pas faire cette technique, car tous les arcs sortants du dépôt sont de type 3 et sont congestionnés ce qui nous retarde obligatoirement.

4.2.3 C101 avec 50 clients

On utilise maintenant l'instance C101 de Solomon avec 50 clients, les profils associés aux arcs sont décrits sur la Figure 4.2.3.

Voici les résultats obtenus suivants les différents scénarios :

Scénario Constant	Scénario 1	Scénario 2	Scénario 3	Scénario 4
Objectif tournée				
Durée : 5166.3 Distance : 647.5 Attente : 18.8 Nombre de tournées : 6	Durée : 4897.1 Distance : 569.9 Attente : 6.10222 Nombre de tournées : 5	Durée : 4944.39 Distance : 533.1 Attente : 46.0715 Nombre de tournées : 5	Durée : 4964.18 Distance : 547.2 Attente : 43.4273 Nombre de tournées : 5	Durée : 4917.97 Distance : 537.1 Attente : 0 Nombre de tournées : 6
Objectif Coût				
Durée : 5166.3 Distance : 647.5 Attente : 18.8 Nombre de tournées : 6	Durée : 4880.43 Distance : 539.9 Attente : 6.10222 Nombre de tournées : 6	Durée : 4905.57 Distance : 543.9 Attente : 0.61 Nombre de tournées : 6	Durée : 4948.25 Distance : 546.5 Attente : 0 Nombre de tournées : 6	Durée : 4917.97 Distance : 537.1 Attente : 0 Nombre de tournées : 6

TABLE 4.III – Résultats TDVRPTW C101 avec 50 clients

Les conclusions sont les mêmes que pour 25 clients, ici les arcs sortants du dépôt sont des arcs de profil 2, pour le scénario 4 on n'a donc plus le même phénomène.

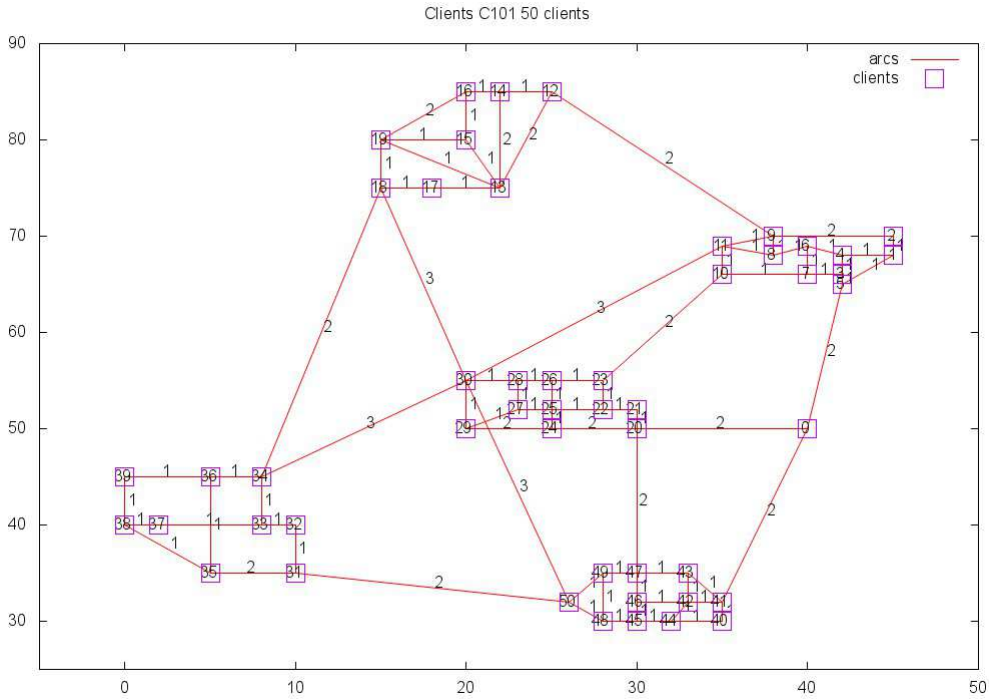

FIGURE 4.15 – Profils sur l'instance C101 à 50 clients

Chapitre 5

Conclusion

5.1 Résumé

Ce mémoire a démarré sur un problème peu abordé en recherche opérationnelle : les tournées de véhicules avec dépendance du temps. Pour aborder ce sujet, il a fallu se renseigner sur les méthodes utilisées pour résoudre les problèmes sans dépendance au temps, mais aussi étudier les recherches en terme d'algorithme de plus court chemin avec dépendance au temps. Ajouter la dépendance au temps augmente la combinatoire du nombre de tournées possibles et rend le modèle non linéaire. C'est pourquoi nous avons essayé d'allier métaheuristique (VND), permettant de résoudre des problèmes non linéaires, et générations de colonnes, permettant la résolution de problèmes avec beaucoup de variables. Nous y avons ajouté la parallélisation qui permet d'avoir plusieurs objectifs et plusieurs façons de générer les colonnes. Les résultats obtenus sont encourageants et les limites qu'ils posent montrent plus une déficience dans les voisinages utilisés qu'une remise en cause de la méthode. Nous sommes partis d'un problème compliqué, peu étudié et avons su combiner les différentes méthodes existentes tout en ajoutant de nouvelles idées.

5.2 Recherche future

C'est un très gros projet touchant à plusieurs domaines de l'optimisation et mêlant plusieurs techniques de recherche opérationnelle, mais aussi de développement infor-

matique. La structure de l'algorithme parallélisé de génération de colonnes est très intéressante et on peut y voir de nombreuses applications. L'absence d'instances pour le TDVRPTW démontre que ce sujet manque encore de formalisme et nous souhaitons d'abord fournir des instances de TDVRPTW afin de pouvoir comparer nos résultats avec d'autres méthodes. Nous souhaitons aussi améliorer les résultats obtenus en optimisant sur le nombre de véhicules, cette amélioration nécessite l'utilisation de nouveaux voisinages. Nous prévoyons de remodeler l'article présenté de ce mémoire afin de prendre en compte ces remarques.

Bibliographie

[1] Soojung Jung Ali Haghani. A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research*, 32 :2959 – 2986, 2005.

[2] Walter Ukovich Ana Osvald, Raffaele Pesenti. Solution methods for a decision support system for vehicle routing with fluctuations in the travel time. *Proceedings*, 2004.

[3] Miguel Andres Figliozzi. The time dependent vehicle routing problem with time windows : Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E : Logistics and Transportation Review*, 48(3) :616–636, May 2012.

[4] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1) :1 – 6, 2012.

[5] S.R. Balseiro, I. Loiseau, and J. Ramonet. An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers & Operations Research*, 38(6) :954 – 966, 2011.

[6] Jack Brimberg, P. Hansen, Pierre Hansen, N. Mladenovic, and Eric D. Taillard. Improvements and comparison of heuristics for solving the multisource weber problem. *Operations Research*, 48 :444–460, 1997.

[7] Lucas Moreira Guedes Christophe Duhamel, Andrea Cynthia Santos. Models and hybrid methods for the onshore wells maintenance problem. *Computers & Operations Research*, 2012.

[8] Jean-François Cordeau and Mirko Maischberger. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & OR*, 39(9) :2033–2050, 2012.

[9] G. B. Dantzig and R. H. Ramser. The truck dispatching problem. *Management Science*, 6, 1959.

[10] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8 :101–111, 1960.

[11] Eulalia Martinez David Soler, Jose Albiach. A way to optimally solve a time-dependent vehicle routing problem with time windows. *OperationsResearch Letters*, 37 :37–42, 2009.

[12] Daniel Delling and Giacomo Nannicini. Bidirectional core-based routing in dynamic time-dependent road networks. *LNCS : Lecture Notes in Computer Science*, 5369 :812–823, 2008.

[13] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.*, 40(2) :342–354, March 1992.

[14] Alberto V. Donati, Luca Maria Gambardella, Andrea E. Rizzoli, Norman Casagrande, and Roberto Montemanni. Time dependent vehicle routing problem with an ant colony system. *European Journal of Operational Research*, 185 :1174 – 1191, 2008.

[15] Bernhard Fleischmann, Martin Gietz, and Stefan Gnutzmann. Time-varying travel times in vehicle routing. *Transportation Science*, 38(2) :160–173, May 2004.

[16] B. Golden, S. Raghavan, and E. Wasil. *Vehicle Routing Problem : Latest Advances and New Challenges*, volume 43 of *Operations Research Computer Science Interfaces*. Springer, New York, United States, 2008.

[17] P. Hansen, N. Mladenović, and J. Moreno Pérez. Variable Neighbourhood Search : Methods and Applications. *4OR : A Quarterly Journal of Operations Research*, 6 :319–360, 2008.

[18] Toshihide Ibaraki Hideki Hashimoto, Mutsunori Yagiura. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5 :434–456, 2008.

[19] Arthur V. Hill and Benton. Modeling intra-city time-dependent travel speeds for vehicle scheduling problems. *Journal of Operational Research Society*, 43 :343–351, 1992.

[20] Stefan Irnich and Daniel Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for k &#8805 ; 3. *INFORMS J. on Computing*, 18(3) :391–406, January 2006.

[21] Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Oper. Res.*, 56(2) :497–511, March 2008.

[22] Niklas Kohl, Jacques Desrosiers, Oli B. G. Madsen, Marius M. Solomon, and François Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1) :101–116, 1999.

[23] Ekkehard Kohler, Rolf H. Mohring, , and Heiko Schilling. Acceleration of shortest path and constrained shortest path computation. *LNCS : Lecture Notes in Computer Science*, 3503 :126–138, 2005.

[24] A. L. Kok, E. W. Hans, and J. M. J. Schutten. Vehicle routing under time-dependent travel times : The impact of congestion avoidance. *Computers & Operations Research*, 39(5) :910–918, 2012.

[25] Yiyo Kuo. Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering*, 59(1) :157 – 165, 2010.

[26] C. Malandraki. *Time Dependent Vehicle Routing Problems : Formulations, Solution Algorithms and Computational Experiments*. University Microfilms Int., 1989.

[27] C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems : formulations, properties and heuristic algorithms. *Transportation Science*, 26 :185–200, 1992.

[28] C. Malandraki and M.S. Daskin. Time-Dependent Vehicle-Routing Problems - Formulations, Properties and Heuristic Algorithms. *Transportation Science*, 26(3) :185–200, 1992.

[29] Yuichi Nagata, Olli Braysy, and Wout Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4) :724 – 737, 2010.

[30] Giacomo Nannicini. Point- to- point sh or test path son dynam ic tim e- dep endent road networks. Master's thesis, Ecole polytechnique, 2009.

[31] Giacomo Nannicini, Philippe Baptiste, Daniel Krob, and Leo Liberti. Fast computation of point-to-point paths on time-dependent road networks. *LNCS : Lecture Notes in Computer Science*, 5165 :225–234, 2008.

[32] Giacomo Nannicini, Daniel Delling, Leo Liberti, and Dominik Schultes. Bidirectional a* search for t ime- dep endent fast paths. *LNCS : Lecture Notes in Computer Science*, 5038 :334–346, 2008.

[33] Eric Prescott-Gagnon. *Methodes Hybrides basees sur la genetion de colonnes pour des probemes de tournees de vehicules avec fenetres de temps.* PhD thesis, Ecole polytechnique de Montreal, 2011.

[34] Peter Sanders and Dominik Schultes. Engineering highway hierarchies. In *Proceedings of the 14th conference on Annual European Symposium - Volume 14*, ESA'06, pages 804–816, London, UK, UK, 2006. Springer-Verlag.

[35] Peter Sanders, Dominik Schultes, Frank Schulz, and Dorothea Wagner. Fast computation of distance tables using highway hierarchies. Technical report, 14th European Symposium on Algorithms (ESA), 2006.

[36] Jean-Yves Potvin Soumia Ichoua, Michel Gendreau. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144 :379–396, 2003.

[37] Kiseok Sung, Michael G.H Bell, Myeongki Seong, and Soondal Park. Shortest paths in a network with time-dependent flow speeds. *European Journal of Operational Research*, 121(1) :32 – 39, 2000.

[38] Éric D. Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2) :170–186, 1997.

[39] P. Toth and D. Vigo. *The Vehicle Routing Problem.* Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2002.

[40] Gilles Goncalves Xin Zhao and Remy Dupas. A genetic approach to solving the vehicle routing problem with time-dependent travel times. In *7e Conference Francophone de MOdelisation et SIMulation - MOSIM*, pages –, 2008.