Problèmes de transport à la demande avec prise en compte de la qualité de service

Maxime Chassaing

N° d'ordre : D. U : 2631
E D S P I C : 727

UNIVERSITE BLAISE PASCAL - CLERMONT II

ECOLE DOCTORALE

SCIENCES POUR L'INGENIEUR DE CLERMONT-FERRAND

T h è s e

Présentée par

MAXIME CHASSAING

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

SPECIALITE : INFORMATIQUE

Titre de la thèse :

Problèmes de transport à la demande avec prise en compte de la qualité de service

Soutenue publiquement le 04 décembre 2015 devant le jury :

M. Dominique FEILLET	Président
Mme Marie-José HUGUET	Rapporteur
M. Aziz MOUKRIM	Rapporteur
Mme Caroline PRODHON	Rapporteur
M. Alain QUILLIOT	Examinateur
M. Christian LAFOREST	Examinateur
M. Gérard FLEURY	Invité
M. Christophe DUHAMEL	Directeur de thèse
M. Philippe LACOMME	Directeur de thèse

Remerciements

J’aimerais tout d’abord remercier mes directeurs de thèse, Christophe Duhamel et Philippe Lacomme, pour avoir été toujours disponibles tout au long de ce travail de recherche et pour leur encadrement exemplaire.

Je remercie Gérard Fleury pour m'avoir aidé sur les probabilités, puis pour m'avoir guidé, encouragé, conseillé notamment sur mon dernier chapitre.

Je tiens à remercier les rapporteurs et membres de mon jury. Caroline Prodhon , Marie-José Huguet et Aziz Moukrim qui ont accepté d'être les rapporteurs de cette thèse, de même que pour leur participation au Jury. Ils ont également contribué par leurs nombreuses remarques et suggestions à améliorer la qualité de ce mémoire, et je leur en suis très reconnaissant. Je remercie particulièrement Caroline Prodhon pour les discussions et échanges que nous avons eu pendant les conférences où nous nous sommes croisés.

Le professeur Dominique Feillet, le professeur Christian Laforest, et le professeur Alain Quilliot m'ont fait l'honneur de participer au Jury de soutenance; je les en remercie profondément.

Je remercie infiniment Christophe Duhamel, Gérard Fleury, Matthieu Gondran, Philippe Lacomme, Damien Lamy, Ren Libo, Nikolay Tchernev, Benjamin Vincent et Jinhua Zhao pour leurs relectures méticuleuses de chacun des chapitres, sans qui cette thèse n'aurait jamais pu voir le jour.

Je remercie mes amis et collègues Alexandre, Benjamin(×10), Cyril, Damien, Djelloul, Marie, Marina, Jason, Pascaline, Jinhua, Luc, Sahar, Vanel, Calvin, Nicolas qui ont fait de cette thèse trois années vraiment agréables, et particulièrement Salsabil pour avoir partagé mon bureau et m'avoir supporté.

Je souhaite remercie aussi tout le personnel du LIMOS pour leur professionnalisme et leur réactivité.

Je passe ensuite une dédicace spéciale à tous les jeunes gens que j'ai eu le plaisir de côtoyer durant ces années à Clermont Ferrand, notamment comme enseignant, et je leur souhaite plein de réussite dans l'avenir.

Pour terminer je remercie ma famille, pour leur soutien et leur support inconditionnel, avec une pensée particulière à Jérémy et Claire à qui je souhaite tous mes vœux de bonheur.

Ce travail a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre du programme Investissements d'avenir dans le cadre du projet LabEx IMobS3 (ANR-10-LABX-16-01), d'une aide de l'Union Européenne au titre du Programme Compétitivité Régionale et Emploi 2007-2013 (FEDER – Région Auvergne) et d'une aide de la Région Auvergne

Ce travail est cofinancé par l'Union européenne. L'Europe s'engage en Auvergne avec le Fonds européen de développement régional.

Résumé

Cette thèse porte sur la modélisation et la résolution de différents problèmes de tournées de véhicules et plus particulièrement sur des problèmes de transport de personnes. Ces problèmes, demandent, entre autre, de respecter une qualité de service minimale pour les solutions proposées. Pour résoudre ces problèmes, plusieurs méthodes d'optimisation de type métaheuristique sont proposées pour obtenir des solutions de bonne qualité dans des temps raisonnables.

Trois problèmes sont traités successivement : le DARP, le TDVRP, le SDARP.

Le premier est un problème de transport à la demande (DARP - *Dial-A-Ride Problem*) qui est le problème de transport de personnes le plus connu de la littérature. Il est proposé dans ce chapitre une méthode de type ELS qui a été comparée aux meilleures méthodes publiées. Les tests montrent que la méthode ELS est compétitive en termes de temps de calcul et de qualité des résultats.

Le deuxième problème est une extension du problème de tournées de véhicules (VRP - *Vehicle Routing Problem*) dans lequel les temps de trajet entre les sommets varient au cours de la journée (TDVRP - *Time Dependent Vehicle Routing Problem*). Dans ce problème, une distinction existe entre les temps de conduite et les temps de travail des chauffeurs. La différence entre les deux correspond aux temps de pause. Ils sont utilisés ici durant les tournées pour éviter aux chauffeurs de conduire durant les périodes à fort ralentissement du trafic. La méthode proposée permet entre autre de positionner stratégiquement ces pauses afin de réduire le temps de conduite et de proposer de nouvelles solutions.

Le dernier problème traité concerne la résolution d'un DARP stochastique. Dans ce problème, les temps de trajet entre les clients ne sont plus déterministes, et ils sont modélisés par une loi de probabilité. L'objectif est de déterminer des solutions robustes aux fluctuations des temps de trajets sur les arcs. Une première approche a permis de calculer des solutions robustes qui ont une probabilité importante d'être réalisables, une seconde approche a permis de générer un ensemble de solutions offrant un équilibre entre la robustesse et le coût.

Mots-clés : transport à la demande, heuristiques, métaheuristiques.

Abstract

In this thesis, we are interested in modeling and solving various vehicle routing problems (VRP), especially passenger transportation problems. These problems aim at finding solutions which guarantee a required quality of service. Several metaheuristics are proposed to obtain high quality solutions within reasonable time. Three problems are addressed: the Dial-A-Ride Problem (DARP), the Time-Dependent Vehicle Routing Problem (TDVRP) and the Stochastic DARP (SDARP).

The DARP is a well-known on-demand transportation problem. We propose an Evolutionary Local Search (ELS) method. It relies on a new randomized constructive heuristic and on adaptive probabilities for selecting neighborhood structures. This approach is compared with existing methods on classical instances. Results show the interest of the proposed method.

The TDVRP is an extension of VRP in which the transportation time varies throughout the day. The driving time is separated from the drivers working time and the difference corresponds to the resting time. The resting time is used to avoid driving during highly congested periods. The proposed method set these resting times in order to reduce the driving time. Hence new solutions avoiding congestion as much as possible are proposed.

In the SDARP, the travel time between clients is stochastic and thus follows a probability distribution. The objective is to compute robust solutions, *i.e.* solutions which handle variations of the transportation time. Two approaches are proposed for this problem. The first one produces robust solutions that have a significant probability of staying feasible. The second one generates a set of compromise solutions, balancing the robustness and the cost.

Keywords: Dial-a-Ride, complexity, heuristic, metaheuristics, DARP

Table des matières

Introduction

Les problèmes de transport jouent un rôle très important dans de nombreux secteurs d'activités. On peut considérer que résoudre un problème de transport consiste à organiser dans le temps et dans l'espace un ensemble de tournées affectées à divers véhicules. Ce type de problème est rencontré, par exemple, dans les chaînes logistiques de grands groupes industriels, dans la grande distribution, dans les systèmes de transports publics (métro, bus…) ou privés (Uber, Blablacar…). Les problèmes se rencontrent aussi bien dans le cadre des services à la personne (transport de personnes à mobilité réduite, transport de personnes âgées, transport scolaire…) que dans le cadre du transport de marchandises (livraison de colis, collecte de déchets ménagers…).

Les problèmes de tournées sont des problèmes combinatoires, c'est-à-dire des problèmes pour lesquels il existe un grand nombre de solutions qu'il est, en général, impossible de parcourir en totalité. Au cours de cette thèse, plusieurs problèmes de tournées sont traités, avec des approches essentiellement heuristiques. Ces méthodes ne garantissent pas l'optimalité de la solution mais permettent de fournir une solution de bonne qualité avec des temps de calcul raisonnables.

Cette thèse se compose de quatre chapitres.

Le premier chapitre présente des définitions générales sur les problèmes de tournées et introduit les notions d'algorithmique et de complexité. Il a pour objectif de donner les bases nécessaires à la compréhension des problèmes rencontrés lors de l'optimisation des tournées. Ce chapitre définit les différentes classes des problèmes de tournées et apporte les notions fondamentales sur leur complexité. Il présente également les principales heuristiques et métaheuristiques utilisées dans la littérature.

Le second chapitre s'intéresse au problème du transport à la demande (*DARP: Dial-a-Ride problem*). Le DARP est un problème qui a été défini initialement en 1978 (Stein, 1978) et pour lequel il est retenu ici la définition de *(Cordeau and Laporte, 2003*). Dans ce chapitre sont présentées différentes procédures qui sont les points clés d'une méthode de résolution de type Evolutionary Local Search (ELS). Cette métaheuristique est définie pour apporter une nouvelle méthode de résolution efficace. Parmi les procédures nécessaires à la définition de cette méthode, une attention particulière est portée à la fonction d'évaluation ainsi qu'à la fonction de recherche locale qui est combinée à un processus d'apprentissage. Cette méthode de résolution est comparée avec les autres méthodes de la littérature sur les instances classiques afin de prouver qu'elle est performante par rapport aux méthodes déjà publiées.

Le troisième chapitre s'intéresse aux problèmes de tournées dans lesquels on distingue le temps de conduite, le temps de travail et les temps de pause. Cette distinction amène à déterminer des solutions qui peuvent, selon les cas, soit favoriser une solution avec un petit temps de transport mais un temps de travail important (le chauffeur a alors passé beaucoup de temps à attendre) soit l'inverse. La détermination de ces solutions oblige à définir une nouvelle approche de résolution qui est introduite dans cette partie.

Le quatrième chapitre aborde la résolution du Stochastic DARP c'est-à-dire un problème de type DARP dans lequel les temps de transport ne sont plus déterministes mais sont des réalisations de variables aléatoires. La résolution de ce problème est spécifique car en choisissant des dates de départs particulières, on peut favoriser l'obtention de solutions plus ou

moins robustes c'est-à-dire des solutions qui ont de grandes chances de rester faisables lors de leur mise en œuvre.

La conclusion reprend les apports de la thèse et dégage les perspectives de recherche.

Dans cette thèse, les problèmes sont identifiés préférentiellement avec leur nom anglais et leurs abréviations anglaises. Lorsqu'il existe un équivalent français, celui-ci est rappelé.

Les travaux présentés dans cette thèse ont fait l'objet des publications suivantes, dont certaines concernent des problèmes d'ordonnancement, et correspondent à des travaux de recherche initiés pendant mon stage de master 2 et ont été terminés au début de la première année de thèse.

Durant la thèse :

Chassaing, M., Duhamel, C., Lacomme, P., 2015. An ELS-based approach with dynamic probabilities management in local search for the Dial-A-Ride Problem. Engineering Applications of Artificial Intelligence, 48,119-133.

Chassaing, M., Duhamel, C., Lacomme, P., 2015. Time-Dependent Vehicle Routing Problem with waiting times. Sixth International Workshop on Transportation and Logistics (ODYSSEUS), Ajaccio, France, Palais des Congrès. 31 Mai-5 Juin 2015

Chassaing, M., Duhamel, C., Lacomme, P., 2015. Tournées de véhicules avec temps de trajet dépendant du temps. Congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Marseille, France, 25-27 février 2015.

Chassaing, M., Lacomme, P., Laforest, C., 2014. Résolution du DARP avec un schéma d'optimisation de type ELS. Congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Bordeaux, France, 26-28 février 2014.

Pendant le stage de master 2 et le début de la thèse :

Chassaing, M., Fontanel, J., Lacomme, P., Ren L., Tchernev, N., Villechon, P., 2014. A GraspxELS approach for the job-shop with a web services paradigm packaging. Expert Systems with Applications, 41, 544-562. doi:10.1016/j.eswa.2013.07.080

Chassaing, M., Huguet, M.-J., Lacomme, P., Tchernev, N., 2013. Application of data mining technology for analyzing Job-Shop solutions. 5th International Conference on Industrial Engineering and System Management. Modeling, Optimisaiton and Simulation in Research and Industrial Applications, Rabat, Maroc, 28-30 Octobre 2013.

Chassaing, M., Fontanel, J., Lacomme, P., Ren, L., Tchernev, N., Villechenon P., 2013. Web Services definition for operational research problems. 14e congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Troyes, France, 13-15 février 2013.

Chassaing, M., Fontanel, J., Lacomme, P., Ren, L., 2013. A GRASP×ELS approach for the Job Shop with generic time-lags and new statistical determination of the parameters. 14e congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Troyes, France, 13-15 février 2013.

CHAPITRE 1

Contexte général de l'étude

Objectifs du chapitre :

Le sujet principal de cette thèse est le transport à la demande. C'est un problème de la famille des problèmes de tournées de véhicules qui consiste à transporter un ensemble de personnes de leur sommet d'origine vers leur sommet de destination. Ce chapitre contient une description des principaux problèmes appartenant à la famille des problèmes de tournées de véhicules, ainsi que des méthodes de résolution qui sont utilisées pour résoudre ces problèmes.

1.1 Les tournées de véhicules

Les problèmes de tournées de véhicules (VRP) rassemblent de nombreux problèmes. Ils consistent à déterminer le trajet d'une flotte de véhicules afin de servir un ensemble de clients. Les différents trajets doivent commencer et finir dans un lieu nommé dépôt. Celui-ci est généralement unique.

Parmi les problèmes de tournées de véhicules, le plus connu est le problème du voyageur de commerce (TSP - *Traveling Salesman Problem*). Proposée avant 1950, la référence la plus ancienne est celle de (Robinson, 1949). Ce problème consiste à rechercher la tournée que doit réaliser un voyageur de commerce souhaitant visiter chaque client, tout en minimisant la distance totale. Pour ce problème, comme pour beaucoup de problèmes de tournées de véhicules, les graphes sont un outil essentiel de modélisation. Une tournée optimale du TSP correspond à un cycle hamiltonien de valeur minimale dans le graphe complet $G = (V, A)$, où chaque sommet $v \in V$ correspond à un client, et où les arcs $(i, j) \in A$ correspondent à un chemin pour aller du client i au client j. Le poids c_{ij} de l'arc $(i, j) \in A$ représente généralement la plus courte distance du client i au client j. Ce problème, bien que simple à énoncer, est *NP-difficile*.

Les problèmes de tournées de véhicules (VRP – *Vehicle Routing Problem*) généralisent le TSP en introduisant une flotte de véhicules. La définition de la classe des problèmes de VRP n'a pas toujours été unique dans la littérature. La première définition a été introduite par (Dantzig and Ramser, 1959). Les problèmes de VRP comportent alors plusieurs contraintes qui sont :

- l'ajout d'une capacité maximale C sur les véhicules ;
- des demandes qui varient en termes de type $X = \{x_i\}$ de produits ;
- des quantités demandées q_j^i qui dépendent du client j et du type de produit i.

Les auteurs incluent aussi dans cette classe, les problèmes où tous les véhicules n'ont pas la même capacité (flotte hétérogène) et le cas où leur nombre est limité. Par la suite, d'autres problèmes sont venus étoffer cette liste et à l'intérieur de cette famille sont apparues différentes sous-classes.

Il est communément admis (Ramdane-Chérif, 2002) que l'ensemble des problèmes de tournées de véhicules se divise en deux : les problèmes de tournées sur nœuds (dont font partie le TSP et le VRP) et les problèmes de tournées sur arcs, dont les plus connus sont le problème du postier chinois (CPP - *Chinese Postman Problem*) (Eiselt et al., 1995) et le problème de tournées sur arcs (CARP – *Capacitated Arc Routing Problem*) (Golden and Wong, 1981). Cette classification est présentée sur la figure 1.1.

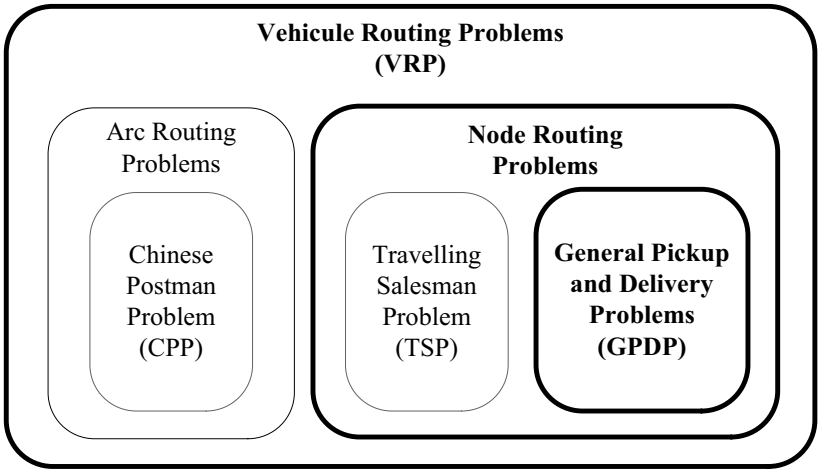


Figure 1.1 : La famille des problèmes de tournées de véhicules

Dans cette thèse, les problèmes traités font partie d'une sous-classe des problèmes de tournées sur nœuds, nommée : problèmes généraux de collecte et de livraison (GPDP - *General Pickup and Delivery Problems*).

1.1.1 Problèmes généraux de collecte et livraison (GPDP)

La famille des problèmes de tournées de véhicules avec collecte et livraison (GPDP) est définie par (Savelsbergh and Sol, 1995). Plus récemment, on peut aussi se référer à l'état de l'art de (Berbeglia et al., 2007). C'est une famille de problèmes où il ne suffit plus simplement de visiter les différents sommets avec les véhicules, mais où il faut également prendre en compte la nature des demandes. Ces demandes concernent un ensemble $X = \{x_i\}$ de types de produits. Les demandes peuvent être de deux natures différentes : soit une demande de collecte, soit une demande de livraison. Une demande est associée à un sommet j du graphe et concerne une quantité q_j^i de produit x_i. Une demande de collecte est associée sur le graphe à une quantité positive $q_j^i > 0$ tandis qu'une demande de livraison est associée à une quantité négative $q_j^i < 0$.

La solution optimale d'un problème de tournée de véhicules avec collectes et livraisons (VRPPD - *VRP with Pickup and Delivery*) se compose d'un ensemble de tournées telles que :

- toutes les demandes sont satisfaites ;
- la charge maximale des véhicules est respectée en tout point de la tournée ;
- la somme des coûts des tournées est minimale.

Dans la littérature, les problèmes de tournées de véhicules sont généralement modélisés sur un graphe orienté $G = (V, E)$, avec $V = \{1, \dots, n\}$ l'ensemble des sommets qui modélisent les demandes. A cet ensemble de sommets V est ajouté un sommet 0 qui correspond au dépôt où est positionnée la flotte de véhicules. $E = \{(i,j) | \, i,j \in V\}$ est l'ensemble des arcs si le problème est asymétrique ou des arrêtes s'il est symétrique. A un sommet i est associé un ensemble de valeurs q_i^j qui correspondent à la demande en produit x_j sur le sommet i. A une

arrête (où un arc) $(i, j) \in A$ est associée une valeur c_{ij} qui correspond au coût pour aller du sommet i au sommet j. La figure 1.2 illustre une solution composée de deux tournées. Chaque véhicule est affecté à une tournée et une tournée est une suite ordonnée de clients à traiter. Elle démarre au dépôt, visite une liste ordonnée de clients, avant de revenir au dépôt.

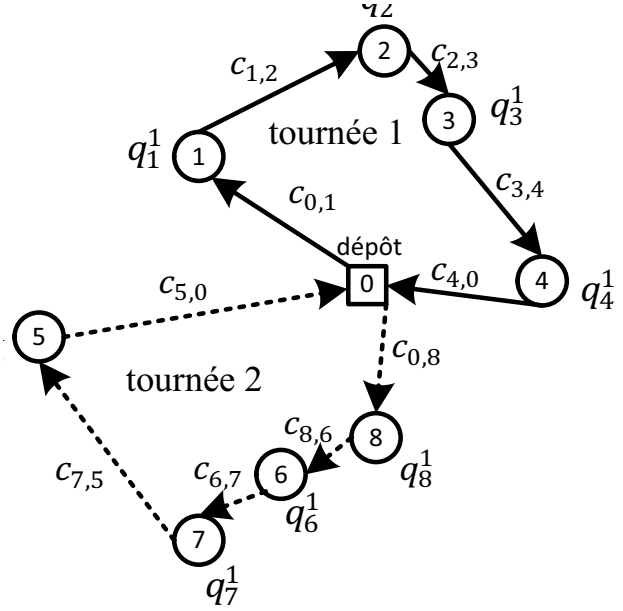
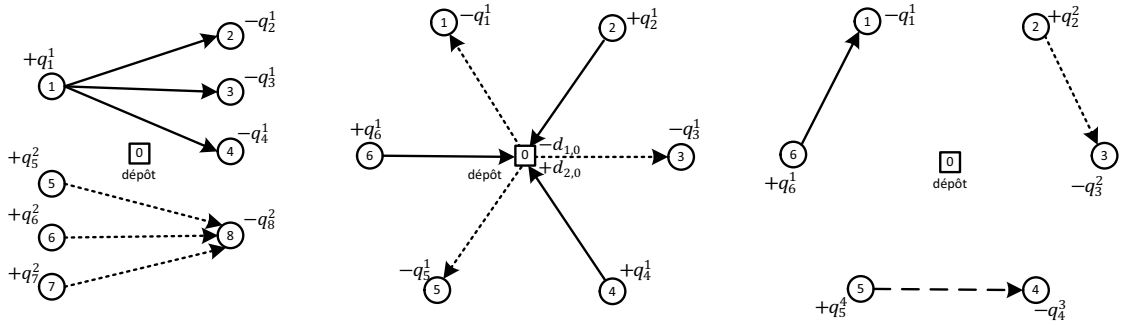


Figure 1.2 : Exemple de modélisation de GPDVRP à 8 sommets et deux véhicules

Cette famille de problèmes est classée selon deux critères principaux : la structure et le type de visites à réaliser.

a) La structure des demandes

La structure concerne la disposition des sommets de collecte et de livraison. Elle est généralement divisée en trois types, représentés sur la figure 1.3. Dans cette figure, les arcs ne représentent pas des tournées de véhicules, mais les liens de transport à réaliser entre les points de collecte et de livraison des marchandises. A chaque type d'arc correspond un type de marchandise x_i. Les arcs en trait plein correspondent aux produits $x_1$et les arcs en pointillé correspondent au produit x_2.



(a) : structure Many-to-Many (b) : structure One-to-Many-to-One (c) : structure One-to-One

Figure 1.3 : Structures pour les GPDP

La première structure est nommée Many-to-Many (M-M) et modélise des problèmes pour lesquels il existe plusieurs sommets de collecte et plusieurs sommets de livraison pour chaque type d'objets x_i (figure 1.3 (a)). La seconde structure, nommée *One-to-Many-to-One* (1-M-1) est une structure dite "centralisée". Un sommet spécifique, le plus souvent le dépôt, contient l'ensemble des produits nécessaires pour réaliser les livraisons sur les sommets. Un autre sommet spécifique, souvent le même sommet, stocke l'ensemble des produits que les véhicules collectent (figure 1.3 (b)). La troisième et dernière structure, nommée *One-to-One* (1-1), modélise des problèmes où, à chaque demande de collecte, correspond une unique demande de livraison située sur un autre sommet du graphe (figure 1.3 (c)). On a dans ce cas des requêtes de transport et les véhicules doivent s'organiser pour acheminer les produits depuis leur point d'origine vers leur point de destination. En général, on n'autorise pas le changement de véhicule pour ces transports.

b) Les types de visite

Le deuxième critère concerne la nature de la visite effectuée par le véhicule sur les sommets du graphe. Dans les problèmes de type GPDP, un sommet peut avoir différentes natures. Un sommet peut être associé avec uniquement des requêtes de collecte (resp. de livraison), dans ce cas le sommet est dit sommet de collecte (resp. sommet de livraison). Un sommet peut aussi comporter à la fois des requêtes de collecte et de livraison. Le problème est noté P/D si chaque sommet est soit un sommet de collecte, soit un sommet de livraison ; il est noté PD si au moins un sommet a simultanément une requête de livraison et une autre de collecte; il est enfin noté P-D si au moins un sommet demande de réaliser les deux opérations mais que le véhicule peut repasser plus tard pour effectuer la deuxième opération.

1.1.2 Les contraintes supplémentaires

Chacun des problèmes de la famille des GPDP peut être ensuite enrichi d'un certain nombre de contraintes additionnelles. Les contraintes les plus répandues dans la littérature sont :

- les problèmes avec fenêtres de temps ;
- les problèmes avec flotte de véhicules hétérogènes;
- les problèmes multi-dépôts ;
- les problèmes avec points de transbordement ;
- les problèmes avec temps de trajet maximal pour le véhicule ;
- les problèmes avec temps de trajet maximal pour les marchandises ;
- les problèmes avec visites multiples et traitement partiel des demandes.

a) Les fenêtres de temps

Une fenêtre de temps (TW - *Time Window*) impose qu'une ou plusieurs requêtes du problème soit traitée en respectant un intervalle pour le début de traitement du véhicule (livraison, collecte). Cet intervalle est défini par une date au plus tôt et une date au plus tard. Cette contrainte peut s'appliquer à des demandes de collecte ou de livraison comme présenté dans (Dumas et al., 1991).

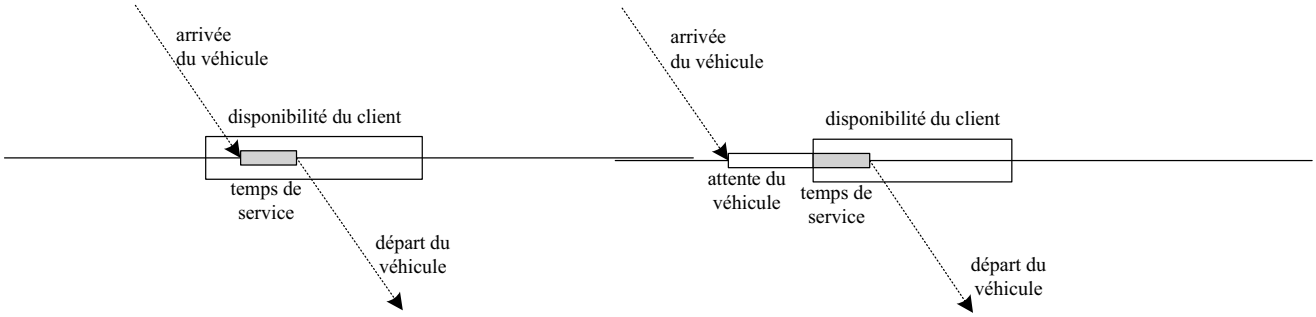


Figure 1.4 : Exemple de fenêtre de temps sur un sommet

Ce type de contrainte implique le plus souvent d'autoriser le véhicule à arriver en avance, auquel cas, il attend sur le sommet afin de démarrer ses opérations au début de la fenêtre de temps, comme illustré sur la figure 1.4. Cette attente peut être autorisée sur tous les sommets du graphe ou, uniquement sur une partie des sommets. En revanche, le véhicule n'est pas autorisé à arriver après la fin de la fenêtre de temps.

Dans la version classique de la contrainte, si une des fenêtres de temps n'est pas respectée alors la solution n'est pas valide. Il existe des variantes dans la littérature, par exemple la fenêtre de temps souple (STW - *Soft Time Window*) introduite pour la première fois sur un problème de TSP par (Sexton and Choi, 1986), puis sur un problème de VRP par (Figliozzi, 2008), qui autorise la violation de la fenêtre de temps mais pénalise la fonction objectif.

b) Les problèmes avec flotte hétérogène

Les problèmes avec flotte hétérogène sont des problèmes où la flotte de véhicules n'est pas composée de véhicules identiques. Les différences se situent le plus souvent au niveau de la charge maximale qu'ils peuvent transporter. Certains problèmes prennent également en compte d'autres différences comme par exemple :

- les coûts de possession qui représentent le coût fixe lié à l'utilisation du véhicule ;
- les coûts variables (coûts kilométriques), qui doivent être payés pour chaque kilomètre parcouru par le véhicule ;
- les types de produits transportables par le véhicule.

Il existe donc plusieurs types de problèmes, par exemple, le Vehicle Fleet Mix Problem (VFMP) introduit par (Golden et al., 1984) dans lequel uniquement la charge maximale varie. Si des coûts de possession sont ajoutés à ce problème, l'extension est appelée le (VFMP-F) dont la première métaheuristique a été proposée par (Salhi and Osman, 1996). L'extension avec uniquement des coûts variables est le VFMP-V proposée par (Taillard, 1999). Si à la fois des coûts fixes et des coûts variables sont présents le problème est un VFMP-VF proposée par (Choi and Tcha, 2007). Enfin un problème de VFMP-VF dans lequel le nombre de véhicules de chaque catégorie est limité est appelé le HVRP, on peut citer les travaux de (Li et al., 2010), (Brandão, 2011) et (Duhamel et al., 2012).

c) Les problèmes avec une durée de trajet maximale pour le véhicule

Cette contrainte impose que les tournées d'une solution respectent une durée maximale qui est la différence entre la date de sortie du dépôt et la date de retour au dépôt. Cette contrainte permet, de modéliser le temps de travail du chauffeur. Elle se rencontre souvent en complément des contraintes de fenêtres de temps et limite indirectement le temps d'attente des véhicules sur les sommets. Elle permet aussi, dans une certaine mesure, de mieux répartir la charge entre les différents véhicules. Ces contraintes peuvent aussi représenter l'autonomie

des véhicules par exemple dans le cas de véhicules électriques rechargé au dépôt (Felipe et al., 2014).

d) Les problèmes de durée de transport maximale pour les marchandises

Les produits transportés par la flotte de véhicules peuvent être de tous types, y compris des produits périssables qui doivent respecter des durées maximales de transport (c'est le cas par exemple pour les produits frais). Après avoir collecté le produit, le véhicule possède donc un délai maximal pour atteindre le sommet de livraison et livrer. Dans la littérature, cette contrainte est appelée le *maximum transit time*. Cette contrainte est également très fréquente pour le transport de personnes et sert aussi à modéliser une qualité de service du point de vue du client. C'est le cas pour le problème du Dial-a-Ride dont l'article de référence est celui de (Cordeau and Laporte, 2003a).

e) Les problèmes multi-dépôts

Dans un problème multi-dépôts, le graphe G comporte plusieurs dépôts d'où les véhicules peuvent partir. Le plus souvent les véhicules doivent retourner à leur dépôt d'origine à la fin de leur tournée. Le nombre de véhicules (ou les types de véhicules) peuvent être différents sur chaque dépôt.

Dans certains problèmes, la méthode d'optimisation doit aussi affecter les véhicules aux dépôts car la répartition des véhicules n'est pas imposée par l'instance. Ces problèmes font partie de la sous-famille des problèmes de localisation et de tournées (LRP - *Location-Routing Problem*). Sur ces problèmes (Prodhon and Prins, 2014) ont publié un état de l'art. La figure 1.5 est un exemple de solution pour un problème avec deux dépôts et trois tournées.

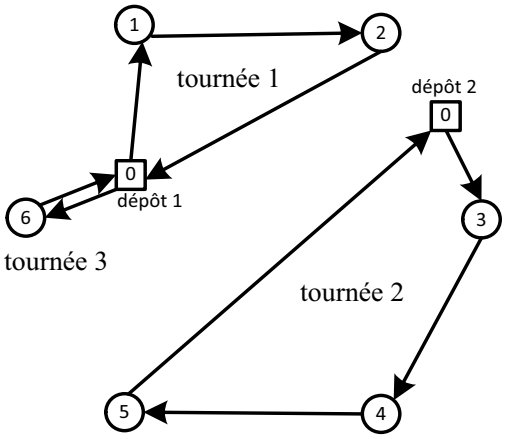


Figure 1.5 : Exemple de solution multi-dépôt d'un VRP

f) Les problèmes avec points de transbordement

Les points de transbordement sont des sommets du graphe sur lesquels il est possible de transférer les produits entre véhicules. Ainsi un véhicule peut déposer une partie des produits qu'il transporte afin soit de les reprendre plus tard, soit pour qu'un autre véhicule les prennent en charge, comme illustré sur la figure 1.6. Les sommets de transbordement sont alors rassemblés dans un ensemble nommé T.

Deux principales versions peuvent être envisagées : une première version dans laquelle les produits peuvent attendre seuls sur les sommets de transbordement et une seconde version dans laquelle les produits ne peuvent pas être laissés sur le sommet de transbordement.

Le deuxième cas apparaît essentiellement lorsque la valeur de ces produits est élevée (à cause du risque de vol) ou lorsque le transport concerne non pas des produits, mais des individus qu'on ne peut pas laisser seuls sur un point de rendez-vous. Cette dernière remarque prend tout son sens dans les cas de transport public de personnes mineures, en particulier des enfants scolarisés, de personnes âgées ou de personnes handicapées (Ren, 2012).

Les problèmes avec des nœuds de transbordement nécessitant la présence d'un véhicule impliquent, lors de la résolution, de synchroniser plusieurs tournées minimisant les temps d'attente sur ces sommets.

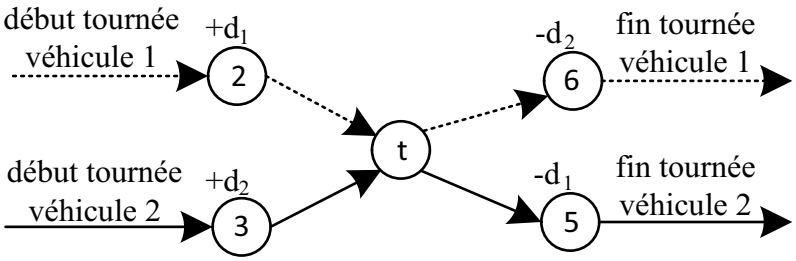


Figure 1.6 : Problème avec un point de transbordement

1.1.3 Les problèmes les plus connus de la famille des GPDP

La figure 1.7 illustre les liens entre quelques problèmes de la famille des problèmes de collecte et de livraison (GPDP - *General Pickup and Delivery Problem*) qui sont les plus étudiés dans la littérature.

Dans cette représentation inspirée de (Berbeglia et al., 2007), les trois branches principales sont représentées avec les problèmes *One-to-One* (1-1), *Many-to-Many* (M-M), *One-to-many-to-One* (1-M-1). Les arcs entre les problèmes représentent les contraintes à ajouter ou à supprimer pour passer de l'un à l'autre. Sous les différents problèmes sont rappelés des articles de référence. Les contraintes situées sur les arcs sont les contraintes qui ont été présentées dans la partie 1.1.2.

Hormis "*Backhauls*" et "$Mixtes$" qui appartiennent à la branche des One-to-Many-to-One, ces contraintes correspondent au cas où le problème est de type P/D, introduit initialement par (Goetschalckx and Jacobs-Blecha, 1989). Les sommets de livraison et de collecte sont donc séparés. Si le problème impose aux solutions de traiter des sommets de livraison avant les sommets de collecte, alors le problème doit respecter la contrainte "*Backhauls*" sinon il est appelé "$Mixtes$".

Le problème du transport à la demande (DARP - *Dial-A-Ride Problem*) qui est traité dans cette thèse appartient à la famille des $One\text{-}to\text{-}One$. Il est détaillé dans la partie 1.4 .

1.1.4 Les différentes versions du problème

Tous les problèmes abordés précédemment peuvent être traités sous plusieurs versions différentes, les plus connues sont :

- les problèmes statiques ;
- les problèmes dynamiques ;
- les problèmes périodiques.

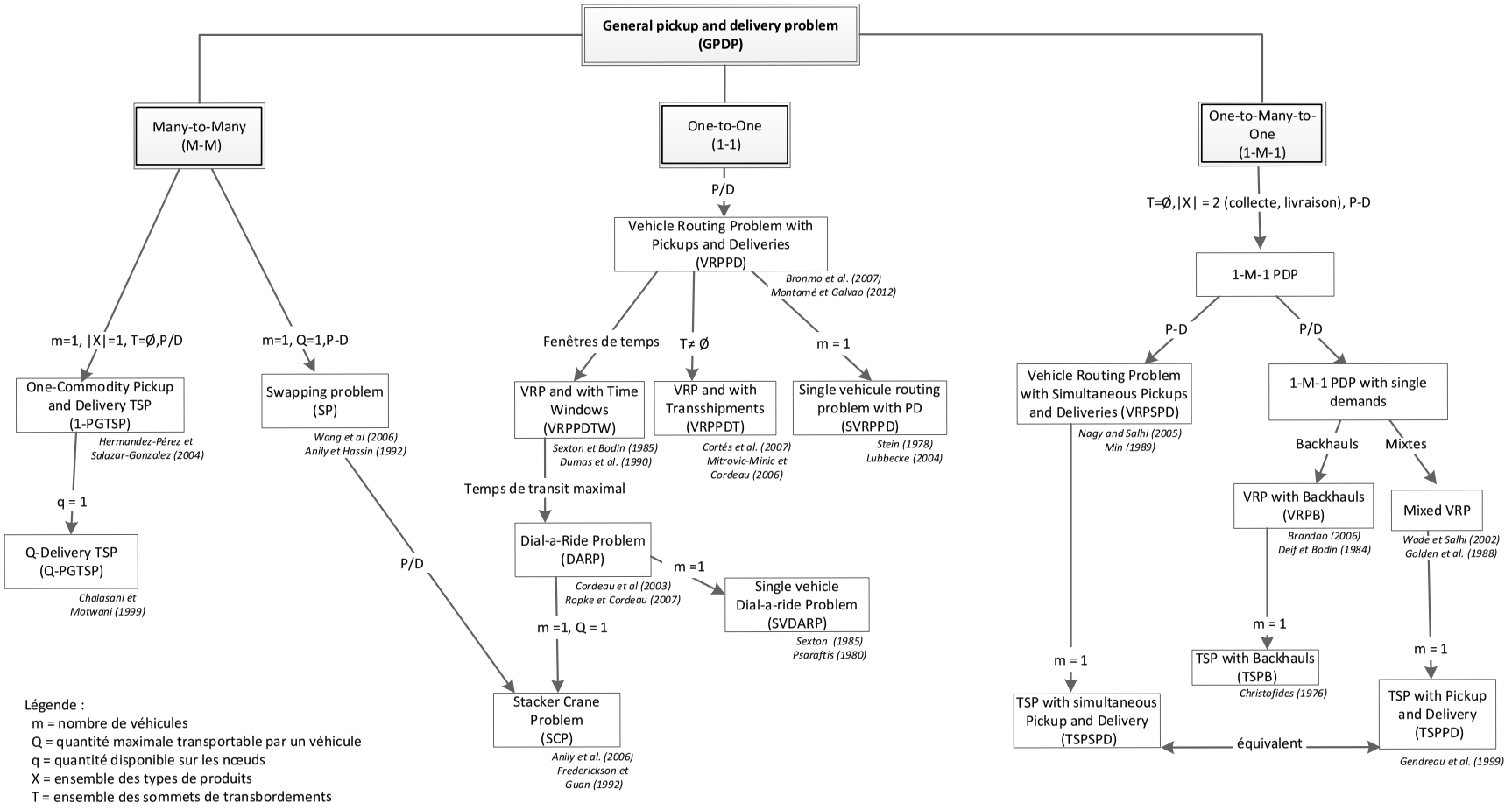


Figure 1.7 : La famille des problèmes de tournées de véhicules avec collecte et livraison (GPDP)

a) Les problèmes statiques

Par défaut, les problèmes sont considérés statiques et la totalité des informations est supposée disponible au moment de l'optimisation. Une solution du problème doit satisfaire toutes les contraintes.

b) Les problèmes dynamiques

Les problèmes dynamiques sont plus proches du cas pratique. Dans cette version, une partie des informations (où la totalité) n'est disponible qu'après le début de la tournée. Les méthodes de résolution employées doivent donc être capables d'anticiper les changements et de modifier rapidement la solution courante pour faire face aux nouvelles informations.

Ces informations sont le plus souvent des nouvelles demandes qui doivent être prises en compte sur l'horizon de planification, mais aussi des demandes qui sont supprimées ou des demandes dont des contraintes sont modifiées (pensons à la fenêtre de temps par exemple).

Le terme "dynamique" est parfois aussi employé pour caractériser une évolution dans le réseau routier qui n'est pas prévisible et qui doit être prise en compte une fois que les tournées ont commencé. On peut imaginer des arcs qui sont supprimés du graphe pour modéliser des rues indisponibles pour travaux ou pour cause d'accidents de la circulation. On peut aussi prendre en compte l'apparition de congestion sur le réseau routier.

c) Les problèmes périodiques

Cette version du problème considère un ensemble de n périodes de temps (généralement des heures ou des jours). Durant chaque période, les sommets doivent être visités un certain nombre de fois déterminé par l'instance. Ceci correspond aux types de fonctionnement tel qu'on le retrouve dans les services publics de transport comme les bus ou les tramways. Ceci englobe aussi les problèmes de livraison périodique pour les entreprises (fuel, gaz, fournitures).

1.2 Complexité et classe de problème

Les problèmes peuvent être regroupés en différentes classes selon leur complexité. Ce sont les *classes de complexités*. La complexité d'un problème est définie par rapport à la plus faible complexité associée aux méthodes qui permettent de le résoudre. Définir cette complexité est délicat puisque plusieurs algorithmes peuvent permettre de résoudre un même problème et leur complexité peut être difficilement comparable. De plus, rien n'interdit, sauf résultat théorique, l'existence d'algorithme offrant une meilleure complexité. C'est le cas par exemple du problème de multiplication matricielle : l'algorithme trivial est en $O(n^3)$, une borne inférieure triviale est en $O(n^2)$ et un des meilleurs algorithme actuels, dû à Coppersmith et Winograd, est en $O(n^{2,376})$. Parmi les différentes classes de problèmes les classes P et NP jouent un rôle particulier pour les problèmes de tournées de véhicules.

1.2.1 La classe P

La classe P contient les problèmes pour lesquels on peut fournir la réponse en temps polynômial par rapport aux données en entrée, sur une machine de Turing déterministe. En pratique, un problème appartient à la classe P si on peut le résoudre par un algorithme de complexité polynomiale en fonction des entrées. Le problème de l'arbre T couvrant de poids minimal (MST – *Minimum Spanning Tree*) dans un graphe connexe $G = (V, E)$ est un exemple de problème de la classe P. Le MST est utilisé dans les algorithmes d'approximation

sur les problèmes de tournées de véhicules, présentés dans la partie 1.3.2. Il existe plusieurs algorithmes de résolution polynomiale du MST, un des plus connus est l'algorithme glouton proposé par Prim en 1957.

1.2.2 La classe NP

La classe NP est formée des problèmes pour lesquels on peut fournir une réponse en temps polynômial en fonction des données en entrée, sur une machine de Turing non-déterministe. La différence avec la classe P réside dans la nécessité d'une machine de Turing non-déterministe, c'est-à-dire une machine capable d'étudier plusieurs transitions simultanément. De fait, la suite des opérations correspond à un arbre de transitions là où elle correspond à une séquence de transitions pour une machine de Turing déterministe.

La classe NP contient la plupart des problèmes d'optimisation combinatoire, dont les problèmes NP-complets. La sous-classe NP-complet a été définie par (Cook, 1971). Elle est formée des problèmes NP tels que tout problème NP peut se ramener à eux par réduction polynômiale. Ainsi, si on pouvait résoudre de manière polynômiale un problème NP-complet, alors tous les problèmes NP pourraient être résolus de manière polynômiale et la conjecture $P = NP$ serait vérifiée. Valider (ou réfuter) cette conjecture est un des problèmes mathématiques contemporains les plus importants et nombre de travaux visent à identifier un algorithme polynômial pour résoudre un problème NP-complet. Evidemment P est inclus dans NP.

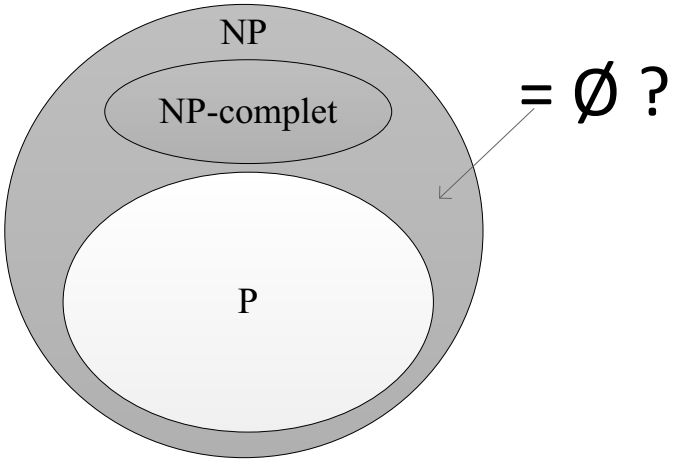


Figure 1.8 : Inclusion des problèmes de classes P, NP, NP-complet

A la base, NP et NP-complet concerne les problèmes de décision. On y inclut aussi les problèmes d'optimisation on considérant qu'on peut toujours transformer un problème de d'optimisation en problème de décision. De fait, nombre de problèmes d'optimisation combinatoires « classiques » sont NP-complets. Les travaux de Karp (Karp, 1972) à partir du problème SAT ($satisfiability$) ont permis de dresser une première liste de 21 problèmes NP-complets par réduction polynômiale entre eux, dont le TSP. Le VRP est NP-complet car il généralise le TSP (Lenstra and Kan, 1981).

Notons enfin la classe des problèmes NP-difficiles, constituée des problèmes tels que tout problème NP peut se réduire polynômialement à eux. Ces problèmes ne sont pas forcément NP et NP-difficile est donc plus large que NP.

1.3 Les méthodes de résolutions

Pour résoudre ces problèmes de tournées de véhicules, différents types d'approches existent. Elles sont réparties en deux catégories principales :

- les méthodes exactes ;
- les méthodes approchées.

Les méthodes exactes garantissent l'obtention d'une solution optimale au problème. Sur les problèmes NP, on ne connaît pas d'algorithme exact en temps polynômial. Par conséquent les méthodes exactes ont une complexité exponentielle et elles ne peuvent résoudre en temps raisonnable que des instances de taille limitée (en fonction de la nature du problème). Par exemple, la taille maximale des problèmes de VRP résolus dans la littérature avec des méthodes exactes n'excède pas 200 clients (Toth and Vigo, 2002).

Les méthodes approchées ne garantissent pas l'optimalité de la solution calculée. Par contre elles permettent d'obtenir des solutions de bonne qualité en des temps de calculs raisonnables. La figure 1.9 représente un ensemble non exhaustif des différentes méthodes de résolution qui sont utilisées pour les problèmes de tournées de véhicules.

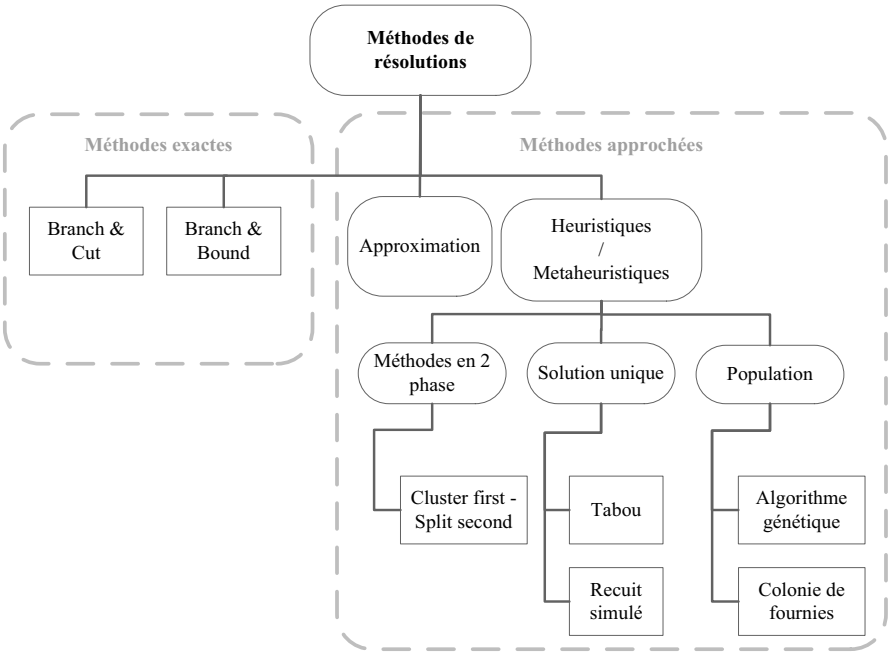


Figure 1.9 : Les méthodes de résolution appliquées aux problèmes de tournées de véhicules

1.3.1 Les méthodes exactes

Les méthodes exactes sont des méthodes efficaces sur les petites instances et les algorithmes exacts reposent souvent sur des formulations mathématiques des problèmes. Malheureusement, pour les instances de tailles moyennes ou grandes, ces méthodes ne permettent pas d'obtenir des solutions en temps raisonnable.

Les méthodes de résolution exactes utilisées pour les problèmes de tournées de véhicules peuvent être répartis en plusieurs groupes (Laporte, 1992). Le groupe des méthodes dites de branchement est présenté ici. Il existe entre autres deux types de méthodes : le *Branch & Bound* et le *Branch & Cut*. Ces deux méthodes reposent sur le principe de séparation/évaluation.

La méthode de *Branch & Bound* consiste à résoudre une relaxation du problème étudiée. Le problème étudié est appelé P et la relaxation associée est appelée P_0. Par exemple pour le cas d'un problème linéaire en nombre entiers, P_o peut correspondre à la relaxation continue. Cette étape est appelée l'évaluation. Si la solution optimale obtenue pour P_o n'est pas une solution du problème étudié P, alors une phase de séparation est utilisée. Cette phase de séparation consiste à ajouter à P_o des contraintes sur les bornes d'une variable afin de supprimer la précédente solution optimale de l'espace des solutions. A chaque contrainte ajoutée, un sous-problème P_i est introduit.

Par exemple la figure 1.10 présente une représentation graphique d'un problème linéaire en nombres entiers à deux variables. Le résultat de l'évaluation de P_o donne la solution dont la variable X_1 est fractionnaire ($x_1 = 1.57$). Deux nouvelles contraintes sur la borne de x_i sont alors introduites. La contrainte $x_1 \leq \lfloor 1.57 \rfloor \leq 1$ créant P_1 et la contrainte $x_1 \geq \lceil 1.57 \rceil \geq 2$ créant le sous-problème P_2.

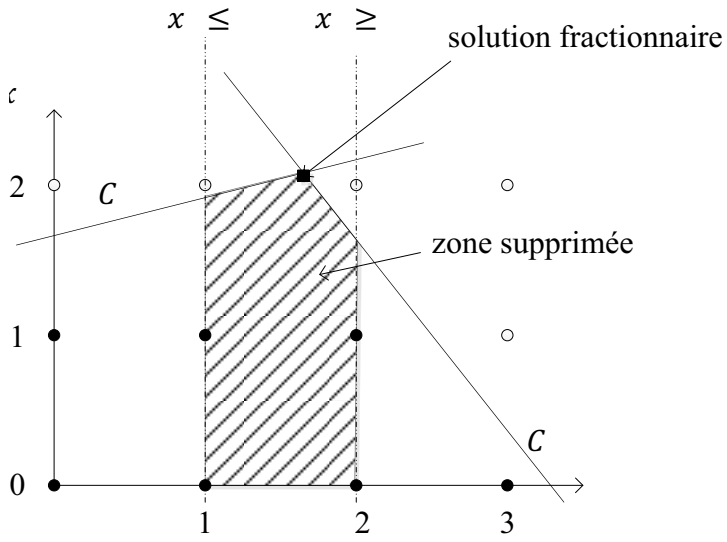


Figure 1.10 : Opération de séparation de P_o

Ces différents sous-problèmes sont alors modélisés sous la forme d'un arbre dans lequel P_o est la racine et à chaque séparation, de nouvelles branches sont créées qui correspondent aux sous-problèmes.

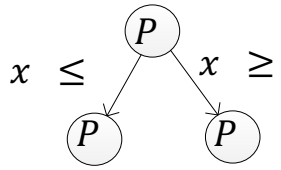


Figure 1.11 : Schémas arborescents et opération de séparation de P_o

A chaque itération, un sous-problème situé sur une feuille de l'arbre est évalué, et si nécessaire une nouvelle séparation est effectuée. Ce procédé est répété jusqu'à ce que toutes les feuilles aient été évaluées. Il est possible de limiter l'exploration de l'arbre en se limitant aux feuilles pouvant conduire à des solutions "intéressantes".

Une feuille dont la solution est aussi une solution du problème P est appelée une feuille "réalisable". La solution optimale de P correspond à la meilleure solution obtenue sur les feuilles de l'arbre.

Une autre méthode de résolution exacte utilisée dans les problèmes de tournées de véhicules est le *Branch & Cut*. Il reprend les mêmes étapes de séparation et d'évaluation que le *Branch & Bound* mais intègre en plus une méthode de plans de coupe (*cutting-plane*). La recherche de plans de coupe s'applique lorsque la solution d'un problème relaxé n'est pas une solution de P_o. Elle consiste à trouver une "coupe" (*cut*), c'est-à-dire une contrainte, telle que :

- toutes les solutions réalisables sont réalisables pour la coupe ;
- la solution fractionnaire courante n'est pas réalisable pour la coupe.

Ajouter cette coupe améliore donc le sous-problème courant sans entraîner de séparation puisque toute solution réalisable reste réalisable. Une nouvelle feuille est alors introduite, comme illustré dans la figure 1.12.

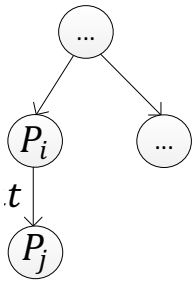


Figure 1.12 : Ajout d'une contrainte de coupe

Après l'évaluation d'une feuille non-réalisable, là où le *Branch & Bound* effectue une phase de séparation, le *Branch & Cut* cherche à trouver une coupe qui lui permet d'améliorer la relaxation. Si une coupe est trouvée, il recommence l'évaluation. Si aucune coupe n'est trouvée alors l'algorithme effectue une séparation comme pour le *Branch & Bound.*

Pour le *Branch & Cut,* la solution optimale de *P* correspond à la meilleure solution obtenue sur les feuilles réalisables de l'arbre.

1.3.2 Les méthodes approchées

Les méthodes approchées permettent de traiter rapidement des instances de grandes tailles. Les solutions obtenues sont généralement de bonne qualité mais leur optimalité ne peut être prouvée *ex-nihilo*. Ces méthodes approchées peuvent être de deux natures :

- les méthodes d'approximation ;
- les méthodes heuristiques.

a) Les méthodes d'approximation

Les méthodes d'approximation permettent d'obtenir des solutions qui ne sont pas prouvées optimales. Cependant la valeur de la fonction objectif associée à chaque solution est inférieure (dans le cas d'une minimisation) à αC^* avec α une constante connue telle que $\alpha \geq 1$ et C^* le coût associé à la solution optimale. Une telle méthode est alors appelée une α-approximation. Comme illustré dans la figure 1.13, le coût de la solution retournée par l'algorithme est donc situé entre C^*et αC^*. Donc plus la valeur de α est proche de 1 est plus l'approximation est intéressante.

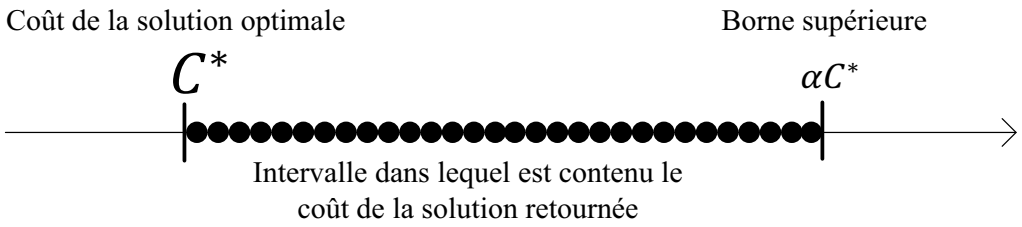


Figure 1.13 : Encadrement de la valeur de la solution dans un algorithme d'approximation

La difficulté de ces algorithmes est que le coût de la solution optimale C^* n'est pas connu. Pour obtenir la preuve de l'approximation, le plus souvent, une borne inférieure B_{inf} est utilisée puisque si $C \leq \alpha B_{inf}$ alors $C \leq \alpha C^*$. Le coût C de la solution retournée par l'algorithme est connu mais comme illustré dans la figure 1.14, il est difficile de savoir où elle se situe par rapport à la solution optimale C^*.

Figure 1.14 : Qualité de l'approximation en sans connaître la valeur optimale

Parmi les problèmes de tournées de véhicules pour lesquels il existe des méthodes d'approximation les deux plus connus sont :

- le problème du voyageur du commerce ;
- le *Staker Crane Problem* (SCP).

Pour le problème du voyageur du commerce (TSP), l'approximation de (Christofides, 1976) propose une 3/2-approximation. Une condition nécessaire à la mise en place de cette α-approximation est le respect de l'inégalité triangulaire dans le graphe $G = (E, V)$ considéré, donc :

$$c(i,j) \leq c(i,k) + c(k,j) \;\; \forall\, i,j,k \in E$$

L'algorithme de Christofides repose sur l'algorithme polynomial de Prim qui calcule un arbre couvrant T de poids minimal et une version simplifiée est illustrée dans la figure 1.15. Pour une instance du TSP (figure 1.15 (a)) l'arbre T est calculé (figure 1.15(b)). Le coût associé à cet arbre est utilisé comme une borne inférieure pour prouver la 3/2-approximation de la solution. Chaque arrête de T est alors remplacée par deux arcs orientés dans le sens opposé l'un à l'autre (figure 1.15(c)). Cette tournée partielle dans laquelle les sommets sont visités plusieurs fois est alors simplifiée en utilisant l'inégalité triangulaire (figure 1.15(d)) afin d'obtenir une solution du TSP (figure 1.15(e)).

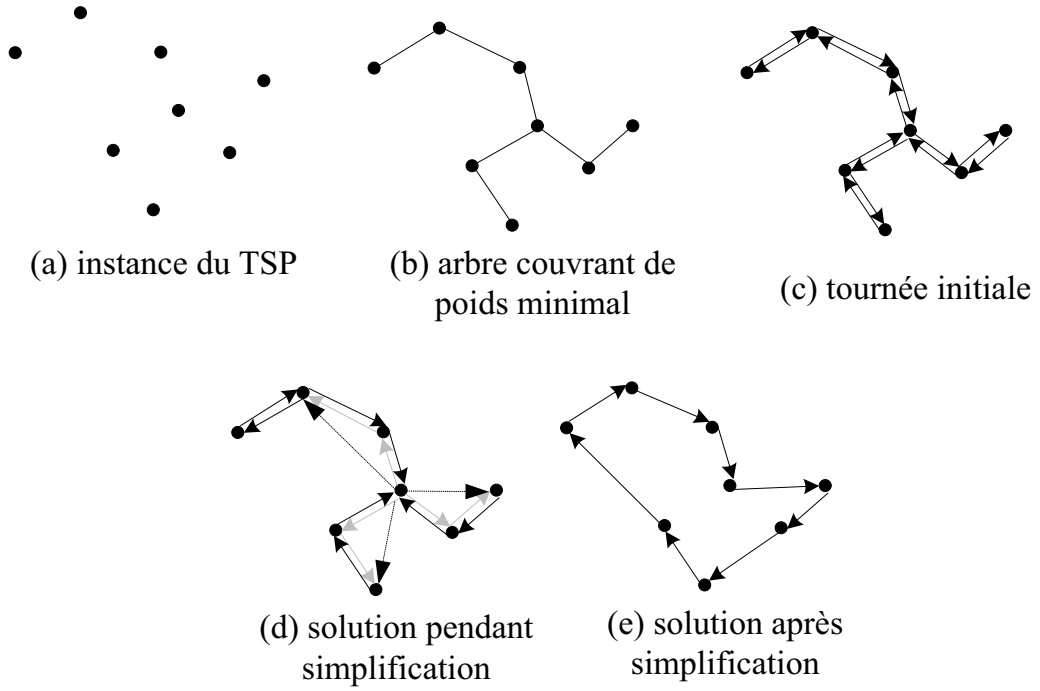


Figure 1.15 : Version simplifiée de l'algorithme d'approximation pour le problème du TSP

La deuxième approximation concerne le *Staker Crane Problem* (SCP) qui est un problème de type *One-to-One* dans lequel le véhicule a une capacité unitaire. Le problème ne comporte pas de sommet de transbordement et il prend donc un client sur son sommet d'origine pour le déposer directement à son sommet de destination, à la manière des taxis. L'algorithme d'approximation concerne uniquement le problème sans fenêtre de temps. L'algorithme est une 9/5-approximation qui a été proposée par (Frederickson et al., 1978) : cet algorithme utilise aussi un arbre couvrant de poids minimal pour obtenir la solution.

Les problèmes de tournées de véhicules comportent souvent beaucoup de contraintes et il est donc difficile de borner la solution et d'obtenir des algorithmes d'approximation.

b) Les méthodes heuristiques et métaheuristiques

Les méthodes heuristiques et les métaheuristiques sont des méthodes qui permettent d'obtenir des solutions de bonne qualité en des temps raisonnables. La différence entre les deux méthodes c'est qu'une heuristique est spécifique à un problème d'optimisation alors qu'une métaheuristique est un schéma algorithmique "général" qui peut s'appliquer à différents problèmes d'optimisation.

Dans cette thèse, les stratégies de résolution proposées sont des métaheuristiques et elles permettent de parcourir efficacement l'espace des solutions associées à un problème. Pour cela, elles utilisent des stratégies indépendantes du problème traité. Elles guident le parcours de l'espace des solutions afin de sortir rapidement des minima locaux et de se diriger vers les régions intéressantes. Il existe de nombreuses métaheuristiques avec des schémas algorithmes plus ou moins simples.

Les métaheuristiques appliquées aux problèmes de tournées de véhicules ont reçu beaucoup d'attention de la communauté scientifique. Un très bon état de l'art est proposé dans le livre de Gendreau et Potvin (Gendreau and Potvin, 2010) et aborde en particulier la recherche Tabou (TS - *Tabu Search*), la recherche par voisinages variables (VNS - *Variable Neighborhood Search*) et les algorithmes génétiques (GA - *Genetic Algorithms*). De nombreux auteurs ont

contribué à ce livre et à chaque méthode est associé un état de l'art complet indiquant, entre autre, sur quels problèmes ces méthodes ont été appliquées et avec quelle réussite.

On peut affirmer que les métaheuristiques intègrent toutes des mécanismes d'intensification et de diversification. Généralement, la diversification est la phase qui permet à la méthode de sortir des minima locaux et l'intensification permet au contraire de rechercher les minima locaux dans le voisinage de solutions de bonne qualité.

Maintenant que les différentes méthodes de résolutions ont été présentées de manière générale, la partie suivante présente le problème traité dans cette thèse qui est le problème du Transport à la demande.

1.4 Le problème du transport à la demande

1.4.1 Positionnement dans la famille des GPDP

Le problème du transport à la demande (DARP - *Dial-A-ride Problem*) fait partie des problèmes de type *One-to-One* (comme illustré sur la figure 1.16). C'est un cas particulier du VRPTW qui étend le VRP par l'ajout de fenêtre de temps sur les clients. Le DARP étend le VRPTW essentiellement par le fait qu'on considère des requêtes de transport et par l'ajout d'une contrainte de temps de transport maximal pour les éléments. La différence fondamentale entre ce problème et les autres problèmes de type *One-to-One* (par exemple le problème de collecte et livraison (PDP – *Pickup and Delivery Problem*) concerne les éléments à transporter qui sont des personnes et non pas des produits. A l'origine, cette contrainte de temps de transport est la seule contrainte ajoutée pour modéliser la qualité de service. Elle impose à la tournée de déposer le client dans un temps raisonnable après son chargement dans le véhicule.

Cette contrainte n'est pas suffisante pour prendre en compte toutes les facettes de la qualité de service à associer au transport de personnes, d'autant plus que le problème de transport à la demande est un problème souvent associé aux transports de personnes âgées ou à personnes à mobilité réduite. C'est pourquoi de nombreuses versions du DARP sont apparues dans la littérature et ces versions prennent en compte d'autres contraintes afin de modéliser cette qualité de service, comme par exemple :

- le temps d'attente ;
- le temps de trajet ;
- le respect des fenêtres de temps ;
- etc…

Plutôt que d'être ajoutés comme des contraintes au modèle, certains auteurs (Jorgensen et al., 2007) et (Parragh et al., 2010) intègrent ces critères dans la fonction objectif. Ils traitent alors le problème de manière multicritère ou utilisent un mécanisme d'agrégation de critères par pondération. La fonction objectif du DARP consiste, comme pour tous les problèmes de tournées de véhicules, à minimiser la distance parcourue par la flotte.

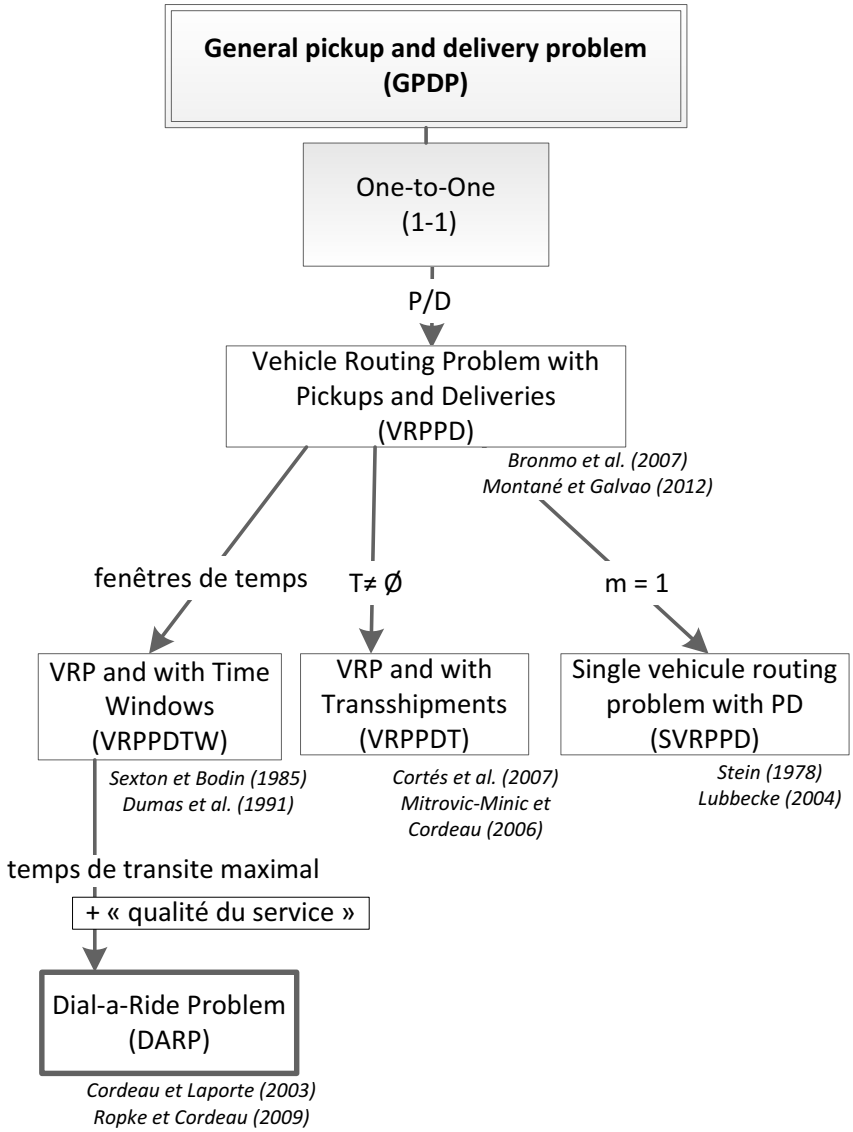


Figure 1.16 : Les problèmes de la famille 1-1

Une étude a été réalisée dans (Paquette et al., 2009) sur les différents critères (associés à la notion de qualité de service) qui ont été pris en compte dans la littérature, ou qui pourraient être pris en compte dans un problème de DARP. Dans cet article, les auteurs proposent des critères plus difficiles à évaluer comme le confort du client ou la sécurité durant le trajet, avec par exemple la probabilité d'avoir un accident, ou encore, pour les personnes en fauteuil roulant, la probabilité de tomber durant le trajet.

Un autre élément de nature à entraîner la création de différentes versions du DARP, consiste à ajouter d'autres points de vue comme celui de l'entreprise, celui du chauffeur ou récemment un point de vue tourné vers l'écologie, à la qualité de service du point de vue des clients (Paquette et al., 2009). La notion de tournée "propre" ou économe en rejet de CO2 définit aussi une qualité de service (Santos and Xavier, 2015).

Ces différentes versions du problème sont apparues surtout durant les 20 dernières années, et elles sont détaillées dans la partie 1.4.3.

1.4.2 Les premières démarches de résolutions

L'un des premiers articles sur le DARP a été réalisé par (Stein, 1978) dans lequel un modèle mathématique est proposé. Puis (Psaraftis, 1980) propose une méthode de résolution sur un DARP mono-véhicule. Dans ce problème mono-véhicule, l'objectif est de minimiser le niveau d'insatisfaction des clients. Ce niveau d'insatisfaction est calculé en fonction de l'attente et du temps de trajet des clients. Il a été résolu dans l'article de manière exacte en utilisant la programmation dynamique sur une instance de 15 sommets soit 7 clients et le dépôt. Les instances ont été traitées de manière statique et dynamique. Sur ce même problème, quelques années plus tard, (Sexton and Bodin, 1985) propose une heuristique basée sur une décomposition de Benders.

Une des premières méthodes de résolution du DARP multi-véhicules est publiée en 1986 par (Jaw et al., 1986). Dans le modèle proposé, les clients spécifient, soit une date pour venir les chercher, soit une date pour les déposer. La fonction objectif, quant à elle, met en balance le coût et la qualité de service associée aux clients.

Le coût est la distance parcourue par la flotte et la qualité de service fait référence aux temps de trajet des clients. L'heuristique proposée est un algorithme d'insertion qui ajoute successivement les clients dans une des tournées partielles sans remettre en cause les choix précédents. Leur algorithme a été testé sur une instance composée de plus de 2600 clients, et avec plus de 20 véhicules. Les années suivantes plusieurs auteurs, ont proposé des améliorations à cette heuristique de construction.

1.4.3 Les articles récents sur le DARP

La définition formelle et actuelle du problème du transport à la demande a été proposée en 2003 par (Cordeau and Laporte, 2003a). Les mêmes auteurs ont ensuite publié deux états de l'art sur le DARP ainsi que les variations qui sont apparues, le premier en 2003 (Cordeau and Laporte, 2003b), puis quelque années plus tard (Cordeau and Laporte, 2007).

Dans (Cordeau and Laporte, 2003a), le modèle traité par les auteurs est le même que celui défini dans la version initiale. La contrainte de temps de transport est imposée et le seul critère optimisé est la distance parcourue. L'avantage de ce modèle c'est qu'il définit les critères de qualité précisément, et même s'ils ne sont pas pris en compte dans la fonction objectif (ni dans les contraintes). Ceci fournit un cadre formalisateur qui a permis aux autres articles d'utiliser les mêmes notations. On peut par exemple affecter à une solution:

- un temps total d'attente ;
- un temps total de trajet ;
- une durée totale de tournée ;
- etc…

Dans le schéma de résolution, les auteurs proposent une fonction d'évaluation qui non seulement, permet de savoir si une tournée respecte les contraintes (ce qui est suffisant pour calculer le coût de la solution) mais qui en plus permet d'obtenir une tournée qui minimise la durée totale de la tournée. Pour finir, cet article a aussi fourni à la communauté un jeu d'instances qui est encore utilisé à l'heure actuelle.

Même si la plupart des articles publiés reprennent la définition de (Cordeau and Laporte, 2003a), d'autres versions du DARP sont apparues dans la littérature. Le tableau 1.1 regroupe

une partie des articles publiés sur le problème du transport à la demande depuis l'article de (Cordeau and Laporte, 2003) et les nuances qui peuvent exister. Dans ce tableau, 9 articles sont cités, parmi lesquels certains reprennent la définition de (Cordeau and Laporte, 2003a). Il s'agit de (Ropke and Cordeau, 2009), (Parragh and Schmidt, 2013), (Masson et al., 2014) et (Braekers et al., 2014).

Pour ces travaux, les méthodes de résolutions sont comparées sur les instances de (Cordeau and Laporte, 2003a) notées instances [1] et celles de (Ropke and Cordeau, 2009) notées instances [2] dans le tableau 1.1. Le tableau 1.1 est construit sur le même modèle que celui proposé par (Cordeau and Laporte, 2003a). Il comporte 7 colonnes qui sont :

- *référence,* la référence de l'article ;
- *type,* le type de flotte utilisée ;
- *fonction objectif*, les différents critères inclus dans la fonction objectif du modèle ;
- *fenêtre de temps,* sur quels types de sommets se situent les fenêtres de temps ;
- *autres contraintes,* les autres contraintes prises en compte par le modèle ;
- *algorithmes,* les méthodes utilisées par les articles ;
- *taille des instances,* la taille des instances sur lesquelles est traité le problème.

Dans l'article de (Ropke and Cordeau, 2009) de nouvelles instances et un algorithme de *Branch & Cut* ont été proposés. Les instances sont de tailles plus réduites que celles proposées par (Cordeau and Laporte, 2003) afin d'être résolues de manière exacte, mais elles correspondent au même modèle.

Récemment, plusieurs articles ont permis d'améliorer les résultats sur les instances de la littérature. Ces articles correspondent aux trois dernières lignes du tableau 1.1. Tout d'abord, (Parragh and Schmidt, 2013) ont proposé une métaheuristique hybride entre une génération de colonne et une métaheuristique de type Large Neighborhood Search (LNS). Ensuite (Masson et al., 2014) ont publié une métaheuristique qui peut résoudre le problème du DARP dans lequel on a la possibilité d'ajouter des sommets de transbordement.

Enfin (Braekers et al., 2014) ont publié deux méthodes sur le DARP multi-dépôt avec une flotte de véhicules hétérogènes : une méthode exacte de type $Branch\ \&\ Cut$ appliquée sur leur instance et une méthode heuristique nommée Deterministic Annealing (DA).

Les méthodes de ces trois articles ont été exécutées sur les instances classiques du DARP. Ceci a été rendu possible du fait que les articles (Masson et al., 2014) et (Braekers et al., 2014) traitent de problèmes généralisant le DARP.

Tableau 1.1 : comparaison des différents DARP étudiés dans la littérature depuis l'article (Cordeau and Laporte, 2003)

Référence	**type de flotte**	**fonction objectif**	**fenêtres de temps**	**autres contraintes**	**algorithmes**	**taille des instances**
[1] (Cordeau and Laporte, 2003a)	homogène	Min distance	sommets d'origine ou de destination	max. route max. transport	métaheuristique recherche Tabou	$24 \leq n \leq 295$ $3 \leq m \leq 13$
(Luo and Schonfeld, 2007)	homogène	Min nb véhicules	sommets d'origine ou de destination	max. transport	heuristique rejected-reinsertion	instances générées aléatoirement
[2] (Ropke and Cordeau, 2009)	homogène	Min distance	sommets d'origine ou de destination	définition de [1]	algorithme exact Branch & Cut	$16 \leq n \leq 96$ $2 \leq m \leq 8$
(Karabuk, 2009)	homogène	Max requêtes	sommets d'origine	max. route max transit multi-dépôts	hydride génération de colonnes et LNS	deux instances : n=357 m=36; n=680 m=48
(Lin et al., 2012)	homogène	Max satisfaction Min coût	sommets d'origine et de destination	max transport	métaheuristique recuit simulé	n = 29 et graphe G(V,E) avec |V|=9 et |E| =17
(Chevrier et al., 2012)	homogène	Min nb véhicules Min temps trajets Min délais	fenêtres avec une tolérance	délais limités	métaheuristique EA multi-objectif + recherche locale	20 inst. $n = 100$ 10 inst. $n = 10000$
(Parragh and Schmidt, 2013)	homogène	Min distance	sommets d'origine et de destination	définition de [1]	hydride génération de colonnes et LNS	Instances [1][2]
(Masson et al., 2014)	homogène	Min distance	sommets d'origine et de destination	définition de [1] + transbordement	Adaptive LNS	Instances [1][2] + instances personnelles
(Braekers et al., 2014)	hétérogène	Min distance	sommets d'origine et de destination	définition de [1] + multi-dépôts	Branch & Cut (Deterministic Annealing)	Instances [1][2] + instances personnelles

1.5 Conclusion

Ce chapitre donne une présentation des problèmes de tournées de véhicules et des liens qui existent entre ces problèmes. Ensuite des notions de complexité sont rappelées avant de présenter les différentes méthodes de résolutions qui peuvent être envisagées. Il y a d'une part, les méthodes exactes qui sont difficiles à mettre en œuvre sur les problèmes de grandes tailles et, les méthodes approchées qui ne garantissent pas l'obtention d'une solution optimale mais, qui donnent de bons résultats dans des temps raisonnables. Pour finir, un problème spécifique de transport à la demande, le DARP, est présenté. C'est le sujet central de cette thèse.

Le DARP est un problème, incluant des contraintes sur la qualité de service proposée aux clients. Ce type de contrainte n'existe pas dans les autres problèmes de tournées de véhicules. Dans le DARP, ces contraintes sont importantes car elles correspondent à la qualité de service perçue par les usagers. De plus, ce type de transport s'adresse souvent à des personnes âgées ou à mobilité réduite, et donc, la qualité de service est primordiale.

Un grand nombre de publications ont été réalisées sur les problèmes de tournées dont le DARP. Toutefois, les articles consacrés aux problèmes statiques de tournées de véhicules avec des temps de trajet non-constants sont peu nombreux. Or, les variations des temps de trajet font partie intégrante du réseau routier et sont dues à de nombreuses causes (embouteillage, accident,…). Il est souvent très difficile de les prévoir à l'avance, en dehors des phénomènes périodiques. Mais avec le développement des nouvelles technologies telles que le GPS et l'accumulation des données concernant l'historique du trafic, dans quelque années, ces fluctuations seront peut-être estimées précisément et devront donc être prises en compte et intégrées dans les méthodes d'optimisation. C'est pour cela que dans cette thèse, premièrement l'intégration de temps de trajet variables en fonction de la période de temps a été réalisée sur un problème de VRP. Dans ce problème, les vitesses moyennes évoluent alors durant la journée. Deuxièmement, sur le DARP, les temps de trajet ont été modélisés par des variables aléatoires afin de prendre en compte les variations. Dans ce problème, la capacité des solutions à absorber ces aléas a été analysée puis des solutions robustes ont été générées.

Références

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. TOP 15, 1–31. doi:10.1007/s11750-007-0009-0

Braekers, K., Caris, A., Janssens, G.K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transp. Res. Part B Methodol. 67, 166–186. doi:10.1016/j.trb.2014.05.007

Brandão, J., 2011. A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. Comput. Oper. Res. 38, 140–151.

Chevrier, R., Liefooghe, A., Jourdan, L., Dhaenens, C., 2012. Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: Application to demand responsive transport. Appl. Soft Comput. 12, 1247–1258. doi:10.1016/j.asoc.2011.12.014

Choi, E., Tcha, D.-W., 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. Comput. Oper. Res. 34, 2080–2095.

Christofides, N., 1976. Worst-case analysis of a new heuristic for the travelling salesman problem. DTIC Document.

Cook, S.A., 1971. The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing. ACM, pp. 151–158.

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Ann. Oper. Res. 153, 29–46. doi:10.1007/s10479-007-0170-8

Cordeau, J.-F., Laporte, G., 2003a. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. Part B Methodol. 37, 579–594. doi:10.1016/S0191-2615(02)00045-0

Cordeau, J.-F., Laporte, G., 2003b. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. Q. J. Belg. Fr. Ital. Oper. Res. Soc. 1, 89–101. doi:10.1007/s10288-002-0009-8

Dantzig, G.B., Ramser, J.H., 1959. The Truck Dispatching Problem. Manag. Sci. 6, 80–91. doi:10.1287/mnsc.6.1.80

Duhamel, C., Lacomme, P., Prodhon, C., 2012. A hybrid evolutionary local search with depth first search split procedure for the heterogeneous vehicle routing problems. Eng. Appl. Artif. Intell., Special Section: Local Search Algorithms for Real-World Scheduling and Planning 25, 345–358. doi:10.1016/j.engappai.2011.10.002

Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. Eur. J. Oper. Res. 54, 7–22.

Eiselt, H.A., Gendreau, M., Laporte, G., 1995. Arc routing problems, part I: The Chinese postman problem. Oper. Res. 43, 231–242.

Felipe, Á., Ortuño, M.T., Righini, G., Tirado, G., 2014. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. Transp. Res. Part E Logist. Transp. Rev. 71, 111–128. doi:10.1016/j.tre.2014.09.003

Figliozzi, M.A., 2008. An iterative route construction and improvement algorithm for the vehicle routing problem with soft and hard time windows.

Frederickson, G., Hecht, M., Kim, C., 1978. Approximation Algorithms for Some Routing Problems. SIAM J. Comput. 7, 178–193. doi:10.1137/0207017

Gendreau, M., Potvin, J.-Y. (Eds.), 2010. Handbook of Metaheuristics, International Series in Operations Research & Management Science. Springer US, Boston, MA.

Goetschalckx, M., Jacobs-Blecha, C., 1989. The vehicle routing problem with backhauls. Eur. J. Oper. Res. 42, 39–51.

Golden, B., Assad, A., Levy, L., Gheysens, F., 1984. The fleet size and mix vehicle routing problem. Comput. Oper. Res. 11, 49–66.

Golden, B.L., Wong, R.T., 1981. Capacitated arc routing problems. Networks 11, 305–315.

Jaw, J.-J., Odoni, A.R., Psaraftis, H.N., Wilson, N.H.M., 1986. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transp. Res. Part B Methodol. 20, 243–257. doi:10.1016/0191-2615(86)90020-2

Jorgensen, R.M., Larsen, J., Bergvinsdottir, K.B., 2007. Solving the dial-a-ride problem using genetic algorithms. J. Oper. Res. Soc. 58, 1321–1331.

Karp, R.M., 1972. Reducibility among combinatorial problems. Springer.

Laporte, G., 1992. The vehicle routing problem: An overview of exact and approximate algorithms. Eur. J. Oper. Res. 59, 345–358.

Lenstra, J.K., Kan, A.H.G., 1981. Complexity of vehicle routing and scheduling problems. Networks 11, 221–227.

Lin, Y., Li, W., Qiu, F., Xu, H., 2012. Research on Optimization of Vehicle Routing Problem for Ride-sharing Taxi. Procedia - Soc. Behav. Sci., 8th International Conference on Traffic and Transportation Studies (ICTTS 2012) 43, 494–502. doi:10.1016/j.sbspro.2012.04.122

Li, X., Tian, P., Aneja, Y.P., 2010. An adaptive memory programming metaheuristic for the heterogeneous fixed fleet vehicle routing problem. Transp. Res. Part E Logist. Transp. Rev. 46, 1111–1127.

Luo, Y., Schonfeld, P., 2007. A rejected-reinsertion heuristic for the static Dial-A-Ride Problem. Transp. Res. Part B Methodol. 41, 736–755. doi:10.1016/j.trb.2007.02.003

Masson, R., Lehuédé, F., Péton, O., 2014. The Dial-A-Ride Problem with Transfers. Comput. Oper. Res. 41, 12–23. doi:10.1016/j.cor.2013.07.020

Paquette, J., Cordeau, J.-F., Laporte, G., 2009. Quality of service in dial-a-ride operations. Comput. Ind. Eng. 56, 1721–1734. doi:10.1016/j.cie.2008.07.005

Parragh, S.N., Doerner, K.F., Hartl, R.F., 2010. Variable neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 37, 1129–1138. doi:10.1016/j.cor.2009.10.003

Parragh, S.N., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 40, 490–497. doi:10.1016/j.cor.2012.08.004

Prodhon, C., Prins, C., 2014. A survey of recent research on location-routing problems. Eur. J. Oper. Res. 238, 1–17. doi:10.1016/j.ejor.2014.01.005

Psaraftis, H.N., 1980. A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. Transp. Sci. 14, 130.

Robinson, J., 1949. On the Hamiltonian game (a traveling salesman problem). DTIC Document.

Ropke, S., Cordeau, J.-F., 2009. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. Transp. Sci. 43, 267–286. doi:10.1287/trsc.1090.0272

Salhi, S., Osman, I.H., 1996. Local search strategies for the vehicle fleet mix problem.

Santos, D.O., Xavier, E.C., 2015. Taxi and Ride Sharing: A Dynamic Dial-a-Ride Problem with Money as an Incentive. Expert Syst. Appl. 42, 6728–6737. doi:10.1016/j.eswa.2015.04.060

Savelsbergh, M.W.P., Sol, M., 1995. The General Pickup and Delivery Problem. Transp. Sci. 29, 17–29. doi:10.1287/trsc.29.1.17

Sexton, T.R., Bodin, L.D., 1985. Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling. Transp. Sci. 19, 378.

Sexton, T.R., Choi, Y.-M., 1986. Pickup and delivery of partial loads with “soft” time windows. Am. J. Math. Manag. Sci. 6, 369–398.

Stein, D.M., 1978. Scheduling Dial-a-Ride Transportation Systems. Transp. Sci. 12, 232.

Taillard, É.D., 1999. A heuristic column generation method for the heterogeneous fleet VRP. Rev. Fr. Autom. Inform. Rech. Opérationnelle Rech. Opérationnelle 33, 1–14.

Toth, P., Vigo, D., 2002. Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discrete Appl. Math. 123, 487–512.

CHAPITRE 2

Résolution du problème de transport à la demande

Objectifs du chapitre :

Dans cette partie une heuristique est présentée pour la résolution du problème de transport à la demande. Les approches les plus couramment utilisées pour résoudre ce problème, qui appartient à la classe des problèmes $One\text{-}to\text{-}One$, sont des méthodes de type métaheuristiques. Un algorithme de ce type a été implémenté, il s'agit d'un algorithme de type "Evolutionary Local Search" (ELS).
Dans ce chapitre sont présentés les différents points clés de la méthode de résolution proposée. Parmi ces méthodes, une attention particulière est portée à la fonction d'évaluation et à la recherche locale combinée à un processus d'apprentissage.
Cette méthode de résolution est comparée avec les autres méthodes de la littérature sur les instances classiques. Les résultats sont, en moyenne, plus proches de la meilleure solution connue, pour des temps de calculs normalisés comparables à ceux des méthodes précédemment proposées.

2.1. Définition du problème

Le problème de transport à la demande (*Dial-A-Ride Problem* - DARP) a été introduit pour la première fois par (Stein, 1978) et la définition la plus utilisée est celle de (Cordeau and Laporte, 2003). L'appartenance et le positionnement de ce problème dans la famille des problèmes de tournées de véhicules ont été présentés dans le chapitre précédent.

Le problème est constitué de n clients et de m véhicules. Chaque client est en fait une requête de transport, associée à deux sommets : un sommet d'origine et un sommet de destination. La demande d'un client correspond alors au nombre de personnes à transporter.

Un ensemble de m tournées représente une solution si cet ensemble respecte un certain nombre de contraintes. Ces contraintes concernent :

- la capacité du véhicule ;
- l'ordre de passage du véhicule sur les sommets associés aux clients (*pickup* et *delivery*) ;
- le temps de trajet de chaque client dans la tournée ;
- les fenêtres de temps pour le début du service sur un sommet ;
- la durée maximale de la tournée.

L'objectif du problème consiste à minimiser la somme des distances parcourues par la flotte de véhicules pour transporter la totalité des clients, tout en respectant les contraintes précédentes.

2.1.1. Les notations et formulation linéaire du problème

Les notations utilisées dans la suite sont celles définies par (Cordeau and Laporte, 2003). A chaque client est associé deux sommets : un sommet d'origine et un sommet de destination. Les sommets d'origine sont numérotés dans $\{1, \dots, n\}$ et les sommets de destination sont numérotés dans $\{n+1, \dots, 2n\}$. Les sommets 0 et $2n+1$ correspondent au dépôt, respectivement pour le début d'une tournée et pour la fin d'une tournée. Les données du problème sont:

- n : le nombre de clients ;
- m : le nombre de véhicules ;
- $[e_x, l_x]$: la fenêtre de temps associée au sommet x ;
- d_x : le temps de service nécessaire pour ramasser ou déposer le client sur le sommet x;
- q_x : la charge que représente le client sur le sommet x. Cette charge est positive si le sommet x est un sommet d'origine d'un client et négative si x est un sommet de destination ;
- Q_k : la charge maximale que peut transporter le véhicule k ;
- c_{xy} : la durée/distance nécessaire pour que le véhicule aille du sommet x au sommet y ;
- L : la durée maximale pour transporter un client de son origine vers sa destination ;
- T_k : la durée maximale totale d'une tournée. Pour le DARP, la valeur est constante $\forall k$;
- T : la fin de l'horizon de planification.

Une solution notée S est constituée d'un ensemble $\{t_1, \dots, t_m\}$ de m tournées, où t_k est la $k^{ème}$ tournée de la solution avec $k = 1, \dots, m$. Pour chaque tournée t_k, un ensemble de variables sont nécessaires :

- n_k : le nombre de sommets dans la tournée k, incluant le dépôt en début et en fin de tournée ;
- s_i^k : le nœud à la $i^{ème}$ position dans la tournée k, avec $s_1^k = s_{n_k}^k = 0$ le dépôt ;
- A_i^k : la date d'arrivée du véhicule sur le $i^{ème}$ sommet de la tournée k ;
- B_i^k : la date à laquelle le véhicule commence son service sur le $i^{ème}$ sommet de la tournée k ;
- W_i^k : le temps d'attente du véhicule sur le $i^{ème}$ sommet de la tournée k. Il est obtenu comme $W_i^k = B_i^k - A_i^k$;
- D_i^k : la date de départ du véhicule du $i^{ème}$ sommet de la tournée k, implique $D_i^k = B_i^k + d_{s_i^k}$;
- L_i^k : le temps de trajet du client correspondant au $i^{ème}$ sommet de la tournée k dans le véhicule ;
- F_i^k : le retard maximal sur le début du service possible pour le véhicule sur le $i^{ème}$ sommet de la tournée k.

La formalisation linéaire du problème proposée par (Cordeau and Laporte, 2003) est présentée ici. La fonction objectif consiste à minimiser la distance parcourue par la flotte de véhicules.

$$min \sum_{k=1}^{m} \left(\sum_{i=1}^{n_k - 1} c_{s_i^k s_{i+1}^k} \right)$$

Les contraintes sont représentées par des inégalités qui doivent être vérifiées pour toutes les tournées. Pour simplifier les notations, l'indice k a été retiré des variables.

Sur le sommet initial, qui est le dépôt, la date de disponibilité correspond au début de la fenêtre de temps et le temps de service est considéré comme nul, d'où les expressions suivantes :

$$A_1 = e_0 \quad (1)$$

$$B_1 = D_1 \quad (2)$$

Pour tout sommet de la tournée k, le début du service est situé dans les fenêtres de temps :

$$e_p \leq B_i \leq l_p \text{ , } \forall i \in \{1, \dots, n_k\} \text{ et avec } p = s_i \quad (3)$$

A l'exception du dépôt, le départ du véhicule correspond au début du service plus le temps nécessaire pour installer les personnes dans le véhicule :

$$D_i = B_i + d_p \text{ , } \forall i \in \{2, \dots, n_k\} \text{ et avec } p = s_i \quad (4)$$

L'arrivée du véhicule sur un sommet autre que le dépôt initial est égale à la date du départ sur le sommet précédant plus le temps de trajet entre les deux sommets :

$$A_{i+1} = D_i + c_{pq} \text{ , } \forall i \in \{1, \dots, n_k - 1\} \text{ et avec } p = s_i \text{ et } q = s_{i+1} \quad (5)$$

Le temps de trajet du client entre son sommet de départ et son sommet de destination doit être inférieur ou égal au temps maximal pour transporter un client :

$$B_j \leq D_i + L, \forall i, j \text{ où } s_i \text{ et } s_j \text{ sont deux sommets associés à un même client} \quad (6)$$

Le temps de trajet du véhicule (temps entre la sortie et le retour au dépôt d'un véhicule) doit être inférieur ou égal à la durée maximale autorisée pour une tournée.

$$B_{n_k} \leq D_1 + T_k \quad (7)$$

Le temps d'attente du client est la différence entre son arrivée et le début du service :

$$W_i = B_i - A_i, \forall i \in \{1, \dots, n_k\} \quad (8)$$

La date de début de service sur un sommet est comprise entre son arrivée et son départ :

$$A_i \leq B_i \leq D_i, \forall i \in \{1, \dots, n_k\} \quad (9)$$

La charge du véhicule ne doit jamais dépasser la charge maximale du véhicule :

$$\sum_{p \in x_t} q_p \leq Q, \forall t \in \{1, \dots, n_k\} \quad (10)$$

avec $x_t = \{s_i | i \leq t \; et \; j > t\} \; \forall i, j$ où s_i et s_j sont les sommets de *pickup* et *delivery* associés au client.

La figure 2.1 représente le passage du véhicule sur deux sommets d'une même tournée afin d'illustrer les différentes notations utilisées. Les sommets s_i et s_j sont respectivement le sommet d'origine et de destination d'un même client dans une tournée. Le temps est représenté par l'axe horizontal. Le déplacement du véhicule sur les sommets est représenté par l'axe vertical. Le véhicule arrive sur le premier sommet s_i à la date A_i. Il attend ensuite une durée W_i pour commencer son service à la date B_i. Cette date B_i est située dans sa fenêtre de temps, entre e_i et l_i. La durée d_i représente le chargement du client puis le véhicule repart du sommet s_i à la date D_i et continue sa tournée. Lorsqu'il arrive sur le sommet s_j, qui est le sommet de destination associé au client s_i, la différence entre B_j et D_i représente le temps de trajet du client noté L_i.

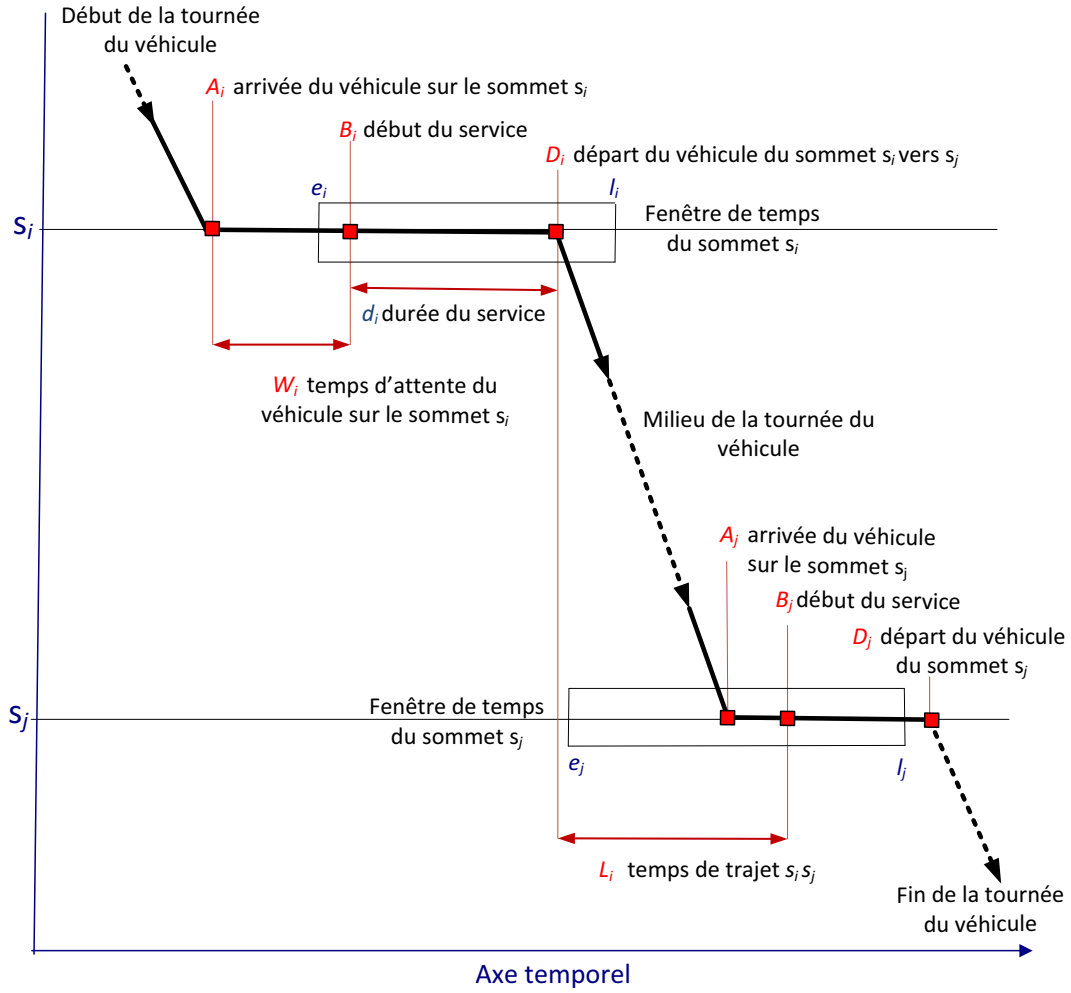


Figure 2.1: Notations utilisées pour le DARP

2.2. Proposition d'une approche de résolution du DARP

La méthode proposée pour la résolution du DARP est une métaheuristique. Elle repose sur un schéma de type "recherche locale évolutionnaire" (Evolutionary Local Search - ELS), qui a été introduit pour la première fois par (Wolf and Merz, 2007). ELS a été utilisé avec succès sur d'autres problèmes de tournées de véhicules comme par exemple dans (Prins, 2009) sur un problème de VRP. Ce schéma métaheuristique repose sur plusieurs points clés illustrés sur la figure 2.2 :

- la méthode d'évaluation d'un élément β qui, une fois évalué, correspond à une solution (figure 2.2 [1]) ;
- la méthode de génération de solutions initiales basée sur un espace de codage indirect (figure 2.2 [2]) ;
- la recherche locale pour améliorer une solution courante (figure 2.2 [3]) ;
- les opérateurs de diversification et le schéma permettant un parcourt efficace de l'espace des solutions (figure 2.2 [4]).

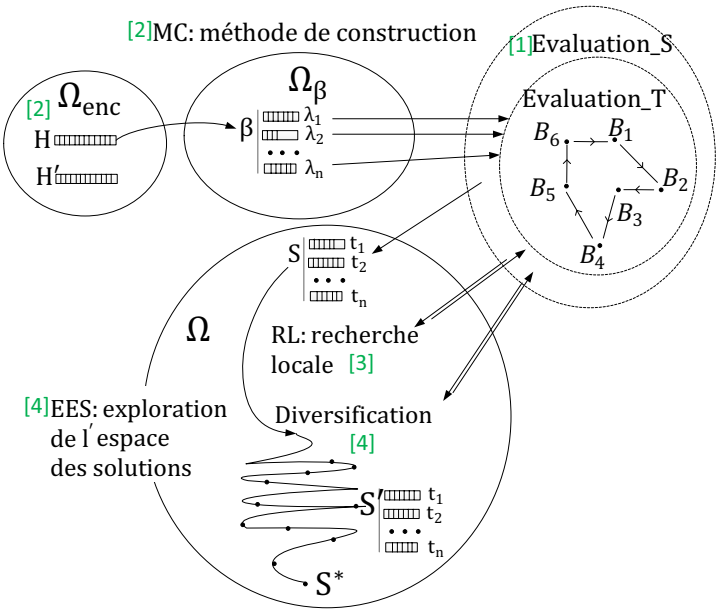


Figure 2.2: Schéma global de la méthode de résolution proposée

Ces différents points sont développés en détail dans les parties suivantes de ce chapitre. Dans un premier temps, la procédure permettant d'évaluer un élément β est décrite, et plus précisément celle qui évalue les éléments nommés λ_i avec $i \in \{1 \dots n\}$ qui composent cet élément β. Les vecteurs λ_i une fois évalués correspondent aux tournées de la solution (figure 2.2 [1]). Dans la littérature, plusieurs fonctions existent. La plus connue et la plus utilisée est celle proposée par (Cordeau and Laporte, 2003). Parmi les autres fonctions, une approche plus récente et de complexité linéaire par rapport au nombre de clients a été proposée par (Firat and Woeginger, 2011). A notre connaissance, cette dernière n'a jamais été utilisée en pratique dans un schéma global de résolution du DARP.

Dans un deuxième temps, la méthode de génération de solutions initiales est abordée (figure 2.2 [2]), elle utilise un espace de codage différent de l'espace des solutions. Cet espace de codage Ω_{enc} se compose d'un vecteur pour représenter l'ordre d'insertion des clients. Cet ordre est généré en utilisant les contraintes de fenêtres de temps associées aux sommets d'origine et de destination de chaque client. Une fois un élément $H \epsilon \Omega_{enc}$ sélectionné, une méthode de construction, notée MC dans la figure 2.2, l'associe à un élément de l'espace des solutions Ω. La fonction de construction est une méthode randomisée qui, pour le même espace de codage H, peut associer plusieurs solutions.

Le troisième point clé de l'ELS concerne la recherche locale (figure 2.2 [3]), qui permet d'améliorer itérativement une solution courante en parcourant des voisinages et en se déplaçant sur des voisins améliorant. Dans ce but, un ensemble de voisinages a été envisagé. Lors d'une recherche locale, ces voisinages ont des probabilités différentes d'être exécutés. Une contribution supplémentaire dans la recherche locale, par rapport à la littérature, concerne le processus d'apprentissage mis en place. Celui-ci permet de faire évoluer automatiquement ces différentes probabilités durant l'exécution.

Enfin l'exploration de l'espace Ω des solutions est abordée dans une quatrième partie, (figure 2.2 [4]). Elle est basée sur un schéma de "recherche locale évolutionnaire" (Evolutionary Local Search - ELS) qui alterne des phases de recherche locale et des phases de diversification. Ces dernières sont générées à partir de plusieurs mouvements et peuvent être considérées comme plus ou moins proche de la solution initiale.

2.3. L'évaluation d'un élément β

Une solution du problème du DARP se compose d'un ensemble de tournées. Une tournée est caractérisée par une liste de sommets pour lesquels des dates de début de service et autres variables de la tournée sont déterminées. Pour obtenir une solution, un objet de codage nommé β est évalué. Il se compose d'un ensemble de vecteurs d'entiers λ_k où $k \in \{1 \dots m\}$ est l'indice de la tournée. Ces entiers dans λ_k correspondent aux sommets d'origine et de destination qui doivent être visités par chaque véhicule.

Vérifier la réalisabilité d'un élément β nécessite de vérifier plusieurs contraintes sur β ainsi que sur les vecteurs λ_k qui le composent. Ces contraintes sont:

- La totalité des clients soit traitée ;
- Le nombre de λ_k dans la β est inférieur ou égal au nombre de véhicules disponibles dans la flotte ;
- Tous les λ_k sont réalisables.

L'évaluation de β pour le DARP est une première difficulté due aux différentes contraintes du problème. Les deux premières propriétés ne sont pas vérifiées à chaque évaluation car elles sont nécessairement respectées dans le schéma de résolution utilisé. Par contre la difficulté réside dans la vérification de la validité des λ_k.

L'évaluation permet aussi d'obtenir le coût d'une solution. Pour les instances classiques du DARP, ce coût correspond à la distance parcourue par la flotte de véhicules. Il peut être calculé en temps linéaire par rapport au nombre de clients en parcourant successivement les arcs qui composent les tournées. Mais il existe d'autres versions du DARP pour lesquelles la fonction de coût est plus complexe. Par exemple (Jorgensen et al., 2007) incluent dans le coût d'autres critères comme le temps de trajet des clients ou encore leur temps d'attente avant d'être servis.

Lors de l'évaluation d'un vecteur λ, plusieurs dates de début de service sont possibles sur les sommets. C'est pourquoi, dans la littérature, plusieurs fonctions d'évaluation ont été proposées. La première a été proposée par (Cordeau and Laporte, 2003). C'est la fonction qui est utilisée dans les métaheuristiques les plus efficaces de la littérature. Elle est appelée "*eight-step evaluation*" par (Parragh et al., 2010).

Sa complexité est en $O({n_k}^2)$ où n_k est le nombre de sommets dans le vecteur λ. Une autre fonction a été proposée par (Hunsaker and Savelsbergh, 2002) pour un problème de transport à la demande avec une contrainte supplémentaire sur le temps d'attente.

Cette fonction, annoncée de complexité linéaire par ses auteurs, ne permet cependant pas de résoudre certaines instances comme l'ont démontré (Tang et al., 2010). Ces derniers ont aussi proposé une correction dégradant la complexité de l'algorithme en $O({n_k}^2)$. Enfin dans (Firat and Woeginger, 2011), les auteurs proposent un test de réalisabilité avec une complexité linéaire en temps par rapport au nombre n_k de sommets dans la tournée. Pour démontrer la linéarité en temps, ils se sont ramenés à un cas particulier du problème de plus court chemin dans un graphe d'intervalle avec des poids positifs sur les arcs. Ce cas particulier a été montré solvable en temps linéaire par (Atallah et al., 1995).

2.3.1. Schéma général d'évaluation d'un élément β

a. Evaluation d'un élément β

L'algorithme "$evaluation\_S$", présenté dans l'algorithme 1, prend en entrée un ensemble de vecteurs d'entiers (ici symbolisé par S), avec $S[i]$ qui représente l'élément λ_i de β. Pour chaque λ_i, parcouru à l'aide d'une boucle "for" allant de la ligne 3 à la ligne 9, l'algorithme fait appel à une procédure "$evaluation\_T$" à la ligne 4. Cette procédure est cruciale pour les performances finales de la méthode de résolution car c'est la fonction la plus appelée.

Elle permet d'obtenir le coût associé à un λ ainsi que de vérifier sa validité. Ensuite, l'algorithme 1 fait la somme des coûts de chacune des tournées qui composent la solution, étape effectuée à la ligne 5. Si un des λ n'est pas valide alors la variable $violationS$ reçoit la valeur $true$ (lignes 6 à 8).

Algorithme 1 $evaluation\_S$

Input parameters
$S[]$: Set of trips

Output parameters
$cout$: value of the solution
$violationS$: boolean with $false$→ valid solution and $true$→ non-valid solution

Begin
1 $cout$:=0
2 $violationS$:= false
3 **for** i := 1 **to** card(S) **do**
4 | {$violation, cout\_t$} = **call** $evaluation\_T$ ($S[i]$)
5 | $cout$:= $cout$ + $cout\_t$
6 | **if** ($violation$ = true) **then**
7 | | $violationS$ = true
8 | **end if**
9 **end for**
10 **return** {$violationS, cout$}
End

b. Evaluation *d'un élément λ*

Concernant l'évaluation d'un λ, il est considéré valide si et seulement si les cinq propriétés suivantes sont respectées:

- la charge transportée par le véhicule est inférieure ou égale à la charge maximale tout le long de λ ;
- les sommets d'origine et de destination d'un client doivent être dans λ et le sommet d'origine doit se situer avant le sommet de destination ;
- la date de début du service sur chaque sommet est située dans sa fenêtre de temps ;
- le temps de trajet du véhicule ne dépasse pas la durée maximale T_k associée au véhicule k ;
- le temps de transport des clients est inférieur à la valeur maximale L.

L'algorithme "$evaluation\_T$" présenté dans l'algorithme 2 procède en deux parties. La première, en $0(n)$, parcourt les clients de λ et calcule le coût ainsi que le respect de la charge. Cette partie se situe entre la ligne 4 et la ligne 12 de l'algorithme. Puis, dans un deuxième temps et si la contrainte de charge est vérifiée, la seconde partie vérifie la réalisabilité de l'élément λ de la ligne 13 jusqu'à la ligne 15.

Cette seconde partie est responsable de la vérification de la réalisabilité d'un élément λ concernant les autres contraintes non vérifiées qui sont :

- les fenêtres de temps sur les sommets ;
- la durée de trajet du véhicule ;
- le temps de transport des clients.

Algorithme 2 evaluation_T

Input parameters

t : a trip which contains all the trip variables like : n_t, s[], B[], A[], D[]

Global parameters

Q : capacity of a vehicle
c_{ij} : distance between node i and j
q_i : demand on node i

Output parameters

$cout$: value of the trip
$violation$: boolean with $true$ associated with a non-valid trip
$false$ associated with a valid trip

Begin

1 $cout$:= 0
2 $load$:= 0
3 $violation$:= false
4 **for** i := 2 **to** $t.n_t$ **do**
5 | $x := t.s[i-1]$
6 | $y := t.s[i]$
7 | $cout := cout + c_{xy}$
8 | $load := load + q_x$
9 | **if** load > Q **then**
10 | | $violation$:= true
11 | **end if**
12 **end for**
13 **if** ($violation$ = false) **then**
14 | {$violation$} := **call** is_feasible_tour(t)
15 **end if**
16 return {$violation, cout$}

End

Pour vérifier que ces contraintes sont valides, la méthode consiste à trouver un ensemble de valeurs à affecter aux B_i tel que toutes les contraintes soient vérifiées.

c. Impact *de la fonction d'évaluation d'un élément λ*

Pour mettre en évidence les différences qui existent entre les différentes fonctions d'évaluation de λ, un exemple a été mis en place. Les données de cet exemple sont illustrées sur la figure 2.3. Le vecteur λ est composé de 4 sommets : le dépôt initial, le sommet d'origine du client A, le sommet de destination de A, puis le dépôt final. Pour chaque sommet, des fenêtres de temps sont définies sur la figure.

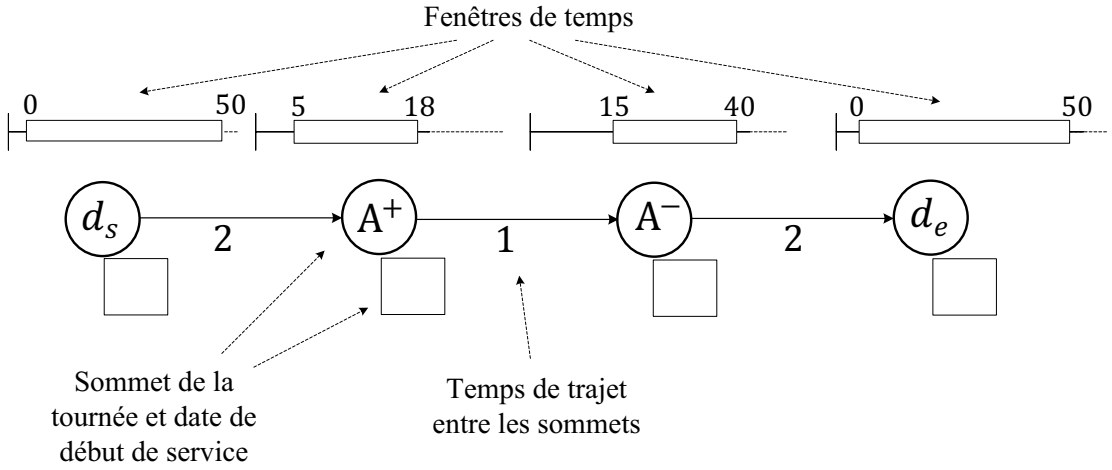


Figure 2.3: Graphe initial

Pour ce vecteur λ plusieurs tournées valides peuvent être générées. La première, illustrée dans la figure 2.4, est construite à l'aide des dates au plus tôt. Les caractéristiques associées à la tournée générée sont une durée totale de 17 et un temps de trajet pour le client *A* de 10.

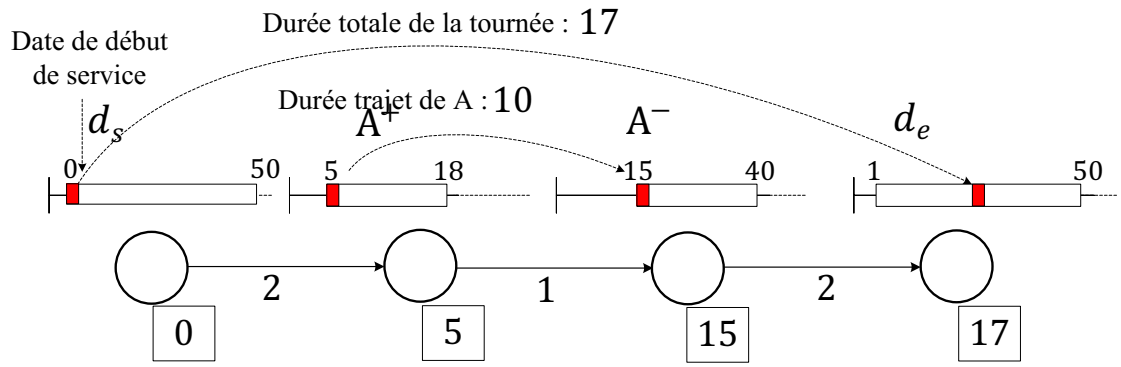


Figure 2.4: Evaluation des dates au plus tôt

Une seconde tournée peut être générée à l'aide cette fois des dates au plus tard, comme illustré sur la figure 2.5. Les caractéristiques associées sont une durée totale de 34 et un temps de trajet pour le client *A* de 22. Sur cet exemple, la seconde évaluation est donc moins intéressante que la première, obtenue avec les dates au plus tôt.

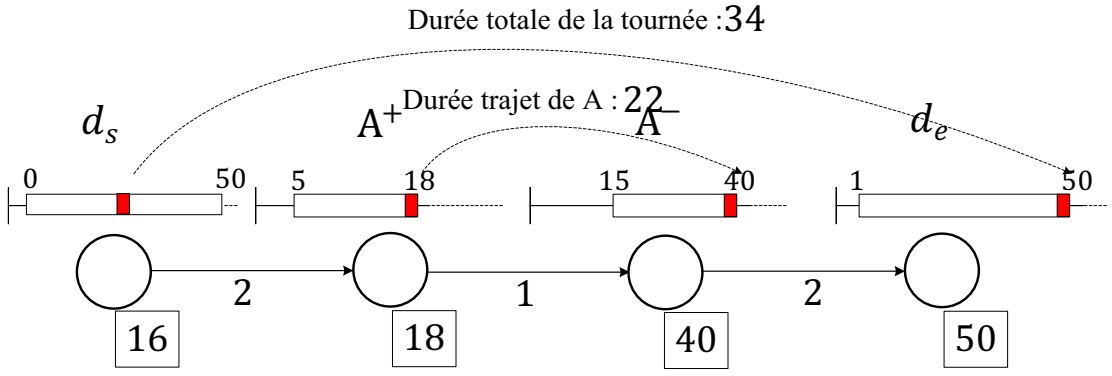


Figure 2.5: Evaluation des dates au plus tard

Il existe donc un certain degré de liberté dans le choix des dates de début de service. C'est pour cela que plusieurs fonctions d'évaluation spécifiques au DARP ont été développées afin d'obtenir une tournée valide qui possède en plus un certain nombre de propriétés intéressantes. Une de ces fonctions est proposée par (Cordeau and Laporte, 2003) et elle est illustrée en figure 2.6. Les caractéristiques de cette tournée sont une durée totale de 5 et un temps de trajet pour le client *A* de 1. Cette évaluation vise à définir des dates telles que la tournée finisse au plus tôt tout en minimisant sa durée totale. Elle permet aussi, dans une certaine mesure, de réduire les temps de trajet des clients ainsi que leurs temps d'attente.

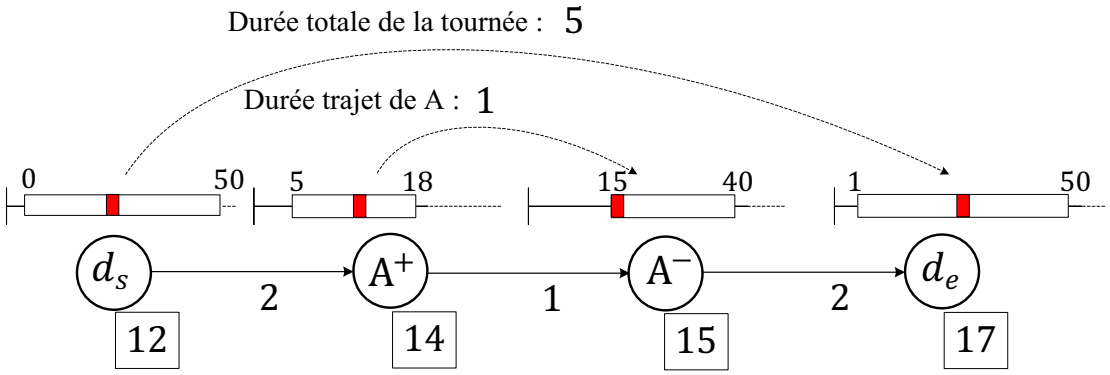


Figure 2.6: Evaluation des dates par (Cordeau and Laporte, 2003)

La somme des distances est égale à 5. C'est le coût de cette tournée, quelles que soient les dates B_i définies pour chaque client i. Pour un même λ, il peut donc exister plusieurs B_i possibles qui conduisent à respecter les contraintes. C'est pourquoi plusieurs fonctions ont été proposées dans la littérature pour tester la réalisabilité de λ. Ces fonctions donnent des valeurs différentes aux B_i pour un même λ et donc, d'une certaine manière, des tournées effectives différentes.

d. Etude *sur les fonctions d'évaluation pour DARP*

Dans la littérature, il existe différentes fonctions d'évaluation de λ. Elles sont regroupées dans le tableau 2.1, où cinq colonnes sont présentées :

- le nom de la méthode ;
- la complexité de la méthode ;
- si la méthode retourne la tournée avec les dates au plus tôt ou au plus tard ;
- si la méthode retourne une tournée de durée totale minimale (min. TD : Minimum Total Duration) ;
- si la méthode retourne une tournée avec des temps de trajet minimaux pour les clients (min. TRT : Minimum Total Riding Time).

Tableau 2.1:
les différentes fonctions d'évaluation de λ

méthode	compl.	plus tôt / plus tard	min. TD	min. TRT
Ordonnancement	$O(n^2)$	X		
(Cordeau and Laporte, 2003)	$O(n^2)$	"X"	X	"X"
(Firat and Woeginger, 2011)	$O(n)$	X		

La première est une fonction issue des problèmes d'ordonnancement. Elle utilise la modélisation sous la forme de graphe conjonctif afin d'évaluer la réalisabilité de λ en utilisant des algorithmes de plus court (ou plus long) chemin. Cette fonction affecte soit des dates au plus tôt soit des dates au plus tard sur tous les sommets de la tournée. Cette modélisation sous la forme de graphe est basée sur des contraintes dites "contraintes de précédence" qui sont définies dans une sous-partie.

La deuxième fonction est celle proposée par (Cordeau and Laporte, 2003) qui est elle-aussi basée sur un graphe conjonctif. Elle tire parti des caractéristiques spécifiques au DARP pour améliorer les performances de l'algorithme et permettre d'obtenir des dates qui donnent des caractéristiques intéressantes à la tournée, notamment celle de minimiser la durée totale. Ces dates réduisent aussi le temps de trajet des clients pour certains λ. Pour finir cette fonction fournit aussi le retard maximal sur les sommets, donc les dates au plus tard.

La troisième fonction est proposée par (Firat and Woeginger, 2011). Contrairement aux deux précédentes fonctions, elle permet de vérifier la réalisabilité de λ en complexité linéaire. Dans la version proposée par les auteurs, seules les dates au plus tard sont retournées. Avec quelques transformations, il est possible d'obtenir les dates au plus tôt et donc obtenir des tournées aux caractéristiques proches de celles proposées par (Cordeau and Laporte, 2003).

Ceci permet de comprendre les différences et les avantages de la fonction basée sur (Firat and Woeginger, 2011). Les dates proposées ont un réel impact sur la qualité du service proposé aux clients, comme cela a été illustré sur les exemples. Donc, même si le coût d'une tournée pour le DARP est constitué uniquement de la distance parcourue par les véhicules, il y a un réel intérêt à s'intéresser aux autres critères d'évaluation.

Dans la méthode de résolution proposée, la fonction d'évaluation est basée sur celle proposée par (Firat and Woeginger, 2011) qui a une complexité linéaire. Cette fonction est modifiée afin d'obtenir les mêmes valeurs pour les B_i que celles fournies par la fonction de (Cordeau and Laporte, 2003). Avant de s'intéresser à ces deux fonctions, la partie suivante présente les contraintes de précédence ainsi que la construction du graphe conjonctif. Ce sont les objets sur lesquels repose l'affectation des valeurs des B_i.

2.3.2. Modélisation des éléments λ

Les graphes disjonctifs sont les modélisations les plus utilisées dans la littérature pour les problèmes d'ordonnancement. Ils ont été introduits par (Roy and Sussmann, 1964) et se composent d'arcs conjonctifs qui représentent les contraintes de précédence entre les opérations et d'arcs disjonctifs qui représentent le séquencement sur les machines. Lorsqu'un séquençage sur les machines est déterminé, le graphe se compose uniquement d'arcs conjonctifs et est appelé graphe conjonctif. Ces graphes peuvent être adaptés pour l'évaluation du vecteur λ associé à un problème de DARP. En ordonnancement, l'utilisation de ces graphes permet d'affecter des valeurs minimales et maximales aux dates de début des opérations qui composent le problème. Ces dates sont affectées de telle sorte que l'ensemble des contraintes représentées dans le graphe sous la forme d'arcs sont satisfaites. Ces arcs modélisent des contraintes de précédence de différents types parmi lesquelles figurent les *time-lags*, qui définissent le retard maximal et minimal entre deux opérations. Ces contraintes sont clairement définies par plusieurs auteurs notamment (Caumond et al., 2008) où (Brinkmann and Neumann, 1996). Elles peuvent poser des difficultés car elles engendrent des cycles dans le graphe conjonctif. Pour les problèmes d'ordonnancement, la présence de cycles de coût positif dans le graphe lors de l'évaluation empêche l'obtention d'un planning réalisable. Ces

contraintes sont aussi présentes dans les tournées associées au DARP notamment avec les contraintes de fenêtres de temps.

Dans la suite de cette partie, il est mis en évidence le lien entre les contraintes associées aux tournées du DARP et les contraintes de *time-lag* entre opérations dans les problèmes d'ordonnancement. Ainsi les différentes fonctions efficaces d'évaluation développées à l'origine pour les problèmes d'ordonnancement pourront être adaptées.

a. Les contraintes de précédence

Les contraintes de précédence sont définies par des inégalités de la forme suivante :

$$X + cst \leq Y$$

Avec X et Y deux variables du problème et cst une constante. Ces contraintes sont utilisées pour les problèmes de recherche opérationnelle, notamment pour les problèmes d'ordonnancement. Elles sont appelées contraintes de précédence car, en considérant X et Y des dates de début d'opération, elles symbolisent le fait que la date de début de l'opération Y doit être supérieure ou égale à la date de début de l'opération X plus une constante. Cette constante correspond, pour les problèmes d'ordonnancement, à la durée de l'opération.

Ces contraintes peuvent être représentées sous la forme d'arcs comme introduit par (Roy and Sussmann, 1964). La figure 2.7 illustre la représentation associée à la contrainte de précédence générale : $X + cst \leq Y$. Les variables X et Y sont représentées par des labels associés aux sommets s_x et s_y. Ces sommets sont reliés par un arc de s_x vers s_y et de valeur cst.

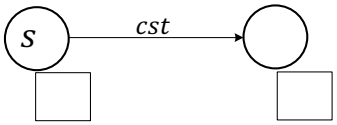


Figure 2.7: Représentation sous la forme d'arc d'une contrainte de précédence

Certaines contraintes appliquées au DARP peuvent être réécrites sous la forme de contraintes de précédence. La partie suivante concerne cette réécriture ainsi que la modélisation sous la forme d'un graphe associé à chacune de ces contraintes.

b. Réécriture des contraintes du DARP

Dans cette partie, les contraintes du DARP présentées dans la partie précédente sont réinterprétées afin de les exprimer sous la forme de contraintes de précédence. Une fois les contraintes de précédence mises en évidence, la représentation sous forme de graphe est introduite en utilisant l'exemple d'un tour λ composé de deux clients, un client devant aller du nœud 1 au nœud 3; un client devant aller du nœud 2 au nœud 4.

Un graphe initial, composé uniquement des sommets et des labels, est introduit figure 2.8. Ce graphe est composé d'un sommet pour chaque élément de λ. Le label associé à chaque sommet représente la date de début de service qui doit être définie.

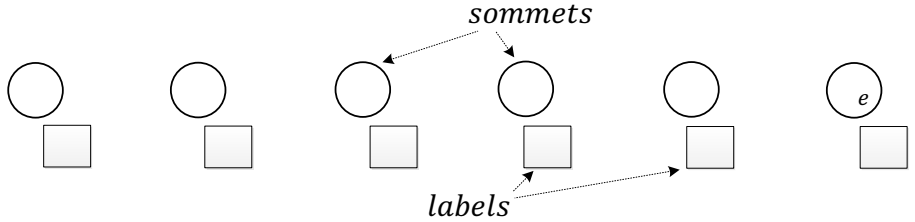


Figure 2.8: Graphe initial

A ce graphe initial sont ajoutés plusieurs types d'arcs représentant les contraintes introduites dans la partie 2.1.1. Dans la définition du DARP, ces contraintes sont exprimées entre variables (comme, par exemple, les dates d'arrivée A_i, les dates de départ D_i et les dates d'attente W_i). Elles doivent par conséquent être modifiées pour s'exprimer uniquement en fonction des variables B_i(dates de début de service) et être utilisées comme labels dans le graphe conjonctif. La première contrainte de précédence est déduite des contraintes (4), (5), (8) et (9). Elle permet de représenter une contrainte générale sur le temps de trajet et de début service des clients. Cette contrainte de précédence est numérotée (5b), sa représentation graphique associée est illustrée en figure 2.9. La démonstration pour obtenir cette contrainte est détaillée ci-dessous :

Contraintes initiales :

$$D_i = B_i + d_p \text{ , } \forall i = \{2, \dots, n_k\} \text{ et } p = s_i \quad (4)$$

$$A_{i+1} = D_i + c_{pq} \text{ , } \forall i = \{1, \dots, n_k - 1\} \text{ et } p = s_i \text{ et } q = s_{i+1} \quad (5)$$

$$W_i = B_i - A_i \text{ , } \forall i = \{1, \dots, n_k\} \quad (8)$$

$$A_i \leq B_i \leq D_i \text{ , } \forall i = \{1, \dots, n_k\} \quad (9)$$

Démonstration :

d'après (5) : $A_{i+1} = D_i + c_{i\,i+1}, \forall i = \{1, \dots, n_k - 1\}$ et $p = s_i$ et $q = s_{i+1}$

$\rightarrow A_{i+1} = (B_i + d_p) + c_{pq}, \forall i = \{1, \dots, n_k - 1\}$ avec (4) (5a)

d'après (8) : $W_{i+1} = B_{i+1} - A_{i+1}, \forall i = \{1, \dots, n_k - 1\}$

$\rightarrow A_{i+1} = B_{i+1} - W_{i+1}, \forall i = \{1, \dots, n_k - 1\}$ (8a)

d'après (9) : $A_i \leq B_i, \forall i = \{1, \dots, n_k\}$

$\rightarrow B_i - A_i \geq 0, \forall i = \{1, \dots, n_k\}$

$\rightarrow W_i \geq 0 \text{ , } \forall i = \{1, \dots, n_k\}$ avec (8) (9a)

d'après (5a) $A_{i+1} = (B_i + d_p) + c_{pq}, \forall i = \{1, \dots, n_k - 1\}, p = s_i \text{ , } q = s_{i+1}$

$\rightarrow B_{i+1} - W_{i+1} = B_i + d_p + c_{pq} \text{ , } \forall i = \{1, \dots, n_k - 1\}$ avec (8a)

$\rightarrow B_{i+1} \geq B_i + d_p + c_{pq}, \forall i = \{1, \dots, n_k - 1\}$ avec (9a)

$\rightarrow \boxed{B_i + (d_p + c_{pq}) \leq B_{i+1}} \text{ , } \forall i = \{1, \dots, n_k - 1\}$ (5b)

Représentation :

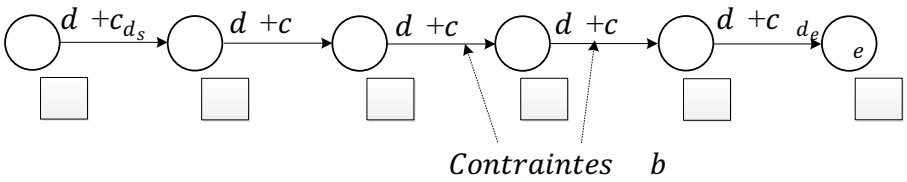


Figure 2.9: Graphe après l'ajout des contraintes (5b)

La seconde contrainte est la contrainte (6) qui modélise le temps de trajet maximal pour transporter un client de son sommet origine à son sommet destination. Le temps de trajet est mesuré entre la date de départ du véhicule du sommet d'origine et la date de début de service sur le sommet de destination. Cette contrainte peut s'exprimer sous la forme (6a) à partir de la

contrainte (4). C'est une contrainte de précédence dans laquelle la différence entre les deux dates de début de service a une valeur négative $-(d_p + L)$. La représentation graphique associée, illustrée sur la figure 2.10, est un arc de poids négatif, allant de la destination vers la source. Des contraintes similaires sont utilisées dans les problèmes d'ordonnancement pour représenter les délais maximaux entre les opérations, comme expliqué dans (Caumond et al., 2008).

Contraintes initiales :

$B_j \leq D_i + L \ \forall i, j$ deux sommets d'un même client (6)

$D_i = B_i + d_p \ , \forall i = \{2, \dots, n_k\}$ et avec $p = s_i$ (4)

Démonstration :

d'après (6) $B_j \leq D_i + L \ \forall i, j$ deux sommets d'un même client

$\rightarrow B_j \leq B_i + d_p + L, \forall (i, j)$ et avec $p = s_i$ d'après (4)

$\boxed{\rightarrow B_j - (d_p + L) \leq B_i}$ (6a)

Représentation :

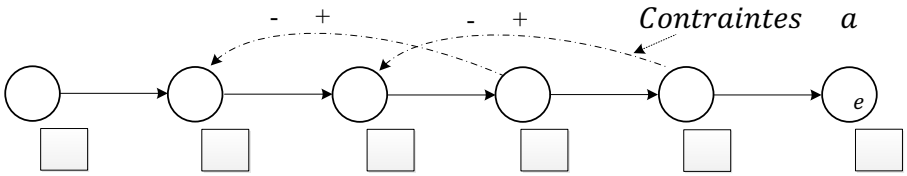


Figure 2.10: Graphe après l'ajout des contraintes (6a)

La contrainte (7) concerne le temps de conduite maximal pour le chauffeur. Elle est similaire aux contraintes sur le temps de trajet maximal des clients. Elle peut s'exprimer sous la forme (7a) à partir de la contrainte (2). Sa représentation sous la forme de graphe, illustrée figure 2.11, est un arc de coût négatif $-T_k$ du dépôt de retour vers le dépôt de départ.

Contraintes initiales :

$B_{n_k} \leq D_1 + T_k$ (7)

$B_1 = D_1$ (2)

Démonstration :

$B_{n_k} \leq D_1 + T_k$ d'après (7)

$\rightarrow B_{n_k} \leq B_1 + T_k$, d'après (2)

$\boxed{\rightarrow B_{n_k} - T_k \leq B_1}$ (7a)

Représentation :

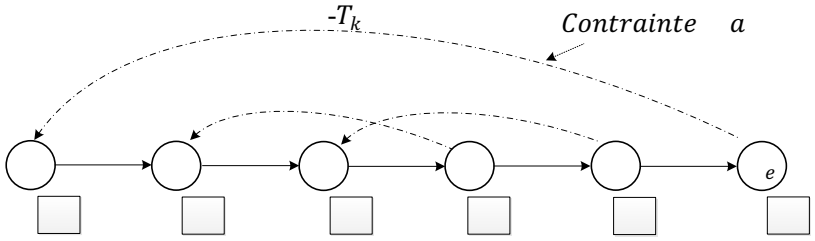


Figure 2.11: Graphe après l'ajout de la contrainte (7a)

La dernière contrainte qui peut être associée à une contrainte de précédence est la contrainte (3) qui représente les fenêtres de temps. Elle comporte deux parties, une première concernant la borne inférieure et une seconde concernant la borne supérieure. En dehors de ces bornes, le service sur le sommet n'est pas possible. Pour modéliser ces contraintes, les dates de service sont vues comme des durées minimales et maximales de retard après une certaine date. Ces durées sont communément appelées *time-lags* dans les problèmes d'ordonnancement. La date de référence choisie est la date 0 qui symbolise le début de la journée. La variable B_0 est alors introduite afin de transformer la contrainte (3) en deux contraintes de précédence (3a) et (3b). Dans la représentation sous la forme de graphe, figure 2.12, la variable fictive est associée à un sommet fictif noté 0. Les contraintes (3a) de fenêtre de temps minimale sont des arcs qui partent du sommet 0 vers les autres sommets du graphe. Les contraintes (3b) de fenêtre de temps maximale sont des arcs qui partent des différents sommets du graphe et qui finissent sur le sommet 0.

Contrainte initiale :

$$e_{s_i} \leq B_i \leq l_{s_i}\ , \forall i = \{1, \dots, n_k\} \qquad (3)$$

Démonstration :

d'après (3) : $e_p \leq B_i$, $\forall i = \{1, \dots, n_k\}$ et avec $p = s_i$

$B_i - B_0 \geq e_p - 0$, avec $p = s_i$ et $\forall i = \{1, \dots, n_k\}$

$\boxed{\rightarrow B_0 + e_{p,} \leq B_i}$ avec $p = s_i\ et\ \forall i = \{1, \dots, n_k\}$ (3a)

d'après (3) : $B_i \leq l_p$ avec $p = s_i$ et $\forall i = \{1, \dots, n_k\}$

$B_i - B_0 \leq l_p$ avec $p = s_i$ et $\forall i = \{1, \dots, n_k\}$

$\boxed{\rightarrow B_i - l_p \leq B_0,}\ \forall i = \{1, \dots, n_k\}$ et $p = s_i$ (3b)

Représentation :

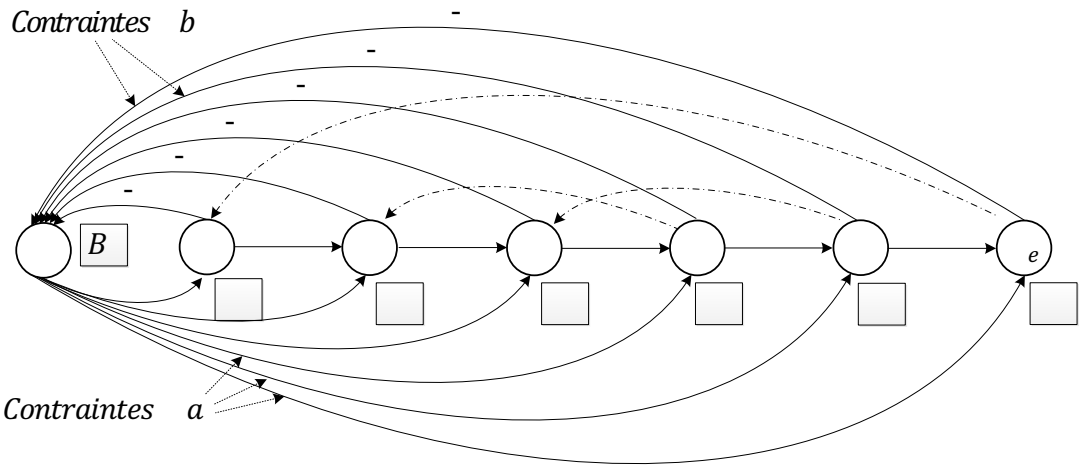


Figure 2.12: Graphe après l'ajout des contraintes (3a) et (3b)

La totalité des contraintes d'une tournée à vérifier dans la seconde partie de l'algorithme 2 (évaluation d'un élément λ) s'exprime sous la forme de contraintes de précédence. Ces

contraintes de précédence sont toutes exprimées en fonction des dates de début de service B_i sur les sommets de λ.

a) Représentation générale des contraintes de précédence

Dans la partie précédente, il a été mis en évidence que toutes les contraintes peuvent être exprimées sous la forme de contraintes de précédence. Il est donc possible de construire un graphe conjonctif, reposant sur les principes introduits par (Roy and Sussmann, 1964). Dans notre cas, le graphe est associé à λ. A partir de ce graphe, l'évaluation visera à affecter une valeur pour chaque sommet de telle manière que l'ensemble des contraintes de précédence soient satisfaites.

Pour l'affectation des B_i à un λ, la construction du graphe conjonctif se fait de la manière suivante :

- le sommet fictif (noté 0) sommet sert de sommet de référence pour les fenêtres de temps et pour le début des opérations ;
- à chaque sommet de λ est associé un sommet de même numéro dans le graphe G. Le dépôt apparaît deux fois dans λ. Il apparaît donc aussi deux fois dans le G. Pour les distinguer, le nœud d_s est utilisé pour le début (*depot start*) et le nœud d_e pour la fin (*depot end*) de la tournée ;
- chaque contrainte de précédence de la forme $B_i + cst \leq B_j$ est associée à un arc orienté de s_i vers s_j dans G et de valeur cst. Les sommets s_i et s_j sont associés respectivement au $i^{\text{ème}}$ et $j^{\text{ème}}$ sommet de λ.

De plus, à chaque sommet s_i est associé un label qui représente la date de début de service, *i.e.* la variable B_i. Le label du sommet fictif reçoit la valeur 0. Elle est nécessaire pour que les contraintes de fenêtre de temps soient valides. Ce graphe est noté G dans la suite de ce chapitre. La figure 2.13 donne une représentation générale de G associée à un problème d'affectation des B_i pour un vecteur λ composé de n_k éléments.

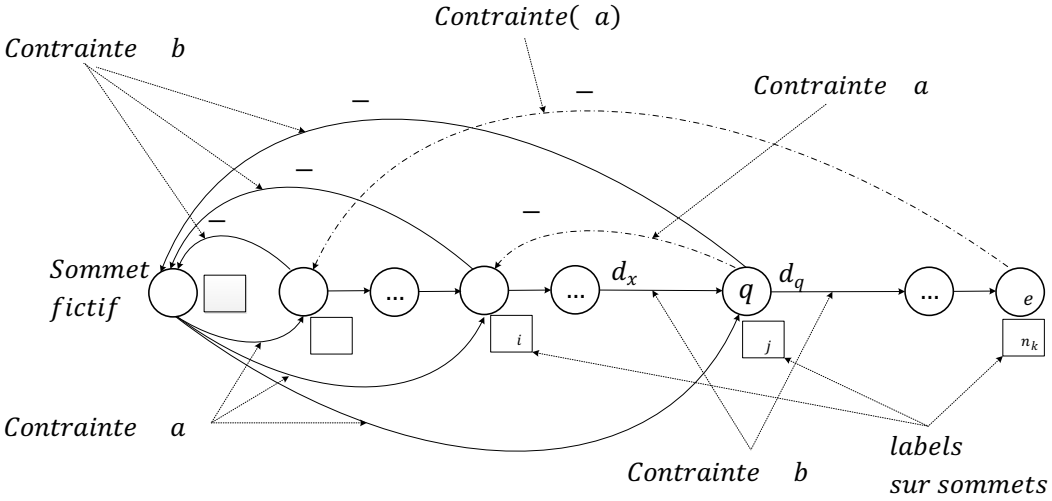


Figure 2.13: Construction générale du graphe G

A chaque sommet non fictif du graphe conjonctif sont associés 5 arcs entrants ou sortants :

- un arc entrant venant du sommet fictif 0 exprimant la fenêtre de temps minimal (3a) et de coût positif ;
- un arc sortant vers le sommet fictif 0 exprimant la fenêtre de temps minimal (3b) et de coût négatif ;

- un arc entrant exprimant le temps minimal entre lui et le sommet qui le précède dans λ (5b) ;
- un arc sortant exprimant le temps minimal entre lui et le sommet qui le suit dans λ (5b) ;
- un arc entrant (resp. sortant) vers le sommet de destination du client (resp. sommet d'origine) si un sommet d'origine est considéré (resp. de destination). Cet arc représente la contrainte de temps de trajet minimal (6a). S'il correspond au sommet de dépôt, l'arc est alors orienté du dépôt final vers le dépôt initial et représente la contrainte sur le temps de conduite minimal pour le véhicule (7a).

L'évaluation du graphe G revient à calculer les plus courts (ou plus longs) chemins depuis un sommet initial vers tous les autres sommets. Les valeurs des labels correspondent aux B_i. Le sommet initial utilisé est le sommet fictif 0 dont on a dit que le label vaut 0 pour que les contraintes $(3a)$ et $(3b)$ soient valides.

Comme expliqué dans (Caumond et al., 2008) certaines contraintes de précédence peuvent engendrer la création de cycles de poids négatif. Dans ce cas, il n'existe pas de plus court (ou plus long) chemin entre le sommet initial et les autres sommets du graphe et par conséquent il n'existe pas de valeur valide à associer aux variables. Cela veut alors dire que λ ne correspond pas à une tournée réalisable au sens des contraintes de temps. L'évaluation doit être en mesure de détecter la présence de tels cycles.

2.3.3. Test de réalisabilité d'un vecteur λ par calcul des dates au plus tôt / au plus tard

A partir du graphe G défini précédemment, une première fonction d'évaluation de la réalisabilité d'un λ est présentée. A notre connaissance, cette fonction n'a jamais été publiée et elle permet surtout d'introduire les fonctions d'évaluation suivantes car elles reposent toutes sur les mêmes principes. Grace aux travaux de (Roy and Sussmann, 1964) sur des problèmes d'ordonnancement et à la modélisation précédente sous forme de graphe, il est non seulement possible de savoir s'il existe ou non une solution valide mais il est également possible d'affecter des valeurs minimales et maximales aux labels tels que toutes les contraintes soient satisfaites. Les deux parties suivantes illustrent la fonction utilisée pour les problèmes d'ordonnancement. Cette fonction peut être appliquée directement pour évaluer la réalisabilité d'un vecteur λ.

a. Calcul des dates au plus tôt

En ordonnancement, les solutions recherchées sont des solutions dites semi-actives dans lesquelles les dates de début des opérations sont positionnées au plus tôt. Calculer les dates au plus tôt des opérations revient à commencer l'opération courante à l'instant où les opérations qui la précèdent sont toutes terminées. Dans G cela revient à trouver les labels B_i tel que :

$$B_i = \max_{j \in \delta^-_{s_i}} \left(B_j + cst(s_j, s_i) \right) \quad \forall i \in \{1, \dots, n_k\}$$

où $\delta^-_{s_i}$ est l'ensembles des arcs entrants sur s_i et $cst(s_j, s_i)$ est la valeur de l'arc de s_j à s_i.

Dans G, le label du sommet fictif 0 est initialisé à 0 afin de modéliser les contraintes de fenêtre de temps (3a) et (3b) comme expliqué précédemment. Il sert de point de départ pour propager la formule ci-dessus dans G. En théorie des graphes, cette formule permet de calculer le plus long chemin. Elle est donc équivalente à calculer la valeur plus long chemin depuis le sommet 0 vers tous les sommets du graphe.

Un plus long chemin existe uniquement si le graphe ne contient pas de cycle positif. L'algorithme Bellman-Ford peut être utilisé. Il a une complexité de $O(nm)$ avec n le nombre de sommets dans G et m le nombre d'arcs dans G. Comme le nombre de voisins d'un sommet est au plus $n-1$ dans un graphe simple, cette complexité peut se réécrire comme $O(n^2)$.

Plus précisément, dans le graphe, le nombre d'arcs dépend du nombre de sommets dans λ noté n_k et du nombre de contraintes. Dans le graphe G, le nombre de sommets n est égal à $n_k + 1$ à cause du sommet fictif.

Le nombre d'arcs est donc égal à :

$$m = \left(7 * \frac{n_k}{2}\right) - 1$$

Si un cycle est détecté dans le graphe alors il n'existe pas de dates de début de service B_i finies telles que toutes les contraintes soient simultanément respectées. Donc λ est irréalisable. Si aucun cycle n'est détecté alors il existe une tournée associée à λ et il existe des labels B_i qui correspondent aux dates au plus tôt pour servir les clients tout en satisfaisant les contraintes.

Exemple :

Soit un DARP avec deux clients, un client devant aller de 1 à 3 avec un temps de trajet maximal de 5 et un client devant aller de 2 à 4 avec un temps de trajet maximal de 6. La durée totale de la tournée est limitée à 12. Les fenêtres de temps sont [1, 29] pour le sommet 1, [15, 30] pour le sommet 2, [26, 36] pour le sommet 3 et [36, 50] pour le sommet 4. Les temps de trajet pour le véhicule entre les sommets sont $C_{D_s1} = 1, C_{12} = 2, C_{23} = 3, C_{34} = 2, C_{4D_e} = 1$. Les temps de chargement et de déchargement sont considérés comme nuls dans cet exemple, illustré figure 2.14.

L'évaluation des dates au plus tôt a été réalisée ne détecte aucun cycle positif. Les labels finaux valent alors respectivement {25 ; 28 ; 30 ; 33 ; 36 ; 37} et correspondent aux dates au plus tôt.

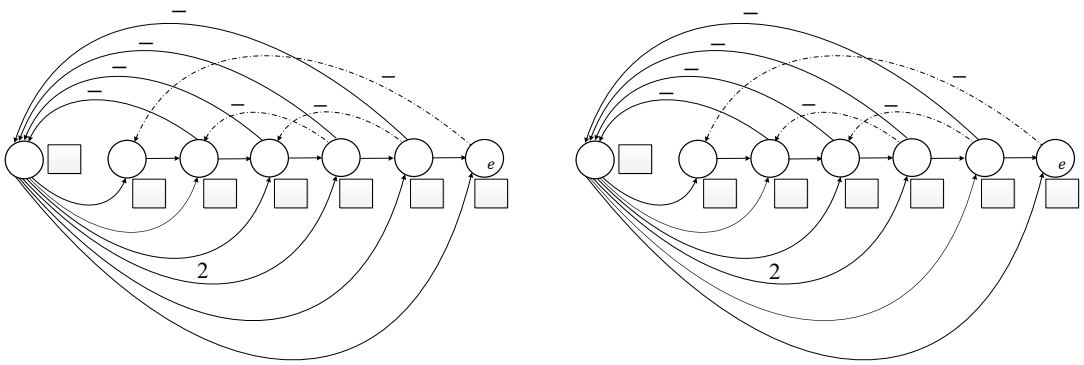


Figure 2.14: Exemple de graphe G avant et après le calcul des plus longs chemins en partant de 0

b. Calcul des dates au plus tard

Un raisonnement similaire peut être appliqué pour rechercher les dates au plus tard de début de service pour une tournée. En ordonnancement, calculer la date au plus tard d'une opération revient à faire coïncider la fin de l'opération courante avec le début de l'opération suivante qui commence le plus tôt. Dans G cela revient à trouver les labels B_i tels que :

$$B_i = \min_{j \in \delta_{s_i}^+} \left(B_j - cst(s_i s_j) \right) \quad \forall i \in \{1, \dots, n_k\}$$

où $\delta_{s_i}^+$ est l'ensemble des arcs entrants de s_i et $cst(s_i s_j)$ est la valeur de l'arcs de s_i à s_j.

Pour simplifier la formule précédente, un graphe G' est introduit. Il est construit de la manière suivante :

- à chaque sommet de G est associé un sommet dans G';
- à chaque arc orienté (i, j) de poids c_{ij} dans G est associé un arc (j, i) de poids $-c_{ij}$. dans G'

La figure 2.15 représente le graphe G de l'exemple précèdent et le graphe G' associé. Dans G' trouver les labels B_i au plus tard revient à appliquer la formule suivante :

$$B_i = \min_{j \in \delta^-_{s_i}} \left(B_j + cst(s_i s_j) \right) \ \forall i \ \in \{1, \dots, n_k\}$$

où $\delta^-_{s_i}$ est l'ensemble des arcs sortant de s_i. Comme pour le calcul des dates au plus tôt, le label du sommet 0 est initialement fixé à 0. En théorie des graphes, cette formule permet de calculer la valeur du plus court chemin. L'algorithme est alors un algorithme de plus court chemins de 0 vers les autres sommets de G'.

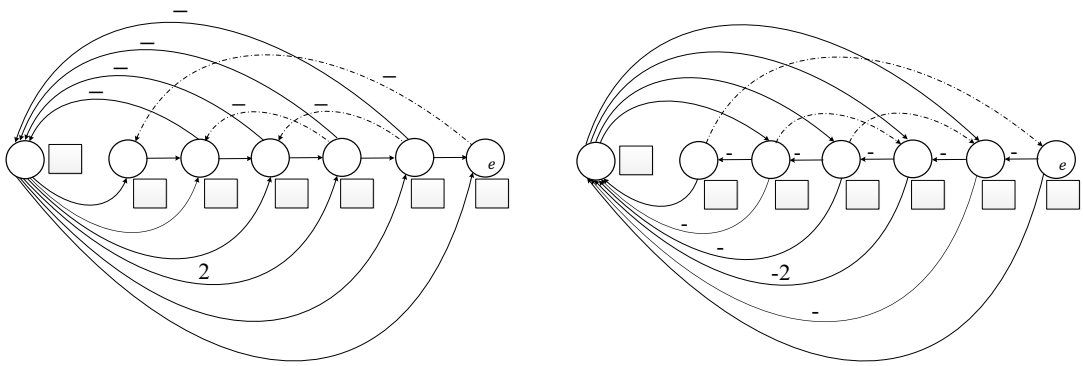


Figure 2.15: Exemple de graphes G (gauche) et G' (droite)

L'algorithme de Bellman-Ford calcule les plus courts chemins en $O(n_k^2)$ et permet d'obtenir les valeurs des labels associés aux sommets ou de détecter la présence de cycles de coût négatif. Si aucun cycle n'est détecté, ces labels représentent les dates de début de service au plus tard sur les différents sommets de la tournée. Si un cycle est détecté alors λ est irréalisable par le véhicule.

Notons que si un cycle est détecté pendant l'affectation des dates aux plus tôt (resp. aux plus tard) alors le même cycle est aussi détecté lors de l'affectation des dates aux plus tard (resp. au plus tard).

Exemple

Le calcul des dates d'arrivées au plus tard est réalisé sur l'exemple précédent, figure 2.16. La liste des dates au plus tard, *i.e.* les labels associés aux sommets (d_s ; 1 ; 2 ; 3 ; 4 ; d_e), est donc (27 ; 28 ; 30 ; 33 ; 36 ; 39) .

Ainsi, pour chaque B_i, on peut obtenir une valeur minimale et une valeur maximale. Toutes les solutions valides associées à λ ont donc une valeur de B_i comprise entre les deux bornes, c'est-à-dire pour l'exemple :

$B_{d_s} \in [25; 27]$; $B_1 \in [28; 28]$; $B_2 \in [30; 30]$; $B_3 \in [33; 33]$; $B_4 \in [36; 36]$; $B_{d_e} \in [37; 39]$.

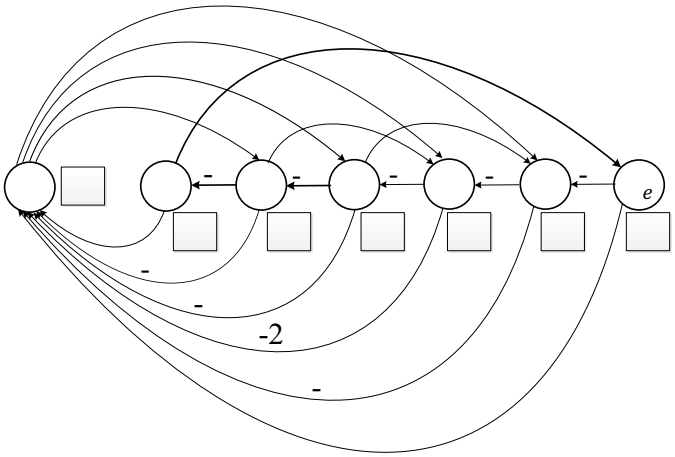


Figure 2.16: Calcul du plus court chemin en partant de 0 dans G'

Pour cet exemple simple, certaines dates minimales de début sont égales aux dates maximales de début. C'est le cas pour B_1, B_2, B_3, B_4.

Les dates de début des solutions retournées par les fonctions d'évaluation de (Cordeau and Laporte, 2003) et de (Firat and Woeginger, 2011) satisfont nécessairement ces intervalles. Ces fonctions sont présentées dans les parties suivantes.

2.3.4. Test de réalisabilité d'un λ par la méthode de (Cordeau and Laporte, 2003)

L'algorithme d'évaluation proposé par (Cordeau and Laporte, 2003) peut être défini à l'aide des graphes G et G' précédents, même s'il n'a pas été décrit ainsi dans l'article original. L'algorithme a la même complexité que l'algorithme de Bellman-Ford dans le pire cas. Il tire cependant parti de la structure du graphe, ce qui permet :

- une détection des cycles plus rapide que l'algorithme de Bellman-Ford ;
- en cas de cycle, le calcul de valeurs pour B_i qui minimisent la violation des contraintes ;
- sinon, le calcul de valeurs de B_i qui minimisent la durée totale de la tournée.

Après avoir construit le graphe G incluant uniquement les contraintes de fenêtres de temps minimales, l'algorithme procède en plusieurs grandes étapes:

- étape 1 : calculer les dates au plus tôt en prenant compte uniquement les fenêtres de temps minimales et les temps de trajet entre les sommets ;
- étape 2 : calculer les dates au plus tard qui terminent la tournée au plus tôt en prenant en compte les contraintes précédentes ainsi que les fenêtres de temps maximales ;
- étape 3 : vérifier la contrainte de durée totale puis calculer des dates au plus tôt en imposant la date de départ du dépôt au plus tard ;
- étape 4 : pour chaque client (origine,destination) dans λ, retarder au maximum les dates de début de service sur l'origine, puis mettre à jour les autres sommets et enfin vérifier la contrainte de temps de trajet associé au client.

Les étapes de l'évaluation sont reprises dans l'algorithme 3. Il prend en paramètre d'entrée-sortie un objet t qui contient plusieurs attributs. Certains de ces attributs doivent être initialisés, comme le nombre de sommets n_t ainsi que la liste des sommets s[]. Pour les autres,

les valeurs sont affectées. La fonction retourne un booléen de violation qui vaut faux si λ est valide et vrai sinon.

Un ensemble de variables sont nécessaires pour l'exécution de l'algorithme. Elles servent notamment à stocker le graphe, les valeurs au plus tard de B_i et un vecteur contenant la position des sommets de destination de chaque client (origine,destination) dans λ. Ces différentes étapes sont détaillées dans les parties suivantes.

Algorithme 3 is_feasible_tour (Cordeau and Laporte, 2003)

Input/Output parameters

t : a structure containing all the trip attributes: n_t, s[], B[], A[], D[]

Variable parameters

G : graph with $n_T + 1$ nodes
$B'[i]$: array of latest departure date of service on each node
$P'[i]$: array of integer : P[i] is the position of the associated delivery node if i is a pickup node

Output parameters

$violation$: boolean, $false$ if the trip is valid, $true$ otherwise

Begin

1 G := **build_Cordeau_graph(t)** //with only minimal time windows and distance constraints
2 **// STEP 1 : earliest starting time with only minimal time windows constraints**
3 **// STEP 2 : latest starting time which end the trip at the earliest date**
4 **// STEP 3 : check the total duration constraint**
5 **// STEP 4 : check the riding time for each client**
6 return $\{violation\}$

End

Etape 1 : affecter les dates au plus tôt avec uniquement les fenêtres de temps minimales

Cette étape consiste à fixer les dates de début de service au plus tôt en compte uniquement les contraintes de fenêtre de temps minimales ainsi que les temps de trajet. Ce graphe se compose uniquement des arcs représentant les contraintes (3a) et (5a) dans la figure 2.13. Ils sont tous orientés de gauche à droite et ont tous des valeurs positives. Il existe donc un ordre topologique qui permet de calculer le plus court chemin en $O(n_k)$, avec n_k le nombre de sommets dans λ. La formule suivante calcule les B_i en respectant l'ordre topologique :

$$\begin{cases} B[i] := B[0] + cst(0,i) & i = 1 \\ B[i] := max\big(B[0] + cst(0,i), B[i\text{-}1] + cst(i\text{-}1\,, i)\big) & \forall i \in \{2, \dots, n_k\} \end{cases}$$

où $cst(i,j)$ est le coût de l'arc (i,j). Une fois les valeurs $B[i]$ affectées, l'algorithme vérifie la violation des contraintes de fenêtre de temps maximale. Pour cela, on ajoute les arcs (3b) dans la figure 2.13 puis on évalue B_0avec la formule :

$$B[0] := \max_{j \in \delta_0^-}(B[0] + cst(j,0))$$

Deux cas se présentent : soit la valeur de B_0 n'est pas modifiée et l'algorithme peut continuer, soit la valeur de B_0 est modifiée et donc au moins une contrainte de fenêtre de temps maximales est violée. Dans ce second cas, deux approches sont possibles. On peut stopper en indiquant la non-réalisabilité de λ, soit noter la violation et poursuivre l'évaluation pour chercher si d'autres contraintes sont violées. Dans l'approche proposée par (Cordeau and Laporte, 2003), les auteurs utilisent un algorithme de recherche tabou (*tabu search*) dans lequel une partie de l'espace des tournées non valides est explorée. A ces λ non valides est affecté un coût prenant en compte les différents types de contraintes violées moyennant pondération. Pour pouvoir continuer l'évaluation de λ et détecter d'autres violations, la valeur

de la fenêtre de temps maximale du sommet associé à la violation est modifiée afin d'autoriser la valeur courante. A la fin de cette étape, la valeur obtenue sur le sommet final est la date au plus tôt E pour finir la tournée.

La figure 2.17 illustre cette étape avec au final $E = 37$.

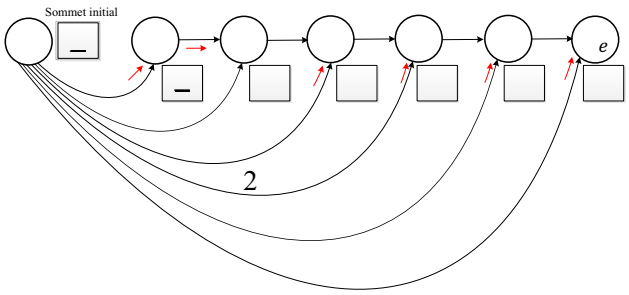


Figure 2.17 : Evaluation par (Cordeau and Laporte, 2003) étape 1

Dans l'algorithme 4, cette étape est comprise entre les lignes 2 et 10. Elles sont reprises dans la figure 2.18. Les labels sont stockés dans un vecteur de taille n_k. La ligne 1 (non présente) est une initialisation avec la construction du graphe. La ligne 2 fixe la valeur sur le sommet 0 ainsi que sur le sommet correspondant au dépôt initial.

La valeur des $B[i]$ est calculée dans les lignes 3 à 10. Comme l'ordre topologique garantit que le sommet courant n'est plus visité par la suite, la même boucle "for" est utilisée pour vérifier directement les contraintes de fenêtres de temps maximales, lignes 5 à 9. Les arcs nécessaires sont ajoutés ligne 8. Lorsqu'une violation est détectée (ligne 5), celle-ci est notée puis l'algorithme continue à vérifier les contraintes à la manière de (Cordeau and Laporte, 2003) (ligne 6) plutôt que de s'arrêter. A la fin, E est la valeur du label du dernier sommet.

```
2   t.B[0] := 0; t.B[1] := t.B[0] + cst(0, i)
3   for i := 2 to n_t do
4   |   t.B[i] := max(t.B[0] + cst(0, i), t.B[i 1] + cst(i 1, i))
5   |   if ( t.B[i] > l_{S_i} ) then
6   |   | violation: = true; call add_Arc(G, i, 0, −t.B[i] )
7   |   else
8   |   | call add_Arc(G, i, 0, −l_{S_i} )
9   |   end if
10  end for
```

Figure 2.18: Algorithme de (Cordeau and Laporte, 2003) étape 1

Proposition : si λ est valide alors il existe au moins une tournée telle que $B_{n_k} = E$.

Démonstration (par l'absurde):

supposons que la tournée valide qui finisse le plus tôt soit telle que $B_{n_k} = F > E$. Comme la tournée est valide et que F est la date au plus tôt sur n_k alors il existe un plus long chemin de valeur F dans G entre 0 et n_k. Il existe donc un plus long chemin tel que le nombre de sommets qui le composent soit minimal (n'incluant pas de cycle de coût nul).

Comme la tournée est valide, le plus court chemin ne peut pas emprunter d'arc correspondant à des fenêtres de temps maximales, sinon le sommet 0 aurait été modifié. De plus comme $F > E$ alors le plus long chemin entre 0 et n_k emprunte au moins un arc correspondant à une contrainte de temps de trajet maximal d'un client ou celui associé au temps total de la tournée sinon le résultat est celui obtenu à l'étape 1, c'est-à-dire E. Définissons donc (i, j) l'arc

associé à cette contrainte sur le temps de trajet du client (s'il y en a plusieurs, le dernier qui fait partie du chemin est considéré).

Comme λ est valide alors le sommet d'origine est situé avant la destination donc $i < j$. Comme l'arc considéré est le dernier du plus long chemin, alors le sous chemin reliant i à n_p se compose uniquement d'arcs directs $(i, i+1)$. Comme $i < j \leq n_p$ alors le chemin passe deux fois par le sommet j. Le plus court chemin entre 0 et n_p contient donc un cycle γ. Trois différents cas sont possibles dépendant du coût de γ. S'il est nul, alors les hypothèses sont contredites. S'il est négatif, alors il existe un plus long chemin entre 0 et n_p qui consiste à ne pas emprunter γ. S'il est positif, alors il n'existe pas de plus long chemin, ce qui contredit les hypothèses. Donc il n'existe pas de plus long chemin entre 0 et B_{n_k}, de valeur $F > E$. Donc la tournée valide qui finit le plus tôt est telle que $B_{n_k} = E$.

Cette démonstration est importante car l'algorithme proposé par (Cordeau and Laporte, 2003) s'intéresse uniquement aux tournées qui finissent à la date E.

□

Etape 2 : affecter les dates au plus tard qui finissent la tournée au plus tôt

L'étape 1 a permis d'obtenir E. L'étape 2 consiste à chercher la date I de départ au plus tard du dépôt initial d_s afin que la tournée finisse à la date E sur le $n_k^{ème}$ sommet d_e. Pour cela, la date sur le sommet final est fixée à E puis le plus court chemin est calculé depuis ce dernier sommet vers les autres dans G', construit comme expliqué dans la partie sur le calcul des dates au plus tard. Pour éviter la construction explicite de ce graphe auxiliaire, l'algorithme travaille sur le graphe G en utilisant la formule suivante :

$$\begin{cases} B'[i] := E & i = n_k \\ B'[i] := min\big(B'[0] - cst(i,O), B'[i+1] - cst(i, i+1)\big) \ \forall i \in \{1, \dots, n_k\text{-}1\} \end{cases}$$

où $cst(i,j)$ est le coût de l'arc (i,j). Dans cette formule, les arcs sont parcourus en sens inverse et on considère l'opposé de leur poids. $B'[i]$ correspond alors au minimum des valeurs obtenues en soustrayant au label d'un successeur, le coût de l'arc entre le sommet i et ce successeur.

G qui prend un compte uniquement les contraintes de fenêtre de temps et les temps de trajets entre les sommets. Une solution existe dans G, elle a été calculée à l'étape 1. Donc il existe aussi une solution valide avec les dates au plus tard. Pour que la solution soit valide, $B'[0]$ doit rester nul. La totalité des arcs de fenêtres de temps minimales ne sont donc pas utilisés. Donc, dans ce graphe, seuls les arcs de fenêtres de temps maximales et les temps de trajets entre les sommets sont considérés. Il existe donc un ordre topologique, cet ordre correspond à l'ordre inverse de traitement des sommets par le véhicule.

La valeur obtenue sur le sommet initial de λ est la date I de départ au plus tard pour finir la tournée au plus tôt. Dans la figure 2.17 qui représente l'étape 2 pour l'exemple, $I = 27$. Dans l'algorithme 4, ces étapes correspondent aux lignes 11 à 14, qui sont reprises dans la figure 2.19. La formule est appliquée à la ligne 13.

```
11  B'[n_T] := t.B[n_t]
12  for i := n_t − 1 to 1 do
13  |   B'[i] := min_{j∈δ_i^+}( B'[j] − cst(G, i, j))
14  end for
```

Figure 2.19: Algorithme de (Cordeau and Laporte, 2003) étape 2

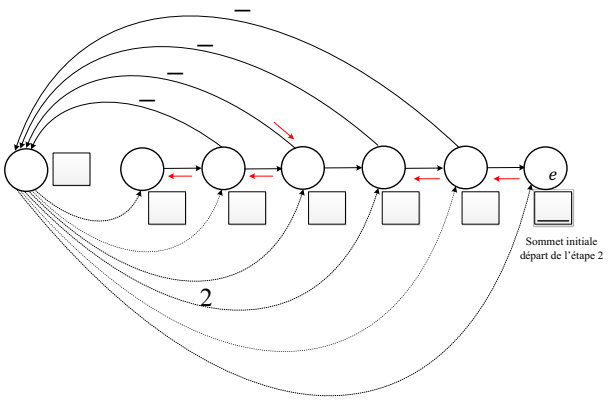


Figure 2.20: Evaluation par (Cordeau and Laporte, 2003) étape 2

Etape 3 : vérifier la contrainte sur la durée totale du trajet

La différence entre les dates I et E est le temps de trajet total minimal du véhicule. L'algorithme commence par vérifier que ce temps de trajet n'excède pas la contrainte imposée. Si ce n'est pas le cas, la solution n'est pas valide et l'algorithme peut soit s'arrêter, soit continuer à minimiser les violations à la manière de (Cordeau and Laporte, 2003). Dans ce second cas, la violation est prise en compte dans le coût puis le coût de l'arc est remplacé par $(E - I)$ afin de ne pas augmenter la violation dans la suite de l'algorithme.

Les valeurs sur les sommets sont ensuite mises à jour de manière à obtenir les labels au plus tôt associés à λ qui part du dépôt initial à la date I et qui finit à la date E. Les plus longs chemins sont à nouveau calculés en fixant à I la date de départ du dépôt initial, la formule et l'ordre topologique sont identiques à ceux de l'étape 1. La figure 2.21 montre les résultats obtenus sur l'exemple. La différence concerne le sommet 3 dont le label passe de 34 à 33.

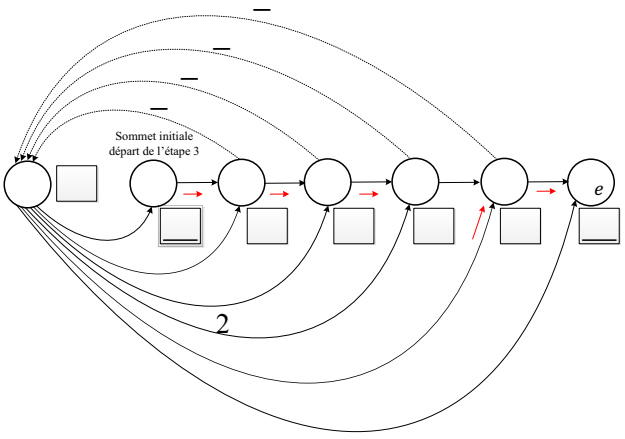


Figure 2.21: Evaluation par (Cordeau and Laporte, 2003) étape 3

La figure 2.22 illustre les différentes valeurs obtenues pour un autre exemple de tournée avec d'autres fenêtres de temps sur les nœuds. Le but est de mettre en évidence les différences qui existent entre les étapes. Les traits horizontaux de représentent le temps. Les rectangles représentent les fenêtres de temps et les carrés à l'intérieur sont les dates de début de service sur chacun des sommets de la tournée. Plus le carré est situé à droite et plus le début du service est tardif. Les flèches noires représentent le déroulement de l'algorithme.

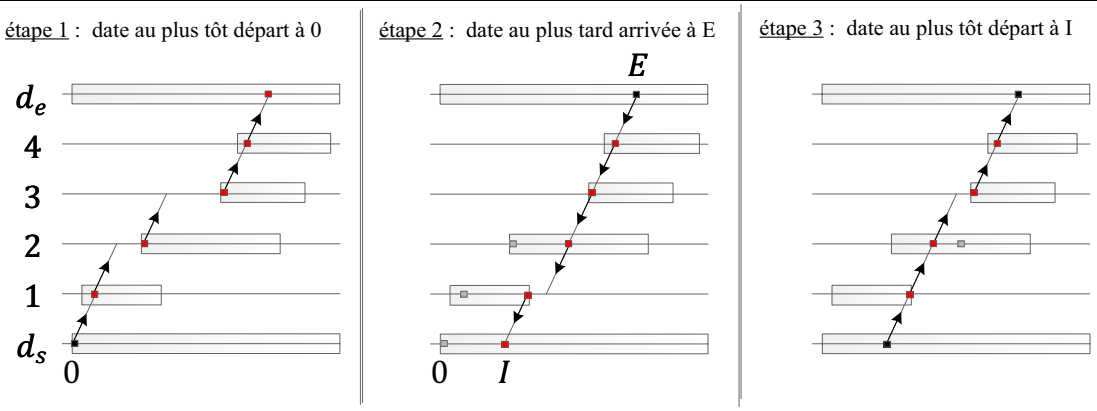


Figure 2.22: Evaluation par (Cordeau and Laporte, 2003) étape 1 à 3

Dans algorithme 4, repris dans la figure 2.23, ces étapes correspondent aux lignes 15 à 23. La vérification de la contrainte de temps de trajet intervient à la ligne 15. La ligne 18 ajoute l'arc dans le cas où la contrainte est valide et la ligne 16 ajoute l'arc dans le cas où la contrainte est violée. Enfin la mise à jour des valeurs est traitée de la ligne 21 à la ligne 23.

```
15  if (t.B[n_T] − B'[1] > T_t ) then
16  |   violation:= true; call add_Arc(G, n_t, 1, −(t.B[n_t] − B'[1]))
17  else
18  |   call add_Arc(G, n_t, 1, −T_t )
19  end if
20  t.B[1] := B'[1]
21  for i := 2 to card(G) − 1 do
22  |   t.B[i] := max_{j∈δ_i^-}( t.B[j] + cst(G, j, i))
23  end for
```

Figure 2.23: Algorithme de (Cordeau and Laporte, 2003) étape 3

Etape 4 : vérifier les contraintes sur les temps de trajet des clients

Cette dernière étape permet de traiter les contraintes de temps de trajet maximal pour chaque client. L'algorithme proposé par (Cordeau and Laporte, 2003) intègre ces contraintes séquentiellement en suivant l'ordre de passage du véhicule sur les sommets d'origines dans le vecteur λ.

Pour valider une contrainte sur le temps de trajet, la durée minimale de trajet d'un client est calculée. Pour cela le début du service sur le sommet d'origine associé au client est retardé au maximum sans modifier la date de fin de la tournée et sans que les contraintes précédentes, déjà valides, ne soient violées. Si la durée de ce trajet minimal respecte la contrainte alors la contrainte est ajoutée, puis les dates aux plus tôt pour tous les sommets situés après le sommet d'origine sont mises à jour.

C'est cette étape de l'algorithme qui induit une complexité en $O({n_k}^2)$. Pour obtenir la date de départ au plus tard sur chaque sommet d'origine, il faut utiliser la formule pour calculer la date au plus tard présentée dans la partie précédente :

$$B'[i] := \min_{j \in \delta_i^+}(B[j] - cst(i,j))$$

Cette formule est appliquée en partant du dernier sommet de λ et s'arrête lorsque le sommet d'origine est atteint. La valeur B_x sur le sommet x est alors fixée à la valeur $B'[x]$ puis les sommets suivants sont mis à jour avec la formule suivante :

$$B[i]:= \max_{j \in \delta_i^-}(B[j] + cst(j,i))$$

Si la contrainte de temps de trajet du client considéré est valide, alors l'arc est ajouté au graphe. Sinon la violation est prise en compte dans le coût associé à λ, puis un arc égal à la valeur du temps de trajet minimal obtenue est ajouté pour ne pas permettre l'augmentation de cette violation lors de l'intégration des contraintes suivantes. Puis on continue.

La figure 2.24 montre la modification de la tournée à l'étape 4 pour le client 2. Le client 2 a pour origine le sommet 2 et pour destination le sommet 4.

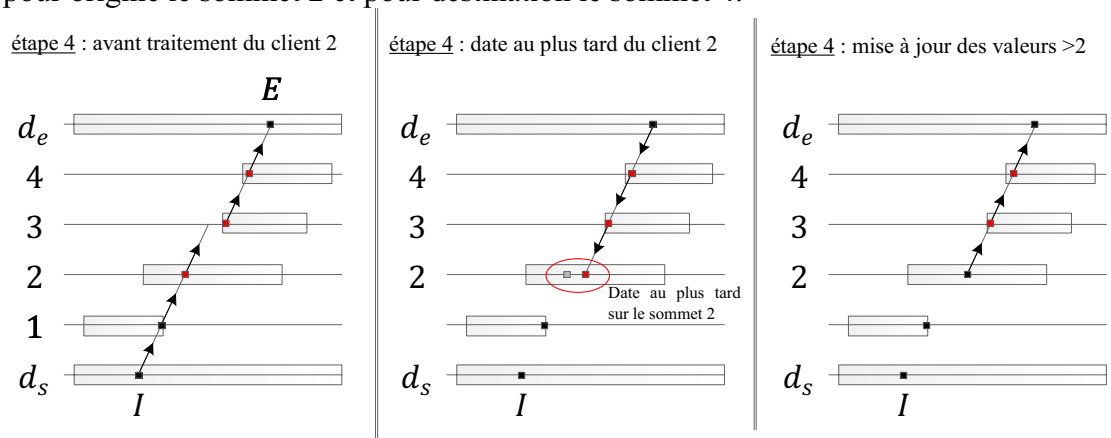


Figure 2.24: Evaluation par (Cordeau and Laporte, 2003) étape 4 pour le client 2

Dans algorithme 4, cette étape correspond aux lignes 24 à 39. Seuls les sommets d'origine sont considérés ligne 25. Pour ceux-ci, pour lequel la date au plus tard est calculée ligne 26 à 28. Puis la date de traitement de ce sommet est affectée à cette valeur au plus tard ligne 29. Les valeurs des sommets suivants sont mises à jours ligne 30 à 37, avec la vérification de la contrainte de temps de trajet associé au client ligne 32. Puis l'ajout ligne 35 de l'arc dans le cas où la contrainte est valide et ligne 33 l'ajout de l'arc le cas où la contrainte n'est pas respectée. Enfin la mise à jour des valeurs est traitée lignes 21 à 23. Cette étape est reprise dans la figure 2.25.

```
24 for i := 2 to n_t − 1 do
25 |   if (S_i is a pickup node)then
26 |   |   for j := n_t − 1 to i do
27 |   |   |   B'[i] := min_{j∈δ_i^+}( B'[j] − cst(G,i,j))
28 |   |   end for
29 |   |   t.B[i] := B'[i] //latest starting time
30 |   |   for j := i + 1 to n_t − 1 do
31 |   |   |   t.B(i) := max_{j∈δ_i^-}( t.B[j] + cst(G,j,i))
32 |   |   |   if ( t.B[P[i]] − t.B[i] > L_{S_i} ) then
33 |   |   |   |   violation: = true; call add_Arc(G, P[i], i, −(t. B[P[i]] − t. B[i] ) )
34 |   |   |   else // add the clients riding time
35 |   |   |   |   call add_Arc(G, P[i], i, −L_{S_i})
36 |   |   |   end if
37 |   |   end for
38 |   end if
39 end for
```

Figure 2.25: Algorithme de (Cordeau and Laporte, 2003) étape 4

L'algorithme 4, ci-dessous, représente l'algorithme générale qui teste la faisabilité d'une tournée proposée par (Cordeau and Laporte, 2003) et dont chaque partie a été présentée dans les paragraphes précédents.

Algorithme 4 is_feasible_tour (Cordeau and Laporte, 2003)

Input/Output parameters

t : a structure containing all the trip attributes: n_t, s[], B[], A[], D[]

Variable parameters

G : graph with $n_T + 1$ nodes
$B'[i]$: array of latest departure date of service on each node
$P'[i]$: array of integer : P[i] is the position of the associated delivery node if i is a pickup node

Output parameters

$violation$: boolean, $false$ if the trip is valid, $true$ otherwise

Begin

$G :=$ **build_Cordeau_graph(t)** //with only minimal time windows and distance constraints

$t.B[0] := 0$; $t.B[1] := t.B[0] + cst(0,i)$ **// STEP 1**
for $i := 2$ **to** n_t **do**
| $t.B[i] := max\big(t.B[0] + cst(0,i), t.B[i\text{-}1] + cst(i\text{-}1,i)\big)$
| **if** ($t.B[i] > l_{S_i}$) **then**
| | $violation :=$ true; **call** add_Arc($G, i, 0, -t.B[i]$)
| **else**
| | **call** add_Arc($G, i, 0, -l_{S_i}$)
| **end if**
end for

$B'[n_T] := t.B[n_t]$ **// STEP 2**
for $i := n_t - 1$ **to** 1 **do**
| $B'[i] := min_{j \in \delta_i^+}(B'[j] - cst(G,i,j))$
end for

if ($t.B[n_T] - B'[1] > T_t$) **then** **// STEP 3**
| $violation :=$ true; **call** add_Arc($G, n_t, 1, -(t.B[n_t] - B'[1])$)
else
| **call** add_Arc($G, n_t, 1, -T_t$)
end if
$t.B[1] := B'[1]$
for $i :=$ 2 **to** $card(G) - 1$ **do**
| $t.B[i] := \max_{j \in \delta_i^-}(t.\mathrm{B[j]} + cst(G,j,i))$
end for

for $i := 2$ **to** $n_t - 1$ **do** **// STEP 4**
| **if** (S_i is a pickup node)**then**
| | **for** $j := n_t - 1$ **to** i **do**
| | | $B'[i] := min_{j \in \delta_i^+}(B'[j] - cst(G,i,j))$
| | **end for**
| | $t.B[i] := B'[i]$ **//latest starting time**
| | **for** $j := i + 1$ **to** $n_t - 1$ **do**
| | | $t.B(i) := max_{j \in \delta_i^-}(t.B[j] + cst(G,j,i))$
| | | **if** ($t.B\big[P[i]\big] - t.B[i] > L_{S_i}$) **then**
| | | | $violation :=$ true; **call** add_Arc(G, P[i], i, $-$(t. B[P[i]] $-$ t. B[i]))
| | | **else // add the clients riding time**
| | | | **call** add_Arc($G, P[i], i, -L_{S_i}$)
| | | **end if**
| | **end for**
| **end if**
end for

return {$violation$}

End

2.3.5. Test de réalisabilité d'un vecteur λ proposé par (Firat and Woeginger, 2011)

L'algorithme proposé par (Firat and Woeginger, 2011) est à l'origine un simple test de réalisabilité d'un vecteur λ. Mais il peut être adapté pour retourner les valeurs associées aux B_i sur les différents sommets. Ces B_i peuvent être les valeurs au plus tôt, au plus tard ou même les valeurs retournées par (Cordeau and Laporte, 2003).

Contrairement aux autres algorithmes présentés, cet algorithme a une complexité linéaire en $O(n_k)$. La différence de complexité provient de la capacité de l'algorithme à lier entre elles certaines valeurs de début de service.

Ainsi, la figure 2.26 représente deux tournées dans lesquelles ont été calculées les dates au plus tôt. Entre la partie gauche et la partie droite, seule la fenêtre de temps minimale imposée pour le début de service sur d_s est passée de 1 à 6. Cette modification entraine un retard des dates au plus tôt sur l'ensemble des sommets de la tournée. Pour les algorithmes classiques, la mise à jour du graphe de la gauche vers la droite demande n_k opérations. Or, sur tous les sommets, la modification consiste à ajouter 5 unités de temps qui représente l'attente supplémentaire introduite sur le premier sommet et qui s'est propagée.

L'idée de l'algorithme proposé par (Firat and Woeginger, 2011) consiste, sous certaines conditions, à lier dans un même groupe les sommets du graphe puis à associer des valeurs aux groupes et non plus aux sommets comme précédemment. Ainsi la mise à jour des sommets d'un même groupe est effectuée en $O(1)$.

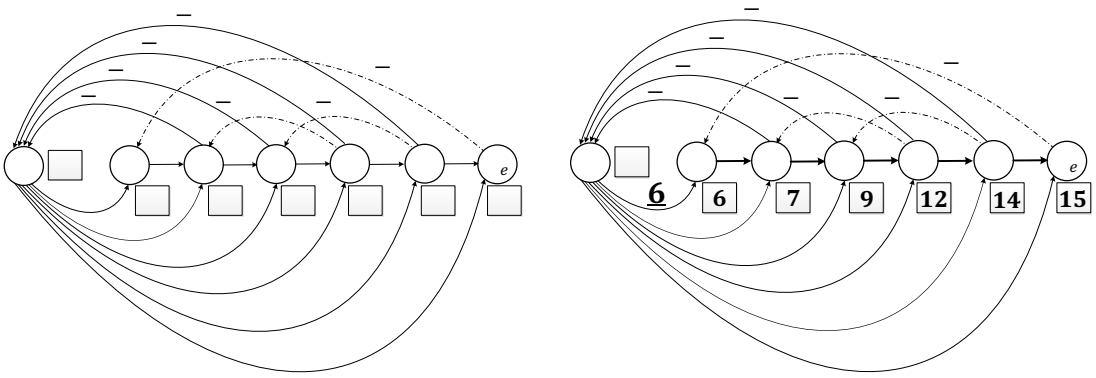


Figure 2.26: Mise en évidence des liens entre les sommets

L'algorithme procède en plusieurs étapes :

- étape 1 : changer de variable, afin de pouvoir rassembler les sommets dans un même groupe. L'algorithme ne calcule plus les dates de début mais s'intéresse au temps d'attente du véhicule ;
- étape 2 : construire le graphe associé ;
- étape 3 : calculer les dates de début à des ensembles ;
- étape 4 : évaluer le graphe ;
- étape 5 : revenir à la solution.

Comme pour l'évaluation de (Cordeau and Laporte, 2003), l'algorithme 5, prend en entrée-sortie un objet t construit à l'aide d'un vecteur λ. Il est composé de plusieurs attributs, certains initialisées à l'aide de λ comme le nombre de sommets n_t ainsi que la liste des sommets s[]. Les autres attributs sont affectés si λ est valide. La fonction retourne un booléen qui est égal à vrai dans ce cas et faux sinon.

Un ensemble de variables est nécessaires pour l'exécution de l'algorithme : une pile d'entiers, un graphe ainsi que des éléments qui sont décrits dans la suite de ce paragraphe. Ces différentes étapes sont détaillées dans les parties suivantes.

Algorithme 5 is_feasible_tour (Firat and Woeginger, 2011)

Input/Output parameters

t : a structure containing all the trip attributes: n_t, s[], B[], A[], D[]

Variable parameters

$stack$: stack of integers
G_2 : graph with $n_T + 1$ nodes
$E[]$: vector for sets representation
$X[]$: value associated with each set

Output parameters

$violation$: boolean, $false$ if the trip is valid, $true$ otherwise

Begin

1 **STEP 1 : compute new variables**
2 **STEP 2 : build the graph**
3 **STEP 3 : initialize the sets and variables**
4 **STEP 4 : compute the lowest X_i on nodes**
5 **STEP 5 : check violation and compute B_i**
6 return $\{violation\}$

End

Etape 1 : changer de variable

Pour comprendre cette étape il faut noter que, la durée entre le départ du dépôt et le début du service sur un sommet se compose de deux durées distinctes : une partie du temps durant laquelle le véhicule se déplace et l'autre partie où il attend. Le temps de déplacement correspond à la somme des temps de trajet entre les sommets de la tournée du dépôt jusqu'au sommet considéré ; le temps d'attente correspond à la somme des attentes sur le sommet considéré ainsi que ceux qui le précèdent dans λ. Donc, pour un λ donné, les tournées peuvent avoir des dates de début de service différentes pour un même sommet en fonction de la politique de parcours. En revanche, le temps de déplacement du véhicule est invariable. C'est donc le temps d'attente qui définit la politique de parcours.

Le temps de service, qui correspond au temps pour prendre en charge les clients dans le véhicule, est constant sur chaque sommet. Il peut donc être inclus dans le temps de déplacement sans perte de généralité.

La première idée de l'algorithme de (Firat and Woeginger, 2011) est de supprimer le temps nécessaire pour se déplacer de sommet en sommet. Pour cela, deux nouvelles variables sont d'introduites pour chaque sommet i :

- X_i : le temps d'attente du véhicule entre le départ du dépôt et le début du service B_i, cette variable vaut :

$$X_i = \sum_{k=1}^{i} w_k$$

- Γ_i : la somme des temps de trajet et de service entre le départ du dépôt et le sommet i.

$$\Gamma_i = \begin{cases} 0 \text{ si } i = 1 \\ \sum_{k=1}^{i-1} c_{S_k S_{k+1}} + d_k \text{ sinon} \end{cases}$$

Il est important de noter que Γ_i peut être calculée pour tous les sommets en $0(n_k)$. Dans algorithme 8, l'évaluation cette étape correspond aux lignes allant de 1 à 4, qui sont reprises dans la figure 2.27.

```
1  Π[1] ≔ 0
2  for i ≔ 2 to t do
3  |    Π[i] ≔ Π[i − 1] + c_{t.s[i−1],t.s[i]}
4  end for
```

Figure 2.27: Algorithme de (Firat and Woeginger, 2011) étape 1

En utilisant les contraintes (4), (5) et (8), on peut montrer que le début du service sur tous les sommets de la tournée est égal à la somme des attentes et aux temps de déplacement du véhicule :

$$B_i = X_i + \Gamma_i, \forall i = \{1, \dots, n_k\} \quad (11)$$

Démonstration :

Pour $i = 1$ c'est trivial,

Pour $i = 2$ alors $B_2 = W_1 + d_0 + c_{0S_2} + W_2 = \sum_{k=1}^{2} w_k + \sum_{k=1}^{1}(c_{S_k S_{k+1}} + d_k) = X_2 + \Gamma_2$

Pour $i \geq 2$ en utilisant les contraintes suivantes:

$$D_i = B_i + d_p\ , \forall i = \{2, \dots, n_k\} \text{ et } p = s_i \quad (4)$$

$$A_{i+1} = D_i + c_{p,q}\ , \forall i = \{1, \dots, n_k - 1\} \text{ et } p = s_i \text{ et } q = s_{i+1} \quad (5)$$

$$W_i = B_i - A_i\ , \forall i = \{1, \dots, n_k\} \quad (8)$$

B_{i+1} peut être exprimé en fonction de B_i :

$A_{i+1} = D_i + c_{p,q}$, $\forall i = \{2, \dots, n_k - 1\}$ et $p = s_i$ et $q = s_{i+1}$ d'après (5)

$A_{i+1} = B_i + d_p + c_{p,q}$, $\forall i = \{2, \dots, n_k - 1\}$ et $p = s_i$ d'après (4)

$B_{i+1} - W_{i+1} = B_i + d_p + c_{p,q}$, $\forall i = \{2, \dots, n_k - 1\}$ d'après (8)

Par récurrence, supposons que la propriété soit vraie au rang i , *i.e.* $B_i = X_i + \Gamma_i, \forall i = \{2, \dots, n_k\}$ et montrons qu'elle est vraie au rang $i + 1$:

$B_{i+1} - W_{i+1} = B_i + d_p + c_{p,q}$, $\forall i = \{2, \dots, n_k - 1\}$ d'après résultat précédant

$B_{i+1} - W_{i+1} = X_i + \Gamma_i + d_p + c_{p,q}$, $\forall i = \{2, \dots, n_k - 1\}$ d'après l'hypothèse

$B_{i+1} = (X_i + W_{i+1}) + (\Gamma_i + d_p + c_{p,q})$, $\forall i = \{2, \dots, n_k - 1\}$

$B_{i+1} = X_{i+1} + \Gamma_{i+1}$, $\forall i = \{2, \dots, n_k - 1\}$

Donc l'hypothèse de récurrence est vraie pour tout $k > 2$ donc (11) est vraie

□

Les contraintes du problème peuvent donc être exprimées en fonction de X_i et Γ_i.

Etape 2 : construire le graphe

L'étape 2 consiste à construire le graphe G_2 correspondant à λ pour appliquer l'évaluation. Par rapport au graphe G précèdent, seules les valeurs des arcs changent. Dans algorithme 8, la construction n'est pas détaillée mais correspond à la ligne 5, reprise dans la figure 2.28.

5	$\mathbf{G_2} \coloneqq$ **build_Firat_graph(T[], Π[])**

Figure 2.28: Algorithme de (Firat and Woeginger, 2011) étape 2

Pour obtenir les nouvelles valeurs des arcs l'égalité (11) est utilisée dans les contraintes (3a), (3b), (5b), (6a), (7a). Un ensemble de 5 nouvelles contraintes est obtenu avec une variable correspondant au label du sommet fictif, initialisé à 0 et notée $X_0 = 0$:

$X_0 + e_{p,} \leq X_i + \Gamma_i \; \forall i = \{1, \dots, n_k\}$ et $p = s_i$ d'après (3a)

$\rightarrow \boxed{X_0 + (e_p - \Gamma_i) \leq X_i} \; \forall i = \{1, \dots, n_k\}$ (3'a)

$X_i + \Gamma_i - l_p \leq X_0, \forall i = \{1, \dots, n_k\}$ et $p = s_i$ d'après (3b)

$\rightarrow \boxed{X_i + (\Gamma_i - l_p) \leq X_0} \; \forall i = \{1, \dots, n_k\}$ (3'b)

$X_i + \Gamma_i + (d_p + c_{p,q}) \leq X_{i+1} + \Gamma_{i+1}, \forall i = \{1, \dots, n_k - 1\}$ d'après (5b)

$\rightarrow X_i + (d_p + c_{p,q} + (\Gamma_i - \Gamma_{i+1})) \leq X_{i+1}, \forall i = \{1, \dots, n_k - 1\}$

$\rightarrow \boxed{X_i \leq X_{i+1}}, \forall i = \{1, \dots, n_k - 1\}$ (5'b)

$X_j + \Gamma_j - (d_p + L) \leq X_i + \Gamma_i$, $p = s_i$ d'après (6a)

$\rightarrow \boxed{X_j - (d_p + L - (\Gamma_j - \Gamma_i)) \leq X_i}$ (6'a)

$X_{n_k} + \Gamma_{n_k} - T_k \leq X_1 + \Gamma_1$ d'après (7a)

$\rightarrow \boxed{X_{n_k} - (T_k - \Gamma_{n_k}) \leq X_1}$ (7'a)

La figure 2.29 donne une représentation du graphe G_2 correspondant à un problème d'évaluation d'un vecteur λ de n_k éléments pour l'algorithme de (Firat and Woeginger, 2011).

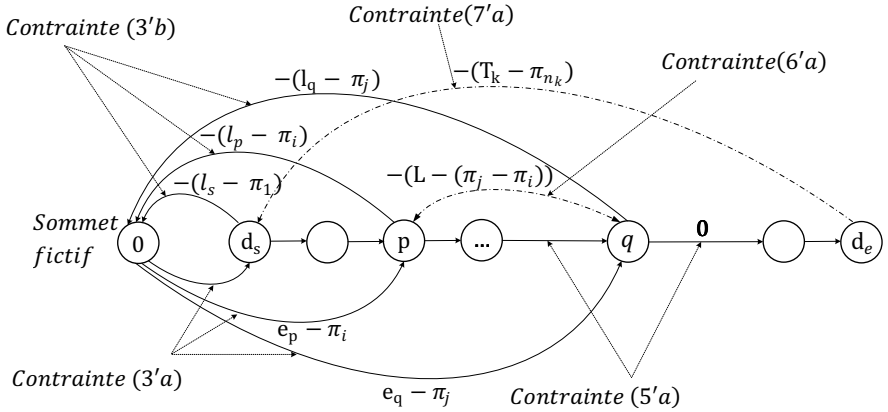


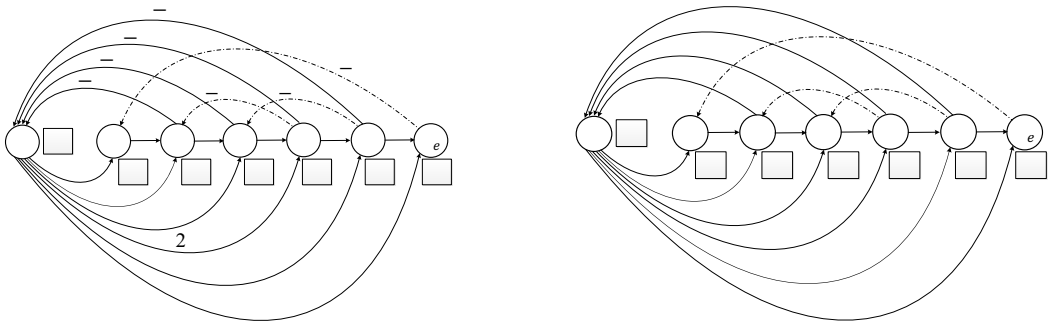
Figure 2.29: Graphe G_2 pour l'algorithme de (Firat and Woeginger, 2011)

Contrairement au graphe G, les seuls arcs strictement positifs sont ceux modélisant les dates minimales pour débuter une opération. Ces dates viennent des contraintes sur les fenêtres de

temps associées à chaque opération. Si un autre arc a un poids positif alors λ est trivialement non valide car il existe un cycle positif composé de cet arc et des arcs issus de la contrainte $(5'a)$ dont les coûts sont nuls. Lors de la construction du graphe, une étape consiste donc à vérifier la valeur des arcs. Si un arc ne respecte pas cette propriété alors le vecteur λ peut être marqué comme non valide ; pour obtenir des dates de début qui minimisent la violation comme pour l'algorithme de (Cordeau and Laporte, 2003) la valeur de l'arc de coût positif est fixée à 0.

Exemple :

La figure 2.30 (a) reprend le graphe G de la figure 2.14 et la figure 2.30 (b) représente le graphe G_2 utilisé pour l'évaluation de (Firat and Woeginger, 2011). Les arcs autres que ceux représentant les fenêtres de temps minimales sont tous négatifs ou nuls. Donc, pour cet exemple il est possible de continuer l'évaluation.



(a) Graphe G, *(b) Graphe G_2*

Figure 2.30: Exemple de graphe G et du graphe G_2 associé au même problème

Comme les seuls arcs strictement positifs de G_2 sont issus du sommet fictif 0, si un cycle de coût positif existe alors il comporte le sommet 0. Le cycle est donc constitué au minimum d'un arc entrant sur 0 qui correspond à une contrainte de fenêtre de temps minimale et un arc sortant de 0 qui correspond à une contrainte de fenêtre de temps maximale.

Etape 3 : associer les dates de début

Le graphe G_2 obtenu à l'étape précédente peut être évalué de la même manière que le graphe G, avec l'algorithme de plus court chemin ou encore avec l'algorithme de (Cordeau and Laporte, 2003) et la complexité reste en $O({n_k}^2)$. Pour évaluer ce graphe avec une complexité linéaire, une nouvelle structure de type *union-find* est introduite. Elle propose des opérations efficaces de recherche et de fusion de sous-ensembles. Un ensemble de sommets est, comme son nom l'indique, le regroupement de plusieurs sommets de la tournée. Tous les sommets qui appartiennent au même ensemble auront la même valeur X_i.

Initialement, dans l'algorithme, à chaque sommet s_i de λ est associé un ensemble qui se compose uniquement du sommet s_i. C'est pourquoi les ensembles sont notés par la suite E_i, avec i l'ensemble dans lequel le $i^{ème}$ sommet s_i de λ est initialement affecté. Par exemple, pour une tournée avec 2 clients (comme dans les exemples précédents), les ensembles initiaux sont :

$$E_1 = \{d_s\}; E_2 = \{1\}; E_3 = \{2\}; E_4 = \{3\}; E_5 = \{4\}; E_6 = \{d_e\}$$

Puis, au cours de l'algorithme, certains ensembles sont fusionnés. Au final, les ensembles sont constitués de :

$E_1 = \{d_s, 1,2\}; E_4 = \{3\}; E_5 = \{4, d_e\}$

Pour manipuler ces ensembles, deux opérations sont nécessaires : $find$ qui permet de trouver à quel ensemble appartient un sommet et $union$ qui permet de faire l'union de deux ensembles. Avant de présenter ces deux fonctions, il faut introduire la représentation qui est utilisée pour manipuler ces ensembles.

Les ensembles sont stockés dans un vecteur de taille n_k, comme le montre la figure 2.31. Chaque case de ce vecteur noté E correspond à un sommet de λ. La valeur de chaque case correspond soit à l'indice d'un autre sommet inclus dans l'ensemble soit à l'indice de l'ensemble lui-même. L'indice de l'ensemble est atteint lorsque la valeur de la case est égale à l'indice du vecteur.

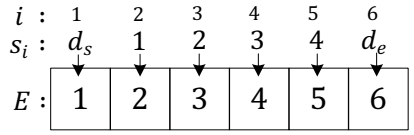


Figure 2.31: Représentation des ensembles à l'état initial (un sommet par ensemble)

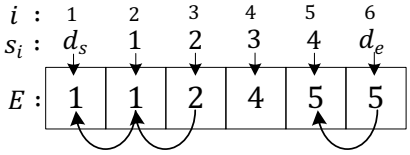


Figure 2.32: Représentation des ensembles $E_1 = \{d_s, 1,2\};\ E_4 = \{3\}; E_5 = \{4, d_e\}$

L'algorithme $find$ appliqué sur le vecteur E permet de retrouver l'indice de l'ensemble auquel appartient un sommet de λ. Il est défini dans algorithme 6. Il prend en argument le vecteur E des ensembles et la position du sommet dans λ. Cette fonction est décrite sous une forme récursive : si elle n'est pas sur l'indice de l'ensemble, la fonction recherche ligne 2 à quel ensemble appartient le nouveau nœud. On répète l'opération jusqu'à obtenir l'indice de l'ensemble, c'est-à-dire lorsque $E[i] = i$.

Algorithme 6 find

Input/Output parameter
$E[]$: vector of sets

Input parameter
i : position of the node in the trip

Output parameters
$E[i]$: indices of sets which contain node s_i

```
Begin
1   if (E[i] != i ) then
2       E [i] := call find (E, i)
3   endif
4   return E[i]
End
```

Dans cette version de l'algorithme, la ligne 2 permet, une fois le résultat obtenu, de mettre à jour les valeurs dans E afin de simplifier les appels suivantes à $find$. La figure 2.33 donne un exemple du vecteur E obtenu après avoir recherché l'ensemble associé au $3^{ème}$ sommet de λ dans le vecteur E de la figure 2.32, Notons que plusieurs vecteurs E différents peuvent donc correspondre au même ensemble.

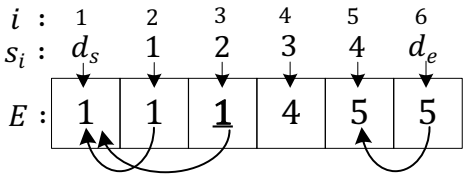


Figure 2.33: Vecteur E après la recherche de l'ensemble associé à i=2

La deuxième opération est l'union de deux ensembles. $Union$ prend en arguments E ainsi que deux indices correspondant à deux sommets différents de λ. Le fonctionnement de $union$ est défini dans l'algorithme 7. Sur les lignes 1 et 2, on recherche les ensembles correspondant à ces deux sommets. Pour les fusionner, on modifie ensuite une unique valeur dans E. Cette valeur est celle qui correspond à l'indice de l'ensemble dans lequel est inclus le j^{eme}sommet.

Les algorithmes de $union/find$ utilisent le plus souvent le nombre d'éléments contenus dans chacun des ensembles pour choisir quel intervalle doit être inclus dans l'autre. Ces algorithmes préférent supprimer l'ensemble le plus petit pour faciliter les futures opérations $find$. Pour l'algorithme de (Firat and Woeginger, 2011) c'est uniquement l'ensemble avec le plus petit indice qui est conservé. Les raisons de ce choix sont abordées dans l'étape 4.

Algorithme 7 union

Procedure name: evaluation

Input/output parameters

$E[]$: List of sets

Input parameters

i : position of a node in the first set
j : position of a node in the second set

Begin

1 i := **call** find (E, i)
2 j := **call** find (E, j)
3 $E[j] := i$

End

La figure 2.34, représente E après la fusion de l'ensemble associé au sommet 4 (sommet en $5^{ème}$ position) et 1 (sommet en $2^{ème}$position). On constate que le $6^{ème}$ sommet est dans le même ensemble que le $5^{ème}$ et qu'il n'a pas été mis à jour. Les ensembles résultant de cette union sont : $E_1 = \{d_s, 1,2,4, d_e\}$; $E_4 = \{3\}$;

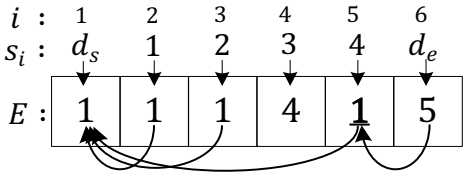


Figure 2.34: Vecteur E après l'Union de pour i=2 et j = 5

Dans l'étape suivante, à chaque ensemble est associée une valeur qui représente l'attente minimale que doit respecter le véhicule avant de traiter les sommets contenus dans l'ensemble. Ainsi dans la suite du chapitre, un tableau de deux lignes est utilisé. Chaque colonne représente un sommet de λ, la première ligne représente les ensembles, comme expliqué précédemment, et la seconde ligne correspond aux valeurs X associées aux ensembles.

Chaque ensemble à une unique valeur qui est stockée dans la colonne correspondant au sommet initial de l'ensemble : un exemple est donné dans la figure 2.35.

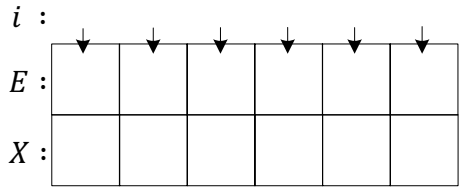


Figure 2.35: Vecteur E associé au vecteur X

Dans l'algorithme 8, l'initialisation des ensembles est effectuée de la ligne 6 à la ligne 8, qui sont reprises dans la figure 2.36.

```
6  for i := 1 to t do
7  |   E[i] := i; X[i]: = 0 ;
8  end for
```

Figure 2.36: Algorithme de (Firat and Woeginger, 2011) étape 3

Comme expliqué précédemment, les ensembles sont stockés dans $E[]$ et leurs valeurs sont stockées dans $X[]$. Ils sont initialisés avec un sommet par ensemble et la valeur associée à chacun est égale 0. La figure 2.37, représente les différents éléments après l'initialisation pour l'exemple du DARP composée uniquement de 2 clients.

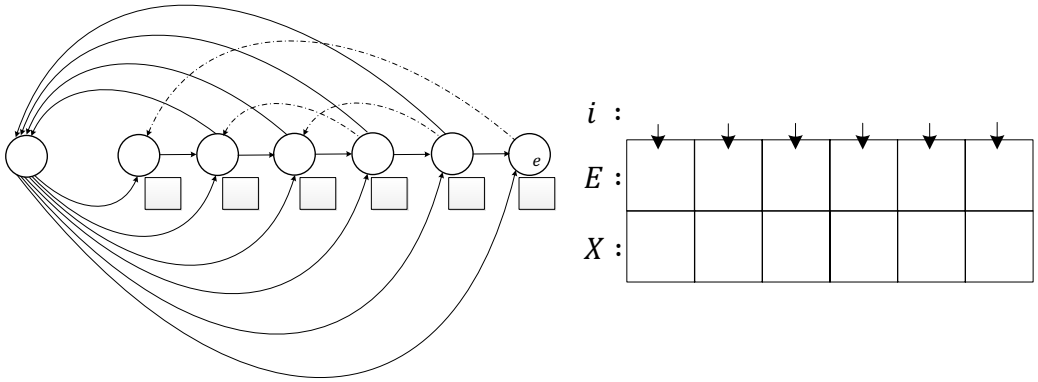


Figure 2.37: Valeur des vecteurs E et X initiaux

Les fonctions $find$ et $union$ sont utilisées à plusieurs reprises dans les étapes 4 et 5.

Etape 4 : évaluer le graphe

Pour l'algorithme 8, il a été fait le choix d'expliquer le fonctionnement de la fonction proposée par (Firat and Woeginger, 2011) pour le calcul des dates au plus tôt. Cette étape se déroule de la ligne 11 à la ligne 23,. Elle arrive après l'initialisation des ensembles et de leurs valeurs associées et commence par une initialisation, lignes 9 et 10. L'élément $pile$ initialisé (ligne 10) est une structure classique qui permet à l'algorithme de stocker les différents indices qui correspondent aux ensembles. Elle est initialement vide.

L'évaluation consiste à calculer le plus long chemin entre le sommet 0 et tous les autres sommets du graphe. Comme pour l'algorithme proposé par (Cordeau and Laporte, 2003), le graphe est évalué plusieurs fois en prenant en compte des arcs différents. Dans l'étape 4, les arcs correspondants aux fenêtres de temps maximales sont omis. Le graphe résultant est

illustré figure 2.38. Sans ces arcs, le graphe ne peut pas contenir de cycles de un coût positif car ce sont les seuls qui entrent sur le sommet 0, comme cela a été expliqué dans la partie précédente.

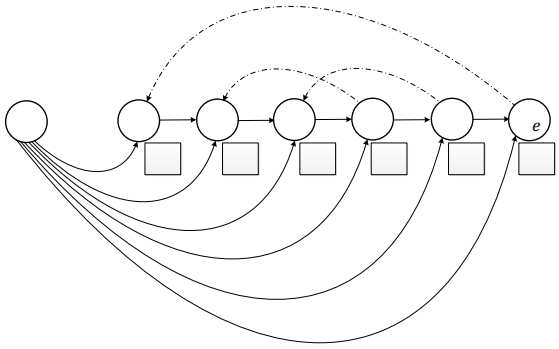


Figure 2.38: Exemple de graphe à évaluer

L'algorithme calcule les plus longs chemins entre le dépôt (0) et les sommet de λ. Lorsqu'un plus long chemin passe par un des arcs horizontaux de coût nul, alors les sommets aux extrémités de l'arc ont la même valeur et ils sont liés. Les sommets sont alors associés au même ensemble.

Pour obtenir une complexité linéaire, l'ordre de traitement des sommets est important. Dans G_2, si les arcs horizontaux de coût nul sont supprimés, alors il existe un ordre topologique qui consiste à traiter le sommet fictif 0 puis à parcourir les sommets dans l'ordre inverse de λ. Tous les arcs entre les sommets étant orientés dans le sens inverse, le traitement les sommets se fait donc en suivant l'ordre topologique, comme illustré sur la figure 2.39. Sur chaque sommet, deux étapes distinctes sont réalisées. La première consiste à obtenir la valeur du plus court chemin entre 0 et le sommet courant sans prendre en compte les arcs de valeur 0. La deuxième propage la valeur trouvée vers les sommets déjà traités.

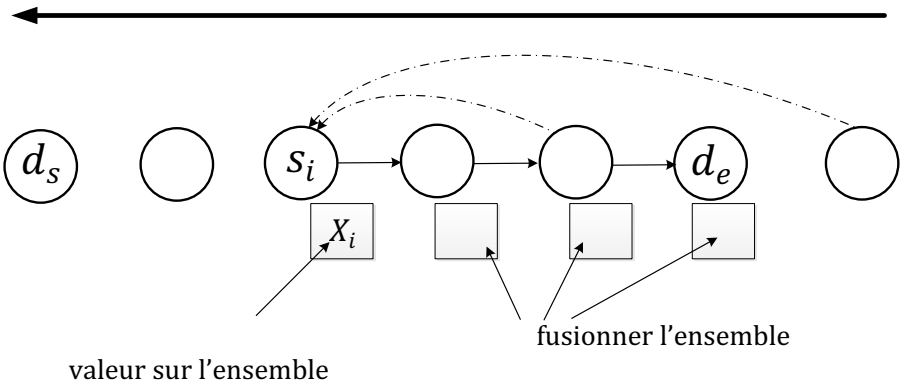


Figure 2.39: Principe de l'évaluation

Sur chaque sommet, la première étape consiste à calculer la valeur du plus long chemin sans regarder les arcs de coût nul. La valeur est obtenue à l'aide de la formule suivante, appelée ligne 12 de l'algorithme 8:

$$X[i] := \max_{j \in \delta_i^-} \big(X[find(E,j)] + cst(G_2,i,j)\big)$$

Le nombre d'arcs entrants sur le $i^{ème}$ sommet δ_i^- dépend du type de nœud et il est au maximum de 2 dans le cas des sommets associés à l'origine d'un client.

La différence par rapport à la formule de plus long chemin classique est l'utilisation de la fonction $find$ pour obtenir la valeur correspondant à l'ensemble associé au sommet d'origine de l'arc.

La seconde partie consiste à fusionner les ensembles qui ont déjà été traités avec l'ensemble courant si la valeur X_i obtenue est plus élevée que la valeur calculée sur les ensembles précédents. Par construction, ces ensembles sont triés par ordre croissant sur les valeurs de X_i, donc le parcourt se fait du sommet courant vers la fin de la tournée comme illustré sur la figure 2.40. Si la valeur rencontrée sur l'un des ensembles précédents est supérieure à la valeur de l'ensemble courant alors la deuxième étape est terminée et l'algorithme passe au sommet suivant.

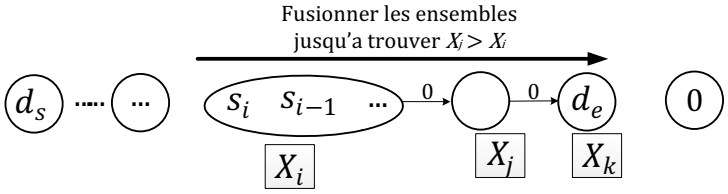


Figure 2.40: Principe de l'évaluation à l'étape 2

Pour obtenir la complexité linéaire recherchée, la recherche des ensembles ne peut pas être effectuée à l'aide de la fonction *find* appliquée à tous les sommets déjà traités. C'est pourquoi une pile d'entiers, construite à chaque étape de l'algorithme, est utilisée pour stocker, par ordre croissant des X_i, les indices des ensembles contenants des sommets.

Le fonctionnement est le suivant : on suppose que la pile contienne les indices des ensembles précédents non vides triés par ordre croissant des X_i. Les ensembles sont alors dépilés et fusionnés jusqu'à obtenir un ensemble dont la valeur est supérieure à la valeur X_i de l'ensemble courant. A la fin, si aucun ensemble de valeur supérieure à X_i n'était présent, alors la pile est vide et l'ensemble courant est ajouté. Si un tel ensemble est obtenu, alors l'algorithme empile son indice avant d'empiler celui correspondant au nouvel ensemble. Dans les deux cas, si la pile été triée initialement, alors elle reste triée par ordre croissant des valeurs X_i et seul les indices des ensembles non vides sont présents.

Dans l'algorithme 8 cette étape correspond aux lignes 13 à 22, reprises dans la figure 2.48.

```
9   violation := false
10  stack := null
11  for i := n_t to 1 do
12  |   X[i] := max_{j∈δ_i^-} ( X[find(E, j)] +cst(G_2, i, j))
13  |   do
14  |   |   {j} :=all pop(stack)
15  |   |   if (X[i] ≥ X[j]) then
16  |   |   | union(E,i,j)
17  |   |   else
18  |   |   | push(stack, j)
19  |   |   | stop:= true
20  |   |   end if
21  |   while (stack ≠null)&& (stop = false)
22  |   call push(stack,i )
23  end for
```

Figure 2.41: Algorithme de (Firat and Woeginger, 2011) étape 4

De la figure 2.42 à la figure 2.47, le déroulement de l'algorithme est illustré sur l'exemple de λ utilisé précédemment. Les arcs représentant les fenêtres de temps maximales ont été supprimés pour une meilleure visibilité.

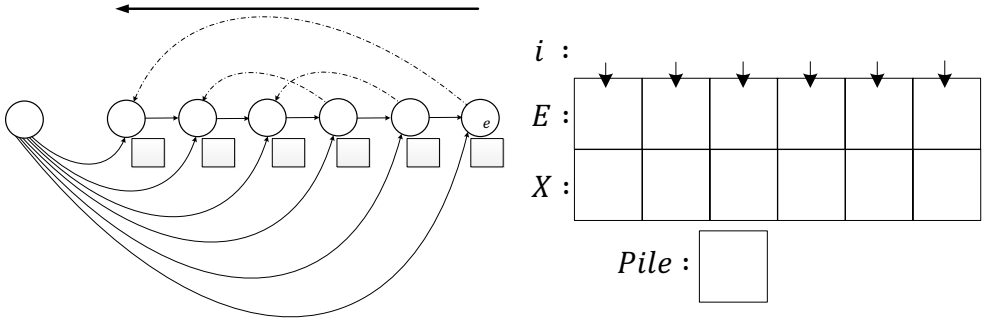


Figure 2.42: Valeurs initiales

La figure 2.42 représente les différents éléments après la phase d'initialisation.

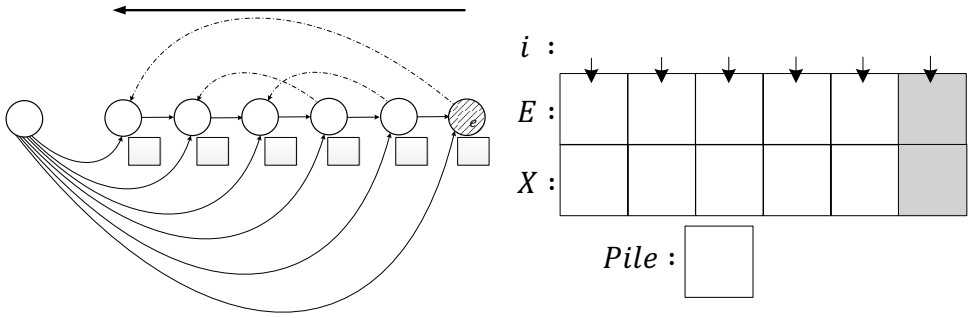


Figure 2.43: Algorithme après le traitement du sommet d_s

Le dernier sommet de λ est le premier sommet traité et il correspond au sommet de dépôt final. Il possède un unique arc entrant de valeur 0. La valeur de l'ensemble associé à l'indice $i = 6$ ne change pas mais l'indice de l'ensemble est ajouté à la pile d'indices.

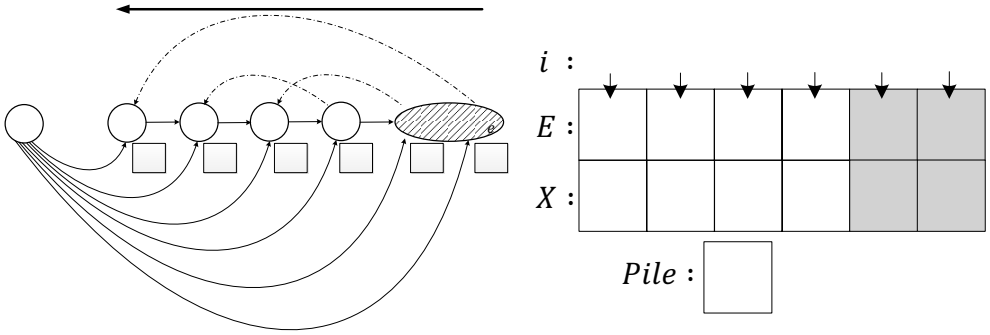


Figure 2.44: Algorithme après le traitement du sommet 4

Le sommet suivant est le sommet 4 associé à l'ensemble 5. L'ensemble prend la valeur 28 à cause de l'unique arc entrant de valeur 28. L'indice 6 est dépilé, comme 0, valeur associée à l'ensemble 6, est plus petit que 28, valeur associée à l'ensemble 5, les deux ensembles sont fusionnés et seul l'indice 5 est empilé.

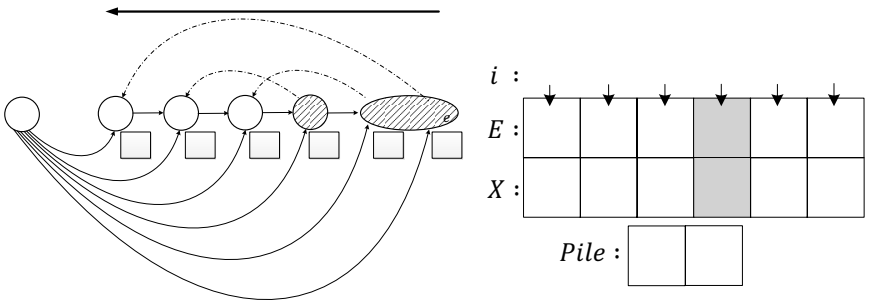


Figure 2.45: Algorithme après le traitement du sommet 3

Le sommet traité à l'itération suivante est le sommet 3 associé à l'ensemble 4. L'ensemble prend la valeur 20 dû à l'unique arc entrant de valeur 20. L'indice 5 est dépilé et comme 28, valeur associée à l'ensemble 5, est supérieure à 20, valeur associée à l'ensemble 4, il n'y a pas d'union, les deux indices sont empilés.

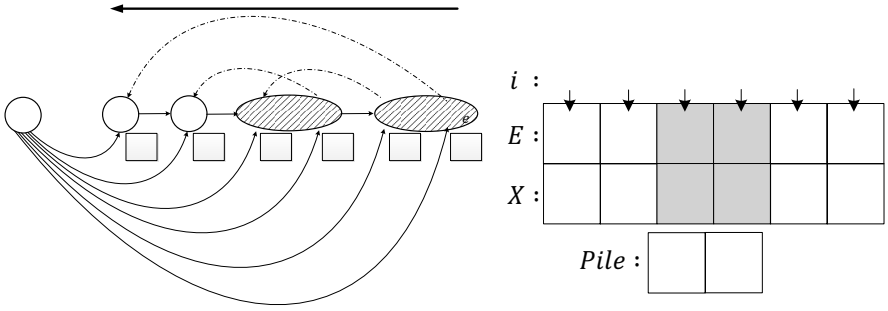


Figure 2.46: Algorithme après le traitement du sommet 2

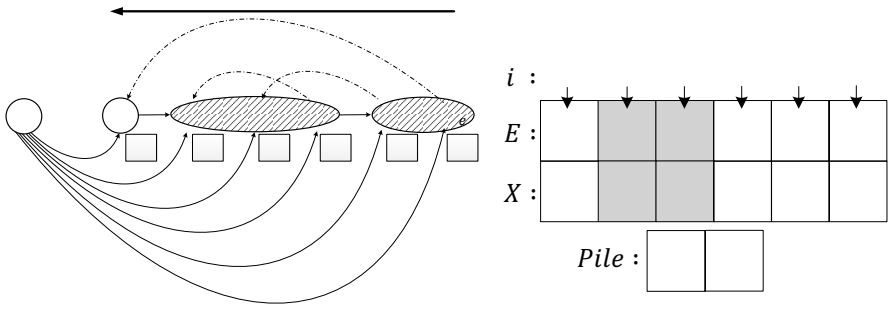


Figure 2.47: Algorithme après le traitement du sommet 1

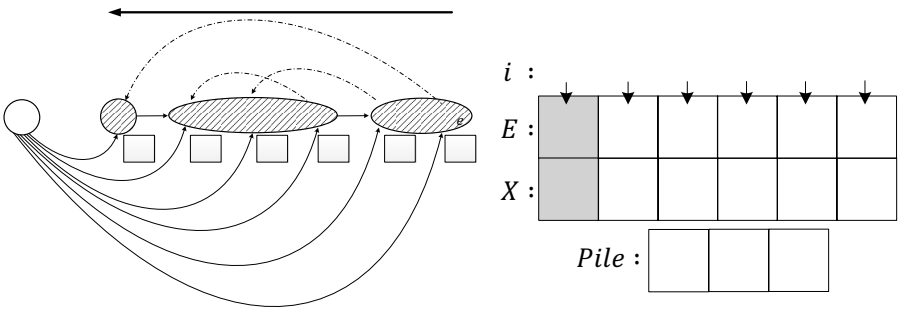


Figure 2.48: Algorithme à la fin de l'étape 2

Cette quatrième étape de l'algorithme a une complexité linéaire, due au fait que l'algorithme travaille dans un cas particulier ou les $union/find$ concernent des ensembles disjoints et

consécutifs. L'algorithme ne peut pas fusionner l'ensemble indicé 1 avec un ensemble autre que celui indicé 2. Dans cette situation comme il a été démontré par (Gabow and Tarjan, 1985) une séquence de m opérations d'$union/find$ sur n_k éléments peut être exécutée en $O(m + n_k)$. Pour la partie 1, chaque sommet est traité une seule fois. Dans cette partie aucune $union$ n'est réalisée, et un nombre de $find$ inférieur à 3 est nécessaire sur chaque sommet. Il correspond au nombre maximal d'arcs entrant. Pour l'étape 2, la fonction $find$ n'est pas utilisée et au total, quand tous les sommets ont été traités, le nombre d'appels à $union$ est égal, dans le pire des cas, au nombre d'éléments c'est-à-dire n_k. Le nombre d'opérations m est donc égal dans le pire cas à $4n_k$ et donc la complexité totale est en $O(n_k)$.

Etape 5 : revenir à la solution

A la fin de l'étape 4 de l'algorithme, les valeurs obtenues représentent les valeurs des X_i minimales pour tous les sommets i de λ. Ces valeurs ont été obtenues sans tenir compte des contraintes de fenêtre de temps maximale comme expliqué précédemment. Dans l'algorithme 8, cette dernière étape correspond aux lignes 24 à 30 qui ont été reprises dans la figure 2.49.

```
24 for i := 1 to n_t do                    // STEP 5 : check violation and compute B_i
25 |   j := find(E, i)
26 |   if (X[j] + cst(G_2, i, 0) ≥ 0) then
27 |   |   violation = true
28 |   end if
29 |   t.B[i] := X[i] + Π[i]
30 end for
```

Figure 2.49: Algorithme de (Firat and Woeginger, 2011) étape 5

L'étape 5 consiste premièrement à vérifier que λ est valide pour les valeurs de X_i obtenues puis dans un second temps à calculer les dates de début de service sur chaque sommet B_i correspondant. Comme expliqué précédemment, λ est valide uniquement s'il n'existe pas de cycle de coût positif et, dans la construction du graphe il a été mis en évidence que si un cycle existe alors il passe par le sommet 0. La dernière étape consiste à utiliser les plus longs chemins de 0 à i qui ont été calculés à l'étape 4 et à ajouter à cette valeur le coût de l'arc entre i et 0. Si pour au moins un sommet i le résultat est positif alors il existe un cycle de coût positif, sinon λ est valide et les valeurs des B_i peuvent être calculées. Avec les valeurs des X_i, les dates de début de service B_i pour chaque sommet sont facilement calculables à l'aide de l'équation (11) et des valeurs Π_i qui ont été calculées à l'étape 1 de l'algorithme 5.

Pour obtenir les valeurs des X_i, la fonction $find$ est appelée à la ligne 25 de l'algorithme 8 sur chaque sommet de λ. La figure 2.50 illustre l'étape 5 de l'algorithme, les valeurs des sommets sont trouvées à l'aide de la fonction $find$ et des vecteurs $E[]$ et $X[]$. Dans cet exemple λ est valide, les résultats obtenus sont stockés dans le tableau $B[]$. Ils correspondent aux valeurs obtenues à l'aide du plus long chemin dans le graphe G.

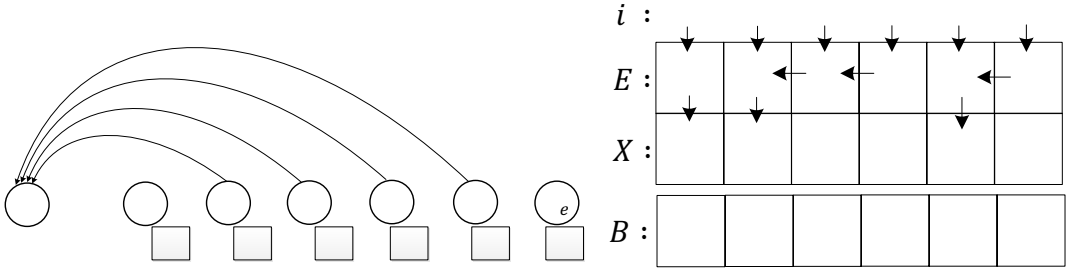


Figure 2.50: Algorithme à la fin de l'étape 3

L'algorithme 8, ci-dessous, présente l'algorithme générale du test de la faisabilité d'une tournée proposée par (Firat and Woeginger, 2011) et dont chaque partie a été présentée dans les paragraphes précédents.

Algorithme 8 is_faisable_tour (Firat and Woeginger, 2011)

Input/Output parameters

t : a structure containing all the trip attributes: n_t, s[], B[], A[], D[]

Global parameters

c_{ij} : distance of the arc (i, j)

Variable parameters

$stack$: stack of integers
G_2 : graph with $n_T + 1$ nodes
$E[]$: vector for sets representation
$X[]$: value associated with each set

Output parameters

$violation$: boolean, $false$ if the trip is valid, $true$ otherwise

Begin

```
1   Π[1] := 0                                          // STEP 1 : compute new variables
2   for i := 2 to t do
3   |   Π[i] := Π[i − 1] + c_{t.s[i−1],t.s[i]}
4   end for
5   G_2 := build_Firat_graph(T[], Π[] )                // STEP 2 : build the graph
6   for i := 1 to t do                                 // STEP 3 : initialize the sets and variables
7   |   E[i] := i; X[i]: = 0 ;
8   end for
9   violation := false                                 // STEP 4 : compute the lowest X_i on nodes
10  stack := null
11  for i := n_t to 1 do
12  |   X[i] := max_{j∈δ_i^-} ( X[find(E, j)] +cst(G_2, i, j))
13  |   do
14  |   |   {j} :=all pop(stack)
15  |   |   if (X[i] ≥ X[j]) then
16  |   |   |   union(E,i, j)
17  |   |   else
18  |   |   |   push(stack, j)
19  |   |   |   stop:= true
20  |   |   end if
21  |   while (stack≠null)&& (stop = false)
22  |   call push(stack,i )
23  end for
24  for i := 1 to n_t do                               // STEP 5 : check violation and compute B_i
25  |   j := find(E, i)
26  |   if (X[j] + cst(G_2, i, 0) ≥ 0) then
27  |   |   violation = true
28  |   end if
29  |   t. B[i] := X[i] + Π[i]
30  end for
31  return {violation}
End
```

Cette fonction de complexité linéaire pour le calcul des dates de début de service au plus tôt peut être adaptée pour calculer les dates au plus tard. Pour cela l'algorithme est similaire mais il faut travailler dans le graphe G_2', version modifiée de G_2 selon le même principe illustré dans la section 2.3.3, et parcourir les sommets de 1 vers n_k en calculant la valeur minimale provenant des arcs entrant.

La ligne 9 :

$$\max_{j \in \delta_i^-} \big(X[find(E,j)] + cst(G_2, j, i) \big)$$

devient dans $G_2{}'$:

$$\min_{j \in \delta_i^-} (X[find(E,j)] + cst(G_2', j, i))$$

Evidement les signes des inégalités sont inversés dans les lignes 11 et 23.

Il est possible d'obtenir des résultats proches de ceux fournis par l'algorithme de (Cordeau and Laporte, 2003), avec des tournées dont les temps de trajet et la date de départ du dépôt sont identiques, mais où les dates de début de service sur les clients peuvent être différentes. Pour cela il faut 4 étapes : la première consiste à évaluer λ avec les dates aux plus tôt pour obtenir E la date de fin au plus tôt ; la deuxième étape consiste à ajouter une fenêtre de temps maximale égale à E sur le dépôt final puis à évaluer λ avec des dates au plus tard ; la troisième étape consiste à modifier la fenêtre de temps minimale sur le dépôt initial pour qu'il soit égal à I. La dernière étape consiste à réévaluer λ avec les dates aux plus tôt. Chaque étape s'effectuant en $O(n)$.

2.3.6. Comparaisons des différentes fonctions d'évaluation

La méthode de (Cordeau and Laporte, 2003) ainsi que celle de (Firat and Woeginger, 2011) adaptée pour obtenir les mêmes valeurs que (Cordeau and Laporte, 2003) ont été implémentées. La complexité linéaire associée à l'algorithme de (Firat and Woeginger, 2011) le rend plus intéressant que l'algorithme proposé par (Cordeau and Laporte, 2003) pour les grandes instances, où beaucoup de clients sont traités par un même véhicule.

Pour les λ de petite taille, les gains en temps générés par l'algorithme de (Firat and Woeginger, 2011) par rapport à l'algorithme de (Cordeau and Laporte, 2003) sont moins significatifs puisque chaque itération de l'algorithme de (Firat and Woeginger, 2011) s'avère plus coûteuse qu'une itération de l'algorithme de (Cordeau and Laporte, 2003).

Pour comparer ces deux fonctions, des λ composés de 1 à 12 clients (origine,destination) ont été générés dans un premier temps. Certains λ sont irréalisables, c'est-à-dire que la séquence de nœuds d'origine et de destination respecte la capacité du véhicule, mais il n'est pas possible de trouver de dates B_i respectant les fenêtres de temps et les contraintes sur le temps de trajet maximal pour les clients ou pour le véhicule. Pour chaque nombre de clients, 10 vecteurs λ sont générés aléatoirement et chacun est évalué un million de fois afin que les temps soient suffisamment grands pour être comparés sans biais liés aux fluctuations de la charge du processeur. Les résultats de la comparaison sont présentés sur la figure 2.51.

Les temps d'exécution des deux fonctions sont très proches pour des λ composés de 1 à 4 clients. Puis pour les λ de tailles supérieures, l'algorithme de (Firat and Woeginger, 2011) est le plus performant. Dans ce cas, les différences de temps entre les deux méthodes sont assez significatives. Par exemple pour des vecteurs λ de 6 clients, l'évaluation de (Cordeau and Laporte, 2003) met plus de 17 secondes contre 12,5 secondes pour la fonction de (Firat and Woeginger, 2011), soit un écart de 40%. L'écart entre ces deux fonctions augmente encore pour des instances de tailles supérieures.

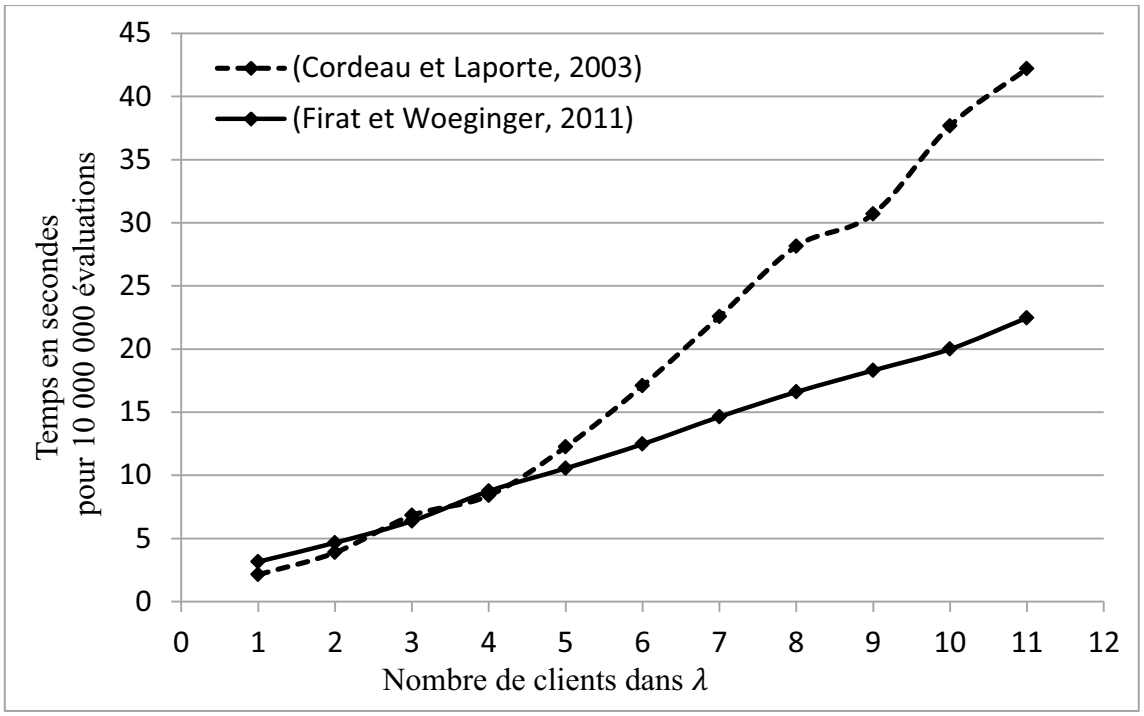


Figure 2.51 : Comparaison des fonctions d'évaluations en fonction du nombre de clients dans λ

Au vu des résultats, l'algorithme de (Firat and Woeginger, 2011) a été retenu dans la métaheuristique pour évaluer les λ, y compris ceux de petites tailles..

2.4. Phase d'initialisation

Trouver des solutions initiales pour le problème du DARP est difficile, à cause notamment des contraintes de temps de transport imposés par les clients et des fenêtres de temps. Cette partie concerne la proposition d'une méthode de construction qui utilise un espace de codage intermédiaire afin d'obtenir des solutions valides de bonne qualité.

Dans la littérature, plusieurs méthodes ont été proposées pour calculer des solutions initiales au DARP. La méthode publiée par (Cordeau and Laporte, 2003) consiste, partant de tournées vides, à choisir aléatoirement un client parmi ceux non traités puis à l'insérer dans une tournée choisie aléatoirement. Pour l'insertion, le sommet d'origine puis le sommet de destination sont positionnés à la fin de la tournée, juste avant le retour au dépôt.

Les solutions ainsi générées respectent les contraintes sur le temps de transport maximal pour tous les clients ainsi que la charge du véhicule. En effet, à la manière d'un taxi, chaque client est déchargé immédiatement après avoir été chargé. Par contre la contrainte sur la durée totale de la tournée ainsi que les contraintes sur les fenêtres de temps des sommets ont été relaxées durant l'étape de construction. Ces solutions relachées peuvent alors être aussi utilisées comme point de départ pour la méthode tabou proposée par (Cordeau and Laporte, 2003).

Une seconde méthode proposée par (Parragh et al., 2010) consiste à affecter une date de début de service à chaque client. Cette date est choisie aléatoirement dans les fenêtres de temps du sommet. Les clients sont ensuite triés en utilisant cette date puis insérés à la fin d'une tournée. Comme pour la solution proposée par (Cordeau and Laporte, 2003), les tournées proposées peuvent être non valides au niveau des fenêtres de temps ou de la durée totale du véhicule. La

contrainte sur le temps de trajet du client est valide car les sommets d'origine et de destination sont consécutifs dans la tournée.

Une troisième méthode proposée par (Parragh and Schmid, 2013) consiste à affecter un unique client dans chaque véhicule. Chaque tournée individuellement est valide, mais comme le nombre de tournées dépasse la taille de la flotte la solution n'est pas réalisable.

Algorithme 9 initialization

Local variable

$G(V, E)$: graph
$P(V_k, E)$: partition
o : pickup nodes order

Output parameters

sol : solution

```
Begin
1    // STEP 1
2    for (i :=1 to n) do
3        for (j := i+1 to n) do
4            res := call check_precedence (i, j)
5            if (res = 1)
6                call add_arc (G, i, j )
7            endif
8            if (res = 2)
9                call add_arc (G, j, i )
10           endif
11       endo
12   endo
13   // STEP 2
14   P := call compute_partition (G)
15  // STEP 3
16   o := call compute_order (P)
17   sol := call generate_solution (o)
18  return sol
end
```

Comme indiqué initialement, la méthode de résolution proposée explore uniquement l'espace des solutions. Cette méthode permet de générer des éléments β qui respectent ou non le nombre maximal de véhicules. Selon les cas, les éléments β modélisent ou non une solution. Le pseudocode de la méthode d'initialisation est décrit dans l'algorithme 9, et il se compose de trois étapes :

- étape 1: pour chaque paire de clients, l'algorithme cherche s'il existe un ordre imposé sur le traitement de leurs sommets d'origine. Ces ordres sont modélisés par des contraintes de précédence, modélisées comme des arcs dans un graphe de précédence ;
- étape 2 : le graphe contenant les contraintes de précédence est ensuite partitionné par niveau ;
- étape 3 : la partition est utilisée pour générer un ordre de traitement des sommets d'origine des clients. Puis une solution est générée en utilisant cet ordre.

Ces trois étapes sont expliquées dans les sections suivantes et sont suivies par une partie présentant les résultats obtenus sur 20 instances de la littérature proposées par (Cordeau and Laporte, 2003).

2.4.1. Les contraintes de précédence entre les clients et la construction du graphe

Pour certains clients (origine,destination) dans un vecteur λ, les fenêtres de temps associées aux sommets peuvent imposer un ordre pour le passage du véhicule. L'étape 1 consiste à identifier l'existence de ces relations de précédence imposées entre les clients traités par un même véhicule. Plus particulièrement, on s'intéresse à déterminer les ordres imposés entre chaque paire de clients. Si un ordre de traitement est imposé, alors une contrainte de précédence existe entre deux clients.

Dans une tournée, le sommet d'origine doit toujours être inséré avant le sommet de destination du même client. Le nombre de tournées possibles composées de 2 clients (origine,destination) est donc uniquement de 6. Soit deux clients, le premier doit aller du sommet i au sommet $(i+n)$ et le deuxième doit aller du sommet j au sommet $(j+n)$. L'ensemble des ordres de traitement possibles est représenté par des vecteur λ_k, $k \in \{1; \dots; 6\}$ suivants :

$$\lambda_1 = \big(i, j, (j+n), (i+n)\big)$$
$$\lambda_2 = \big(i, (i+n), j, (j+n)\big)$$
$$\lambda_3 = \big(i, j, (i+n), (j+n)\big)$$
$$\lambda_4 = \big(j, i, (j+n), (i+n)\big)$$
$$\lambda_5 = \big(j, i, (i+n), (j+n)\big)$$
$$\lambda_6 = \big(j, (j+n), i, (i+n)\big)$$

On peut diviser les λ_k en deux sous-ensembles. Les $\lambda_{i\to j} = \lambda_k$, $k \in \{1,2,3\}$, sont tels que le nœud d'origine du client i se situe avant le nœud d'origine du client j. Le contraire pour les $\lambda_{j\to i} = \lambda_k$, $k \in \{4,5,6\}$.

Un ordre entre deux clients est imposé au véhicule si et seulement si les ordres $\lambda_i$valides appartiennent tous ou même sous-ensemble $\lambda_{i\to j}$ ou $\lambda_{j\to i}$. Si aucun ordre n'est valide ou s'il existe des ordres λ valides dans les deux sous-ensembles, alors aucune contrainte n'est imposée. Il faut donc vérifier dans les deux sous-ensembles qu'il existe au moins un ordre possible.

Dans l'algorithme 10, cette partie est effectuée de la ligne 2 à la ligne 6 pour le premier ensemble et de la ligne 7 à la ligne 11 pour le second. La fonction d'évaluation est simplifiée : elle teste la réalisabilité sans prendre en compte les contraintes de charge et de durée totale de la tournée.

Dans les lignes 12 à 18, l'algorithme vérifie si un seul des deux ensembles, $\lambda_{i\to j}$ ou $\lambda_{j\to i}$, possède au moins une λ réalisable. Si une telle situation se présente, alors une contrainte de précédence existe et l'algorithme retourne la valeur correspondante : 1 si le sommet d'origine de i doit être traité avant celui de j; 2 si le sommet d'origine de j doit être traité avant celui de i; autrement l'algorithme retourne la valeur 3. Cette $3^{ème}$ situation indique soit que les deux ordres de ramassage sont valides soit que les deux clients considérés ne peuvent pas être dans la même tournée.

Algorithme 10 check_precedence

Input parameters

i : client
j : client

Local variables

λ_k : trip corresponding to the permutation lists with k {1…6}

Output parameters

Res : 1 -> i before j
2 -> j before i,
3 -> otherwise

Begin

```
1   k :=1
2   test1:= false
3   while (test1 = false) and (k <= 3) do
4   |{ test1} := call is_feasible_tour(λ_k)
5   |k := k+1
6   End while
7   test2:= false
8   while (test2 = false) and (k <= 6) do
9   |{test2 } := call is_ feasible_tour(λ_k)
10  |k := k+1
11  End while
12  if (test1= true) and (test2 = false) then
13  |return 1
14  End if
15  if (test1= false) and (test2 = true) then
16  |return 2
17  End if
18  return 3
End
```

Le graphe de précédence est construit au fur et à mesure que sont identifiée les contraintes de précédence. Il contient un sommet par client et un sommet fictif noté 0 qui représente la source et qui est lié à tous les autres sommets du graphe. Les contraintes de précédence sont ajoutées sous la forme d'arcs dans le graphe G. Cette étape est présentée dans l'algorithme 9 aux lignes 6 et 9.

Les résultats obtenus dépendent des fenêtres de temps. Un prétraitement peut permettre de réduire les fenêtres de temps sans modifier l'espace des solutions grâce aux autres contraintes du problème. On facilite ainsi le test de réalisabilité des tournées dans l'évaluation mais aussi dans le reste de la méthode. Ce prétraitement est détaillé dans la sous-partie suivante.

a. *Prétraitement* des données initiales

Ce prétraitement restreint les fenêtres de temps. Il être effectué en même temps que le calcul des contraintes de précédence. Il ne modifie pas la complexité de l'algorithme 9 et est valide pour la méthode de résolution du DARP. Ainsi, en réduisant les fenêtres de temps sur les sommets, il est plus facile de s'apercevoir que l'insertion de certains clients dans une tournée n'est pas réalisable sans devoir la réévaluer. C'est le cas notamment dans la phase de recherche locale.

Ce prétraitement est très important sur certaines instances de la littérature dans lesquelles on impose une fenêtre de temps uniquement sur un des deux sommets d'un client. Le prétraitement consiste à utiliser les autres contraintes du problème pour en déduire des fenêtres de temps réduites sur le sommet d'origine (i) ou/et sur le sommet de destination (j) associés au client.

Pour cela les formules suivantes sont utilisées, figure 2.52 :

- $e'_j = \max\{ e_i + d_i + c_{ij} \ ; \ e_j\}$;
- $l'_j = \min\{ l_i + d_i + L \ ; \ l_j\}$;
- $e'_i = \max\{ e_i - d_i - L \ ; \ e_i \}$;
- $l'_i = \min\{ l_j - d_i - c_{ij} \ ; \ l_i\}$.

Ce prétraitement n'est possible que dans le cas où l'inégalité triangulaire est respectée. C'est le cas pour les instances de la littérature, basées sur la distance euclidienne. Il peut être placé avant la fonction *check_precedence* dans l'algorithme 9 et ne doit être effectué qu'une fois pour chaque client.

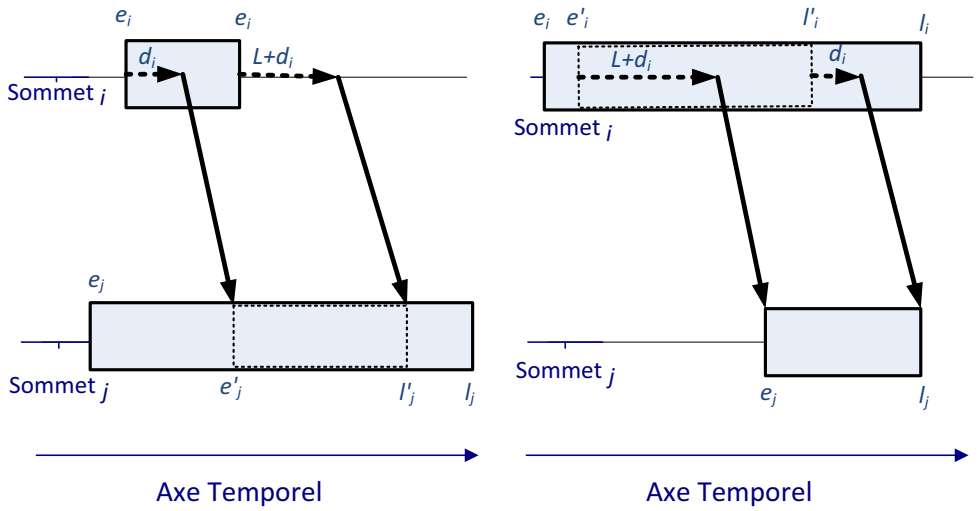


Figure 2.52 : Prétraitement sur les fenêtres de temps

Le prétraitement revient à remplacer les fenêtres de temps d'un client par les dates aux plus tôt et aux plus tard obtenues lors de l'évaluation d'une tournée composée uniquement de ce client.

b. Présence de cycle dans le graphe généré

Il est intéressant de souligner que le graphe généré n'est pas toujours acyclique car il existe des cas où le client A doit être ramassé avant le client B, le client B doit être ramassé avant le client C et pour finir le client C doit être ramassé avant le client A. Dans ce paragraphe un exemple est mis en évidence. L'instance contient 3 clients (origine, destination). Le dépôt est omis afin de simplifier l'exemple. La figure 2.53 représente les distances c_{ij} séparant les différents sommets de l'instance, la figure 2.54 représente les fenêtres de temps min et max associées aux sommets pour le sommet d'origine du client 1, symbolisées par $e_{1^+} = 0$ et $l_{1^+} = 1$.

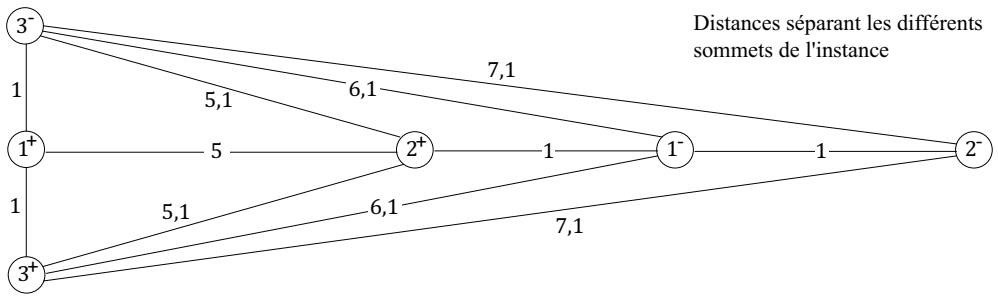


Figure 2.53 : Exemple d'instances engendrant des cycles dans la construction du graphe

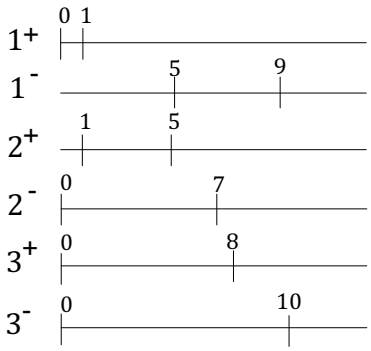


Figure 2.54 : Fenêtres de temps associées aux sommets

Lors du calcul des contraintes de précédence entre les sommets présentés avec la fonction *check_precedence,* le graphe G, illustré par la figure 2.55, est obtenu. Dans ce graphe un cycle composé des clients 1, 2 et 3 existe. Il signifie que si les clients 1 et 2 et 3 sont dans le même véhicule le client 1 doit être traité avant le client 2, le client 2 avant le client 3 et le client 3 avant le client 1. Le détail pour les clients 1 et 3 est illustré sur la figure 56 et sur la figure 57. Ce cycle composé des sommets 1-2-3 est possible à cause de la combinaison entre les contraintes de distance et de fenêtres de temps. En d'autres termes, les clients 1,2 et 3 ne peuvent pas être servis par le même véhicule.

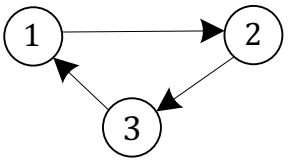


Figure 2.55 : Graphe G obtenu

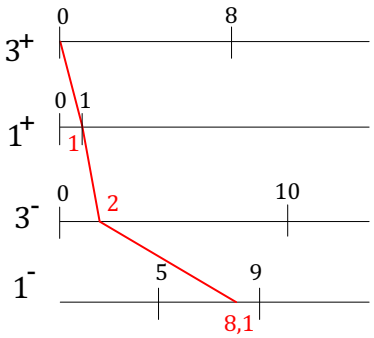


Figure 56 : Preuve de l'existence d'un ordre $\lambda_{3\to1}$ valide

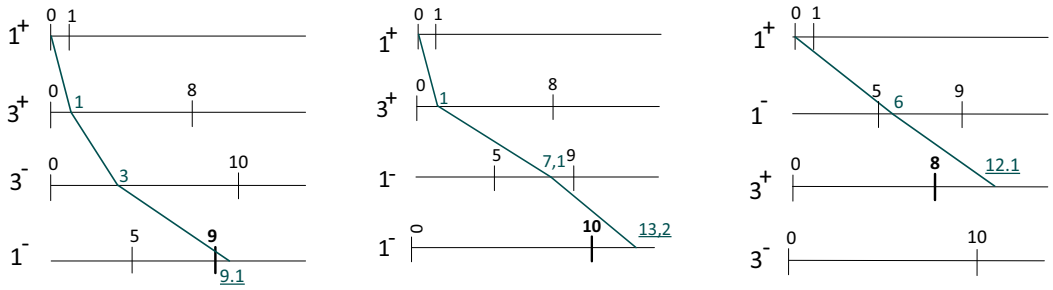


Figure 57 : Preuve de la non-existence d'un ordre $\lambda_{1\to3}$ valide

Cette configuration, assez particulière, ne se produit pas pour les instances classiques de la littérature. Cependant, lorsqu'il arrive sur d'autres instances, il engendre des difficultés dans l'étape suivante de partitionnement du graphe. Ces difficultés doivent être prises en compte,

comme expliqué dans la partie suivante, dans ce cas, la totalité des arcs qui constituent le cycle est supprimée du graphe.

2.4.2. Construction d'une partition

L'objectif est de transformer le graphe G en une partition P. Cette étape correspond à la ligne 14 de l'algorithme 9 avec l'utilisation de la fonction *compute_partition*.

Un sommet source est un sommet sans arc entrant. La partition d'un graphe $G = (V, E)$ est la séparation de V en k sous-ensembles disjoints deux à deux $V_1, \dots, V_k$ tels que $V = \bigcup_{i=1}^{k} V_i$. k est le nombre de niveaux dans la partition. La partition est construite ainsi :

$V_1 \coloneqq sources(G)$

$V_i \coloneqq sources(G \setminus \bigcup_{j=1}^{i-1} V_j)$

Par exemple, soit un graphe G dans lequel les arcs : (1,3) (2,5) (3,4) (3,5) (5,6) (7,4) ont été ajoutés. Le graphe par niveau résultant est illustré par la figure 2.58. La partition associée est alors $V_1 = \{*\}$; $V_2 = \{1,2,7\}$; $V_3 = \{3\}$; $V_4 = \{4,5\}$; $V_5 = \{6\}$.

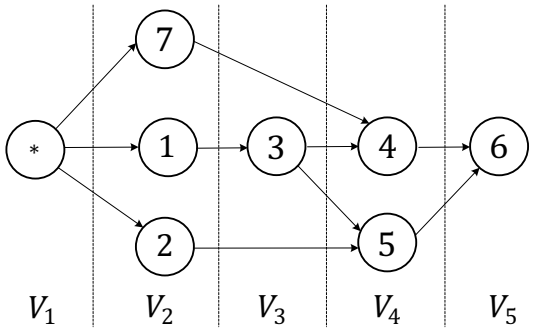


Figure 2.58 : Exemple de partition

Algorithme 11 compute_partition

Input parameters

$G(V, E)$: graph

Output parameters

$P(V_k, E)$: partition with $k \in \mathbb{N}$

Begin

```
1   k :=0
2   while (V ≠ ∅) do
3   |   V_k := ∅
4   |   for all i in V do
5   |   |   modif := false
6   |   |   if (Γ_i^- = ∅) then
7   |   |   |   modif := true
8   |   |   |   V_k := V_k ∪ {i}
9   |   |   |   V := V\{i}
10  |   |   end if
11  |   |   if (modif = true) then
12  |   |   |   k := k + 1
13  |   |   else
14  |   |   |   E_x := cycle_detection(G)
15  |   |   |   E := E\E_x
16  |   |   end if
17  |   end for
18  end while
19  return P
End
```

De la ligne 2 à la ligne 18, l'algorithme 11 construit niveau par niveau la partition. Le processus itératif s'applique tant que tous les sommets de G n'ont pas été traités. Pour cela, on parcourt tous les sommets présents dans V à la recherche des sources, lignes 4 à 17. Lorsqu'une source est trouvée (lignes 6 à 10), elle est supprimée de G et ajoutée au $k^{ème}$ niveau de P. Une fois tous les sommets de V parcourus, le graphe recommence pour le niveau suivant si au moins un arc a été ajouté ligne 12. Sinon le graphe G contient au moins un cycle et que la construction d'une partition est impossible, lignes 10 à 16. Dans ce cas, la totalité des arcs qui constituent le cycle est supprimée de G et l'algorithme recommence la recherche de sources.

2.4.3. Construction d'un ordre sur les clients

L'espace de codage est un espace d'objets qui peut être associé à l'espace des solutions du problème. Plutôt que de construire directement une solution initiale, l'algorithme commence par générer un élément de l'espace de codage qu'il l'utilise ensuite pour générer une solution initiale.

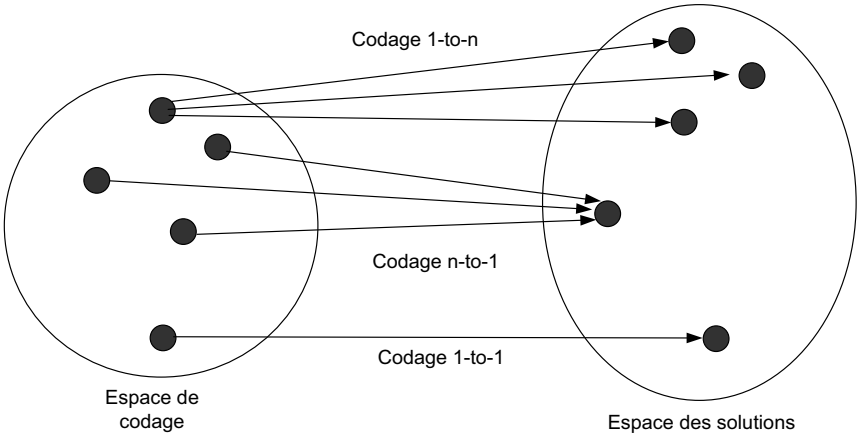


Figure 2.59 : Espaces de codages

Il existe trois codages définis dans (Cheng et al., 1996). Ces codages sont illustrés dans la figure 2.59 :

- le codage 1-to-n, où un élément de codage est associé à plusieurs solutions ;
- le codage n-to-1, où plusieurs éléments de codage sont associés à une solution ;
- le codage 1-to-1, où chaque élément de codage est associé à une unique solution, ce type de codage est le plus recherché.

L'espace de codage utilisé dans cet algorithme est un vecteur d'entiers de taille n, noté o. Chaque élément de ce vecteur représente un client (origine, destination). L'ordre respecte les niveaux obtenus lors de la partition du graphe G, contenue dans P. Pour ordonner les sommets issus du même niveau, un ordre aléatoire est choisi, comme expliqué dans algorithme 12.

2.4.4. Construction de la solution

La construction de la solution consiste à insérer successivement les clients en suivant l'ordre du vecteur o. Le client ne peut être inséré qu'après le dernier sommet d'origine de la tournée considérée. Ce qui limite le nombre de possibilités, et comme l'ordre o respecte les niveaux de la partition, les insertions respectent les contraintes de précédence entre les clients.

Algorithme 12 compute_order

Input parameters
$P(V_k, E)$: partition with $k \in \{0, X\}$

Output parameters
o : order of clients

Begin
1 $i := 0$
2 **for** $k := 1$ to X **do**
3 | **while** ($V_k \neq \emptyset$) **do**
4 | | $i := i + 1$
5 | | $j :=$ call randomly_selected_client(V_k)
6 | | $V_k := V_k \backslash \{j\}$
7 | | $o[i] := j$
8 | **end while**
9 **end for**
10 **return** λ
End

L'algorithme 13 prend en entrée un vecteur o et commence avec une phase d'initialisation qui va de la ligne 1 à la ligne 4. Dans cette phase plusieurs éléments sont initialisés dont : la liste ban à la ligne 2 qui sert à retenir la liste des positions déjà traitées pour un client ; la solution partielle s à la ligne 3 dans laquelle chaque tournée est constituée uniquement des dépôts ; et enfin une pile qui permet à l'algorithme de sauvegarder les étapes intermédiaires à la ligne 4. Chaque élément de la pile est constitué de 3 objets : la solution partielle, l'étape à laquelle se situe cette solution et la liste ban.

Pour chaque client de o, l'algorithme recherche les différentes positions d'insertion possibles dans la solution partielle, lignes 6 à 12. Lorsqu'une tournée valide est générée alors le numéro de la tournée j, la position d'insertion du sommet d'origine p et du sommet de destination d, ainsi que la tournée correspondante et son coût sont stockés dans un ensemble E initialement vide (ligne 8). L'ensemble E a une taille limitée et ne contient que les 10 meilleures insertions en considérant le coût. Seules les insertions qui ne sont pas répertoriées dans la liste ban sont explorées à la ligne 12. Enfin la fonction appelée à la ligne 13 insère le sommet d'origine $pickup$ et le sommet de destination $delivery$ en positon respective p et q dans la tournée courante, avant de l'évaluer à la ligne 14. Si la tournée est valide alors le schéma d'insertion est ajouté à l'ensemble E.

Une fois la totalité des mouvements testés, il existe deux cas : soit l'ensemble E contient au moins une position d'insertion valide lignes 22 à 28, soit il n'en contient pas, lignes 23 à 37.

Dans le premier cas, une des positions de E est choisie. La position qui génère la tournée de coût le plus faible est choisie avec une probabilité de 80% sinon la suivante est choisie avec là-aussi une probabilité de 80% et ainsi de suite.

Dans le second cas, l'ensemble E est vide et il n'existe donc pas d'insertion valide du $i^{ème}$ client dans la solution partielle. L'algorithme revient alors à une solution partielle précédente afin d'explorer un autre choix d'insertion. Pour cela, la pile qui sauvegarde les étapes intermédiaires est dépilée à la ligne 32 de l'algorithme 13. Il peut être intéressant de remonter plus loin dans les solutions partielles précédentes afin de débloquer une situation sans perdre de temps.

Algorithme 13 generate_solution

Input parameters

o : order of clients

Global parameters

n : number of clients in the instance
mii : maximal number of iterations for initialization

Output parameters

s : set of trips
$result$: boolean result of the initialization with
$true$: success and $false$: failed

Begin

```
1   i := 1
2   ban := ∅ ; iter := 0
3   initialized(s)
4   stack := null; push (stack, {s, i, ban})
5   while (i ≤ n) do
6   |  pickup := o[i]
7   |  delivery := o[i] + n
8   |  E := ∅
9   |  for j := 1 to m do
10  |  |  for p := P[j] + 1 to n_j do
11  |  |  |  for d := P[j] + 1 to (n_j + 1) do
12  |  |  |  |  if ((j, p, d) ∉ ban) then
13  |  |  |  |  |  T := call insertion(s[j], pickup, delivery, p, d)
14  |  |  |  |  |  {violation , cout} := call evaluation_T(T)
15  |  |  |  |  |  if (violation = false) then
16  |  |  |  |  |  |        E := E + {j, p, d, T, cout}
17  |  |  |  |  |  end if
18  |  |  |  |  end if
19  |  |  |  end for
20  |  |  end for
21  |  end for
22  |  if (E ≠ ∅) then
23  |  |  {j, p, d, T, cout}:= selection one element of E
24  |  |  ban = ban + ( j, p, d)
25  |  |  push (stack, {s, i, ban})
26  |  |  s[j] := T
27  |  |  ban := ∅
28  |  |  i = i + 1;
29  |  else
30  |  |  iter := iter + 1
31  |  |  if (iter < mii) and (is_not_empty(stack)) then
32  |  |  |  {S, i, ban} := pop(stack)
33  |  |  else
34  |  |  |  result := false
35  |  |  |  return {result , s}
36  |  |  end if
37  |  end if
38  end while
39  result := true
40  return {result, s}
End
```

Le critère pour choisir d'insérer les clients est uniquement le coût. C'est le plus efficace parmi les critères testés. L'avantage de ce critère vient du fait qu'en choisissant les tournées selon coûts les plus faibles, l'algorithme gère en priorité les tournées les moins remplies. Parmi les autres critères qui ont été expérimenté sans parvenir à des résultats aussi intéressants, il y a le temps de trajet des clients et le temps de trajet total.

Comme deux vecteurs o peuvent mener à la construction de la même solution l'espace de codage est du type 1-to-n, figure 2.59.

2.4.5. Résultats obtenus lors de la génération de solutions initiales

Pour montrer l'efficacité de la méthode d'initialisation, le tableau 2.2 contient les résultats obtenus sur les 20 instances proposées par (Cordeau and Laporte, 2003). Ces instances font parties des instances de référence dans la littérature et sont les plus difficiles à résoudre. Elles sont présentées plus en détails dans la partie concernant les résultats numériques.

Les résultats concernent la génération de 1000 solutions initiales pour chaque instance. Le tableau 2.2 est composé de 11 colonnes. Les 4 premières concernent les caractéristiques des instances, avec $name$ le nom, m le nombre de véhicules disponibles, n le nombre de clients et BKS la meilleure solution publiée dans la littérature. Les six colonnes suivantes concernent les résultats obtenus sur les 1000 exécutions pour générer des solutions initiales avec :

- $\#\ infeas.$ le nombre de fois où la procédure a échoué ;
- $\#\ feas.$ le nombre de fois où elle a réussi ;
- $best$ et $gap\ best\ (\%)$ représente les meilleurs résultats obtenus parmi les 1000 tentatives et l'écart par rapport à la BKS ;
- $avg.$ et $gap\ avg.\ (\%)$ représentent la moyenne des résultats obtenus parmi les solutions valides et l'écart par rapport à la BKS.

Enfin la colonne $time$ indique le temps total pour effectuer les 1000 tentatives de génération de solutions initiales, exprimé en minutes pour un ordinateur à 2529 Mflops. Ce nombre de Mflops a été estimé en utilisant les travaux publiés de (Dongarra and Luszczek, 2011) ainsi que les travaux sur le LINPACK publiés dans (Dongarra et al., 1979).

Le tableau 2.2 montre que des solutions valides ont pu être générées pour les 20 instances et qu'il y a eu des échecs uniquement sur l'instance R9a. Les écarts sont en moyenne de 59% par rapport à la meilleure solution connue. Pour obtenir les 1000 solutions il faut en moyenne moins de 4 min, ce qui représente 0,24 s en moyenne pour une solution.

Pour en revenir à l'instance R9a, seuls 891 vecteurs ont conduit à une solution valide sur les 1000. De plus, le temps d'exécution pour cette instance est de 60 min, ce qui est beaucoup plus important que pour les autres instances, y compris R9b, R10a et R10b qui sont de taille supérieure ou égale. Cette différence s'explique par le fait que l'algorithme peut revenir à une solution partielle précédente lorsque la solution partielle courante n'est pas valide. L'algorithme est stoppé après un certain nombre de retours infructueux. Ainsi les 189 tentatives échouées ont pris beaucoup plus de temps d'exécution que les tentatives fructueuses.

Au vu des résultats, 5 ordres différents dans l'initialisation de la méthode de résolution du DARP sont générés. De plus le nombre de retour à des solutions partielles précédentes est limité par la variable globale mii. Ainsi lorsqu'un vecteur ne permet pas à l'algorithme de construire une solution, il est rapidement rejeté. Lorsque plusieurs solutions sont obtenues, la meilleure est sélectionnée. Sur les instances de la littérature, 5 exécutions de l'heuristique de calcul de solutions initiales est suffisant pour obtenir au moins une solution initiale.

Tableau 2.2:
résultats de la génération de solutions initiales

name	m	n	BKS	résultats sur 1000 solutions générées						time
				# unfeas.	# feas.	best	gap best (%)	avg.	gap avg. (%)	
R1a	3	24	190.02	0	1000	220.86	16.23	265.23	39.58	0.01
R2a	5	48	301.34	0	1000	391.37	29.88	451.68	49.89	0.04
R3a	7	72	532.00	0	1000	726.04	36.47	828.68	55.77	0.08
R4a	9	96	570.25	0	1000	813.15	42.60	891.17	56.28	0.17
R5a	11	120	626.93	0	1000	901.31	43.77	981.32	56.53	0.28
R6a	13	144	785.26	0	1000	1152.60	46.78	1250.14	59.20	0.40
R7a	4	36	291.71	0	1000	354.15	21.40	410.17	40.61	0.02
R8a	6	72	487.84	0	1000	692.67	41.99	821.70	68.44	0.15
R9a	8	108	658.31	109	891	1011.26	53.61	1133.19	72.14	59.72
R10a	10	144	851.82	0	1000	1294.30	51.95	1428.20	67.66	14.42
R1b	3	24	164.46	0	1000	206.20	25.38	247.99	50.79	0.01
R2b	5	48	295.66	0	1000	384.97	30.21	435.32	47.24	0.04
R3b	7	72	484.83	0	1000	714.57	47.39	791.89	63.33	0.09
R4b	9	96	529.33	0	1000	775.13	46.44	871.82	64.70	0.17
R5b	11	120	577.29	0	1000	884.43	53.20	965.19	67.19	0.30
R6b	13	144	730.67	0	1000	1108.17	51.66	1190.92	62.99	0.43
R7b	4	36	248.21	0	1000	329.03	32.56	388.22	56.41	0.02
R8b	6	72	458.73	0	1000	641.96	39.94	722.73	57.55	0.09
R9b	8	108	593.49	0	1000	936.81	57.85	1027.51	73.13	0.21
R10b	10	144	785.68	0	1000	1232.89	56.92	1381.26	75.80	1.97
average :							**41.31**		**59.26**	**3.93**

L'étape qui suit l'initialisation est une phase de recherche locale. Elle est expliquée dans la partie suivante.

2.5. La recherche locale

La recherche locale proposée est combinée avec une méthode d'apprentissage. Au cours de la résolution, cette méthode d'apprentissage utilise les résultats obtenus lors des précédentes phases de recherche locale pour adapter ses paramètres. Grace à cette méthode d'apprentissage, la méthode converge plus rapidement. La recherche locale est composée de plusieurs structures de voisinage. Chacune a une probabilité $P[x]$ d'être appelée lors de l'exécution d'une itération de recherche locale. Le pseudocode correspondant est écrit dans l'algorithme 14. Ces différents blocs ont un unique paramètre d'entrée/sortie qui est une solution s.

Chacune des 6 structures de voisinages (blocs) permet d'explorer un voisinage différent de la solution courante. Comme seuls les mouvements améliorants sont acceptés, la solution retournée par une structure ne peut pas être de coût supérieur à la solution initiale.

Les différentes probabilités sont passées en paramètre de la fonction, dans le tableau P. Après que les blocs aient été exécutés, lignes 24 à 32, les tableaux $Succ$ et $Fail$ sont mis à jour en fonction de l'amélioration réalisée. Le tableau T est initialisé avec des valeurs 0 à la ligne 5. Il permet de savoir quels blocs ont été appelés à chaque itération de la recherche locale. La variable $saved\_cost$ est initialisée à la ligne 2. Elle est mise à jour lors de chaque amélioration, ligne 22, pour savoir si la solution a été améliorée durant cette exécution. Si l'amélioration n'est pas considérée comme suffisante, ligne 25, alors la variable na est augmentée. Si na atteint une valeur supérieure à npi, alors l'exécution est arrêtée. Lorsqu'un gain assez important est réalisé, na est réinitialisée et le nouveau coût est sauvegardé, lignes 28 et 29.

Algorithme 14 Local_search

Input parameters

P : array of probabilities for each block

Input/output parameters

s : a solution
$succ[]$: array of success for each block
$fail[]$: array of failure for each bloc

Local variable

$Bloc[]$: save which block is used during an iteration

Global parameters

nls : maximal number of iterations
npi : number of unproductive iterations
saved_cost : saved cost of the solution

Begin

1 $i := 0$
2 $saved\_cost := s.cout$
3 **while** ($i < iter\_max$) **and** ($na \leq npi$) **do**
4 | i := i + 1
5 | $\forall i = \{1,\ldots,7\}\ T[i] := 0$
6 | **with probability** $P[1]$
7 | | call r_4_Opt(s)
8 | | $Bloc[1] := 1$
9 | **with probability** $P[2]$
10 | | call cut_and_paste(s)
11 | | $Bloc[2] := 1$
12 | **with probability** $P[3]$
13 | | call worst_customer_ejection(s)
14 | | $Bloc[3] := 1$
15 | **with probability** $P[4]$
16 | | call Relocate(s)
17 | | $Bloc[4] := 1$
18 | **with probability** $P[5]$
19 | | call 2_Opt*(s)
20 | | $Bloc[5] := 1$
21 | **with probability** $P[6]$
22 | | call exchange(s)
23 | | $Bloc[6] := 1$
24 | **if** ($abs(saved\_cost - s.cost) < epsilon$) **then**
25 | | $na := na + 1;$
26 | | $\forall i \in \{k \in \{1,\ldots,6\} \mid Bloc[k] = 1\} : Fail\ [i] := Fail\ [i] + 1$
27 | **else**
28 | | $na := 0;$
29 | | $saved\_cost := s.cost$
30 | | $\forall i \in \{k \in \{1,\ldots,6\} \mid Bloc[k] = 1\} : Succ\ [i] := Succ\ [i] + 1$
31 | **end if**
32 **end while**

end

2.5.1. Prérequis, fonction d'insertion de client

Il peut être intéressant de présenter la fonction d'insertion des clients dans une tournée. Cet algorithme est utilisé dans les différents blocs de recherche locale qui sont présentés ainsi que dans les algorithmes de mutation qui sont abordés dans la partie suivante concernant l'exploration de l'espace des solutions.

L'algorithme 15, présente l'algorithme permettant d'insérer un client à la meilleure position dans une tournée donnée. Il prend en entrée un vecteur λ, le sommet d'origine et de destination ainsi qu'une variable mc représentant le détour maximal qui peut être induit par

l'ajout du nouveau client. Par défaut, la valeur associée à mc représente le gain obtenu lors de la suppression du client de sa précédente tournée. L'algorithme utilise les contraintes de précédence, obtenues durant la phase d'initialisation, pour supprimer des positions d'insertions impossibles. Cette étape correspond aux lignes 3 à 9. Dans les lignes 11 à 28, l'algorithme explore les différentes possibilités d'insertions. Si le coût du détour est intéressant, ligne 14, *i.e.* il est inférieur à mc, alors le vecteur λ' correspondant est évalué. Si λ' est valide alors il est sauvegardé dans t_best et la valeur de mc est modifiée pour explorer uniquement les solutions de coût inférieur à celui de la tournée générée.

Algorithme 15 insertion_client

Input parameters

t	:	a trip which contains all the trip variables like : n_t, s[], B[], A[], D[]
p	:	pickup node of the client
d	:	delivery node of the client
mc	:	maximal value for the insertion

Local variable

i, j	:	courant position of insertions nodes
t', t_best	:	trips

Global parameters

P	:	partition graph

Begin

1 $t_{best} = t; find := false;$
2 $i := 2;\ stop := false$
3 **repeat**
4 | $q := t.s[i]$
5 | **if** $(is\_a\_pickup(q))$ **AND** $\left(exist\left(Arc(P,p,q)\right)\right)$ **then**
6 | | $stop := true$
7 | **end if**
8 | $i := i + 1$
9 **until** $(i < n_t)$ **AND** $(stop = false)$
10 $stop := false$
11 **while** $(i > 1)$ **AND**$(stop = false)$ **do**
12 | $j := n_t + 1$
13 | **while** $(i < j)$ do
14 | | **if** $(cout\_insertion(t,p,d,i,j) < mc)$ **then**
15 | | | t' := call $insert(t,p,d,i,j)$
16 | | | $\{violation, new\_cost\}$:= **call** $evaluation\_T$(t')
17 | | | **if** $(violation = false)$ **then**
18 | | | | $find := true; mc := new\_cost - t.cout;\ t\_best := t'$
19 | | | **end if**
20 | | **end if**
21 | | $j := j - 1$
22 | **end while**
23 | $i := i - 1$
24 | $q := t.s[i]$
25 | **if** $(is\_a\_pickup(q))$ AND $\left(exist\left(Arc(P,q,p)\right)\right)$ **then**
26 | | $stop := true$
27 | **end if**
28 **end while**
29 **return** $\{find, t\_best\}$
end

La complexité de cette fonction est en $O(n_t^3)$. Dans le pire des cas, il y n_t^2 positions d'insertion possibles et l'insertion comme l'évaluation est en $O(n_t)$. Pour randomiser cette fonction et obtenir une des meilleures insertions, le plus simple est de remplacer la ligne 14

if ($cout\_insertion(t, p, d, i, j) < mc$**) then**

par :

if $((find = false)$ **OR** $(alea \leq Q))$ **AND (**$cout\_insertion(t, p, d, i, j) < mc$**) then**

avec $alea$ un nombre choisi aléatoirement et Q la probabilité d'évaluer un mouvement. Ainsi toutes les possibilités sont testées jusqu'à ce qu'un λ valide soit trouvé. Après quoi seulement une partie des λ est évaluée. Cette modification simple permet de randomiser cette fonction sans introduire de nouvelles variables autres que $alea$ et Q tout en garantissant toujours de retourner une solution si au moins une insertion est possible.

A la ligne 14, la procédure $cout\_insertion(t, p, d, i, j)$ retourne la différence entre le coût de la tournée t avec les sommets p d'origine et d de destination du client en position i et j, et le coût de la tournée initiale. L'évaluation de ce détour peut être calculée en $0(1)$.

2.5.2. Les différents blocs de la recherche locale

Les blocs constituent différents structures de voisinage et donc différent types de mouvements qui peuvent être réalisés sur une solution courante. Chaque mouvement génère une nouvelle solution qui est qualifiée de "voisine" avec la solution initiale. La recherche locale proposée se compose de 6 blocs, 5 sont des mouvements inter-tournée et 1 est un mouvement intra-tournée (le r-4-opt*).

*Le mouvement r-4-opt**

Le mouvement r-4-opt* est une version limitée du mouvement *k-opt* proposé par (Lin and Kernighan, 1973), qui consiste d'abord à choisir k arcs dans une tournée. Ces arcs sont supprimés puis l'ensemble des possibilités pour reconstruire une tournée est essayé. Il a été défini pour le TSP. Une version appliquée au DARP a été proposée par (Braekers et al., 2014). Elle consiste à sélectionner uniquement des arcs consécutifs dans une même tournée. Ce processus est nommé r-4-opt. Dans une tournée, pour chaque ensemble de quatre arcs consécutifs, illustrés figure 2.60, un ensemble de 5 nouveaux λ est évalué, illustré figure 2.61 et figure 2.62. Les mouvements de la figure 2.61 modifient la totalité des 4 arcs sélectionnés alors que les mouvements de la figure 2.62 en modifient uniquement 3.

Le coût associé à un nouveau λ peut être calculé en 0(1). Un test de réalisabilité est effectué uniquement si le coût est strictement plus petit que le coût de la tournée initiale. Si λ est valide, la tournée initiale est remplacée par la nouvelle et l'algorithme recommence en prenant les 4 derniers arcs de la nouvelle tournée.

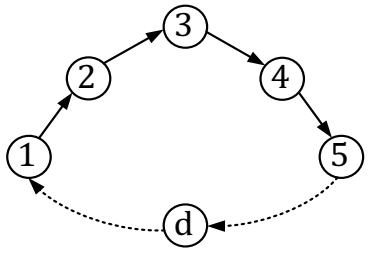


Figure 2.60 : Représentation de la tournée initiale

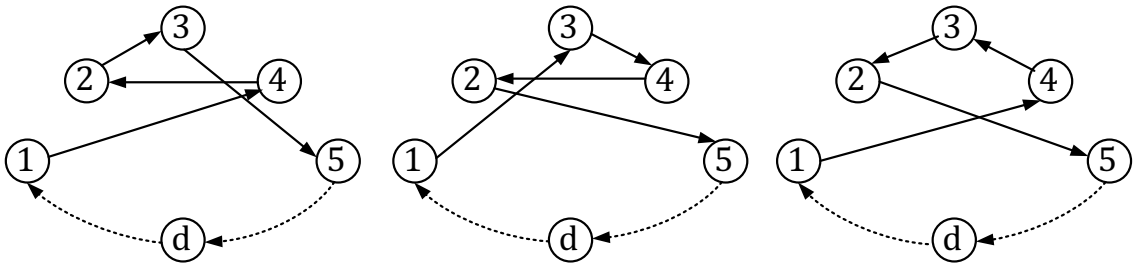


Figure 2.61 : Les trois mouvements du bloc r-4opt dans une tournée*

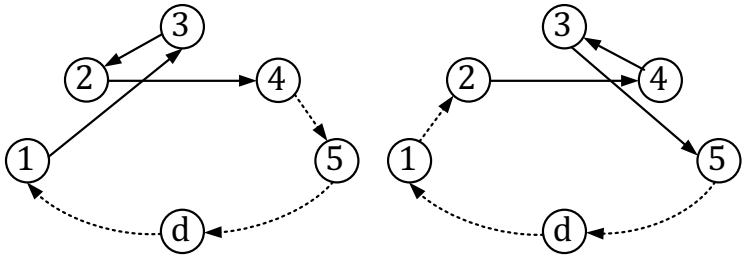


*Figure 2.62 : Deux autres mouvements explorés dans le bloc r-4opt**

Le mouvement "cut and paste"

Le "cut and paste" est un mouvement qui est aussi utilisé dans la mutation lors de l'exploration de l'espace par ELS. Ce mouvement consiste à sélectionner deux vecteurs λ_i et λ_j issus de deux tournées différentes de la solution, de les couper en deux puis de les recombiner en deux nouveaux vecteurs λ_i' et λ_j', qui sont évalués afin d'obtenir 2 tournées. Ce mouvement utilisé sous le nom de "crossover" dans les algorithmes génétiques pour des problèmes de tournée comme le TSP (Grefenstette et al., 1985) pose plusieurs problèmes pour être appliqué au DARP.

Premièrement, les λ_i' et λ_j' générés peuvent ne pas respecter les fenêtres de temps et les contraintes de temps de trajet des clients. Pour réduire ces situations, un point de coupe est sélectionné aléatoirement sur λ_i. Puis le deuxième est choisi de manière à ce que les dates de début de service dans λ_j soient inclues dans les fenêtres de temps des sommets situés avant et après la première coupure.

Le deuxième problème concerne les sommets origine et destination des clients qui peuvent se retrouver de part et d'autre du point de coupe. Or ils doivent appartenir au même vecteur, il faut donc choisir dans lequel les regrouper.

L'algorithme fonctionne en plusieurs étapes. Premièrement deux vecteurs $\lambda_1$et λ_2 associés à deux tournées de la solution sont choisies (figure 2.63) ainsi qu'un point de coupe dans chaque tournée. Les clients dont les sommets d'origine et de destination sont séparés par le point de coupe sont marqués (figure 2.64).

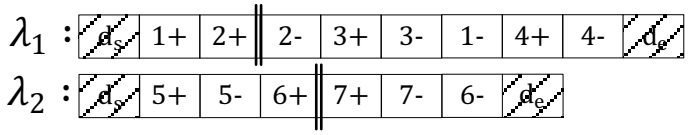


Figure 2.63 : Représentation des vecteurs initiaux et des points de coupe

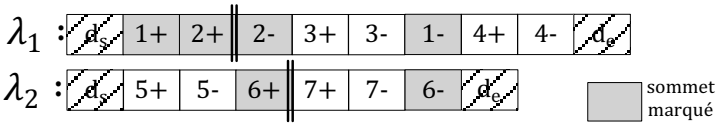


Figure 2.64 : Représentation des vecteurs initiaux après le marquage des sommets

Ensuite la construction des nouveaux vecteurs commence. Un vecteur λ_1' est construit en recopiant les sommets non-marqués de λ_1 précédant le point de coupe puis les sommets non-marqués de λ_2 qui sont situés après le point de coupe (figure 2.65).

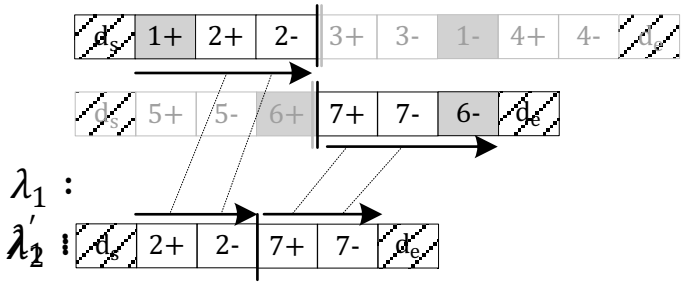


Figure 2.65 : Construction du vecteur t_1'

Un processus similaire est appliqué au vecteur λ_2' avec cette fois le début du vecteur λ_1 et la fin de λ_2(figure 2.66).

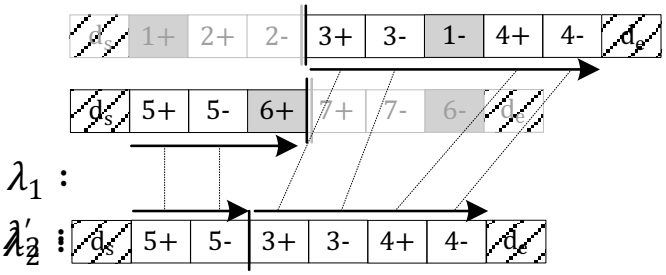


Figure 2.66 : Construction du vecteur t_2'

Pour que la solution reste valide, la dernière étape consiste à réinsérer les sommets marqués. Pour cela, les clients sont choisis dans un ordre aléatoire et sont réinsérés dans un des deux vecteurs à la meilleure position possible comme le montre la figure 2.67.

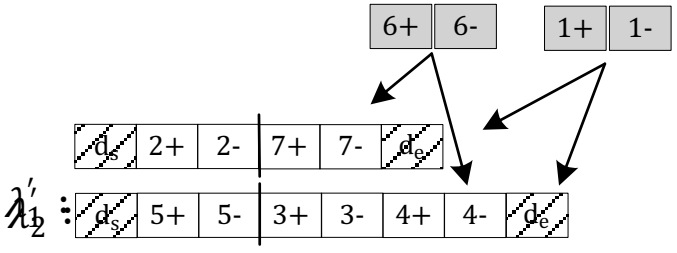


Figure 2.67 : Insertion des clients marqués

Le vecteur dans lequel est réinséré le client et aussi choisi aléatoirement. Si l'insertion dans le 1^{er} véhicule n'est pas possible alors la deuxième insertion est testée dans le second. Le

mouvement est conservé uniquement si les deux vecteurs sont valides et que la somme des tournées finales est inférieure à la somme des tournées initiales (figure 2.68).

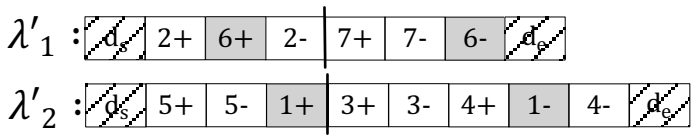


Figure 2.68 : Les vecteurs finaux obtenus

Le mouvement "worst_customer_ejection"

Ce mouvement consiste à rechercher, dans la totalité de la solution, les clients qui entraînent les plus importants détours pour les véhicules. Parmi les clients sélectionnés l'un est choisi aléatoirement, afin qu'il soit réinséré dans une autre tournée.

Les tournées sont parcourues successivement suivant un ordre aléatoire. L'insertion est faite à l'aide de la fonction $insertion\_client$ jusqu'à obtenir une solution ou jusqu'à avoir essayé l'ensemble des tournées.

Le mouvement "relocate"

Ce mouvement est proposé par (Braekers et al., 2014). Une tournée est choisie aléatoirement et les clients qui la composent sont retirés successivement puis sont réaffectés à une autre tournée **(**cette tournée est sélectionnée aléatoirement**)**. Pour tester l'insertion du client, la fonction $insertion\_client$, qui a déjà été présentée, est utilisée. Les clients de la tournée initiale λ_1 sont parcourus suivant l'ordre de passage du véhicule. Pour randomiser cette fonction, le sommet initial de parcours de la tournée est choisi aléatoirement comme dans la figure 2.69.

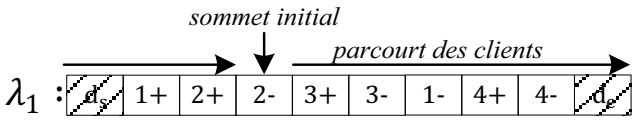


Figure 2.69 : Ordre de traitement des clients de la tournée

Le mouvement 2-opt inter-tournée*

Le bloc nommé "2-opt*" a été proposé pour le TSP par (Potvin and Rousseau, 1995) puis il a été adapté au DARP par (Braekers et al., 2014). Ce mouvement modifie deux tournées. Il consiste, pour chacune des tournées, à supprimer un arc puis à reconstituer deux nouvelles tournées, comme illustré dans la figure 2.70.

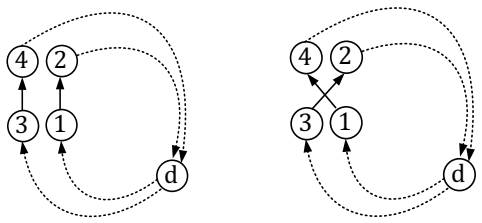


Figure 2.70 : Mouvement 2-opt pour le DARP*

Pour le DARP, les arcs supprimés sont des arcs sur lesquels les véhicules sont vides ainsi aucun couple de sommets d'origine et de destination ne sont séparés. De plus, le sens sur les arcs précédents et suivants les arcs sélectionnés ne doit pas être modifié pour ne pas risquer que les sommets d'origine des clients soient traités après les sommets de destination.

Ce mouvement fait partie de la liste des mouvements accessibles par le bloc "cut and past". S'intéresser uniquement aux points de coupe pour lesquels le véhicule est vide fait que les mouvements sont très rapides car aucune insertion n'est à effectuer. De plus le nombre de points de coupe est assez limité dans une solution. Il est donc intéressant de développer une fonction spécifique à ces mouvements et ainsi pouvoir tester l'ensemble des 2-opt* possibles entre deux tournées.

Le mouvement "exchange"

Enfin la procédure "exchange" consiste à choisir deux clients aléatoirement puis à regarder si le coût de la solution diminue lorsque leurs positions dans les tournées sont échangées, c'est-à-dire lorsque le sommet d'origine ainsi que le sommet de destination des deux clients sont permutés. Si le coût diminue, les vecteurs correspondant aux tournées après la permutation dans lesquels (ou lequel) les clients se trouvent sont alors évalués et si le résultat est valide le mouvement est conservé. Ce mouvement à des performances plus intéressantes sur les instances avec des fenêtres de temps assez larges.

Ce mouvement a été proposé par (Braekers et al., 2014) où l'ensemble des clients de deux tours est traité. Dans la version développée ici, à chaque exécution, les clients sont choisis aléatoirement et indépendamment de la tournée dans lequel ils se situent.

2.5.3. Evolution des paramètres et mécanisme d'apprentissage

L'efficacité des différents blocs de la recherche locale dépend des instances et de la zone de l'espace des solutions qui est explorée par la méthode. C'est pour cela qu'une stratégie d'apprentissage a été mise en place afin de faire évoluer le paramétrage de la recherche locale automatiquement pour qu'elle s'adapte non seulement à l'instance considérée mais aussi à la position courante dans l'exploration de l'espace des solutions. Ces paramètres évoluent de manière ponctuelle tout au long du déroulement de la métaheuristique et facilite la convergence de la méthode. Les paramètres évoluent en fonctions des résultats pour chacun des blocs de la recherche locale obtenus depuis la dernière modification.

Comme expliqué dans l'algorithme 14, à chaque itération un ensemble de blocs est exécuté. Chaque bloc a une probabilité spécifique d'être choisi. Cette probabilité est initialisée arbitrairement 0.25. Elle évolue au cours de la méthode en fonction des résultats obtenus par les différents blocs.

Dans l'algorithme 14, les tableaux $succ$ et $fail$ sont mis à jour à chaque itération en fonction des mouvements utilisés lors de l'itération ainsi que du résultat obtenu. Si le gain est assez important alors le tableau $succ$ est incrémenté pour les blocs ayant été appelés, même s'ils n'ont pas tous contribués à améliorer la solution. Si l'amélioration n'est pas suffisante, alors ce sont les valeurs du tableau $fail$ qui sont incrémentées.

Il est ainsi possible de compter le nombre de fois où le bloc a été appelé avec :

$$n(i) := S \qquad (i) + Fail(i) \text{ avec } i \in \{1 \dots 6\}$$

Le ratio de réussite de chaque bloc est noté:

$$q(i) = {Succ(i)}\Big/{n(i)} \text{ avec } i \in \{1 \dots 6\}$$

Lorsque le nombre d'exécutions de recherche locale est suffisant, les probabilités d'exécuter chaque bloc sont mises à jour en utilisant la formule suivante :

$$P'[i] := \max_{i \in \{1 \dots 6\}} \left(0.05; \frac{95}{q_{max}} . q_i\right) \; avec \; q_{max} = \max_{i \in \{1 \dots 6\}} \left(q(i)\right)$$

Ainsi le bloc avec le plus grand ratio de réussite aura une probabilité de 0,95 d'être exécuté dans les prochaines recherches locales. Les autres blocs ont des probabilités qui dépendent linéairement de l'écart avec q_{max}, figure 2.71. Un bloc avec un ratio de réussite deux fois moins important aura une probabilité deux fois moins élevée soit $0{,}475$. Pour ne pas supprimer totalement des blocs de la recherche locale, la probabilité minimale est fixée à 0,05.

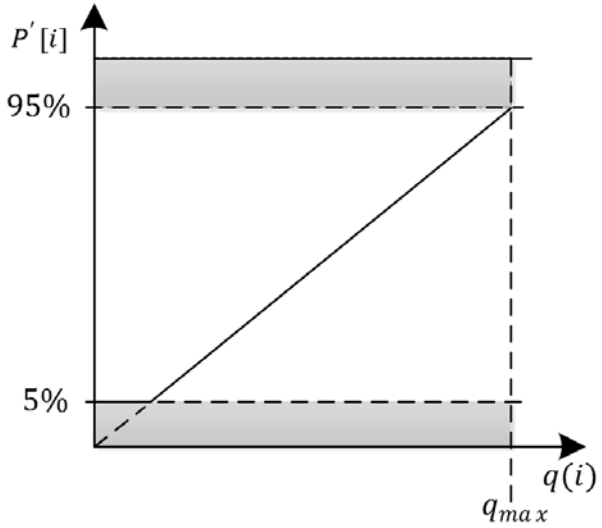


Figure 2.71 : Mise à jour des probabilités d'exécution des blocs en fonction des résultats obtenus

2.5.4. Résultats du mécanisme d'apprentissage

La convergence est plus rapide avec la recherche locale associée à la méthode d'apprentissage. La figure 2.72 est un exemple sur l'instance pr03 proposée par Cordeau et Laporte, qui montre l'évolution de la meilleure solution en fonction du temps. Dans cet exemple, les courbes ont une trajectoire très différente après 12 s. La méthode avec apprentissage converge alors vers une solution proche de la meilleure solution connue alors que l'autre méthode reste aux alentours de 1% de la 25ème à la 40ème seconde.

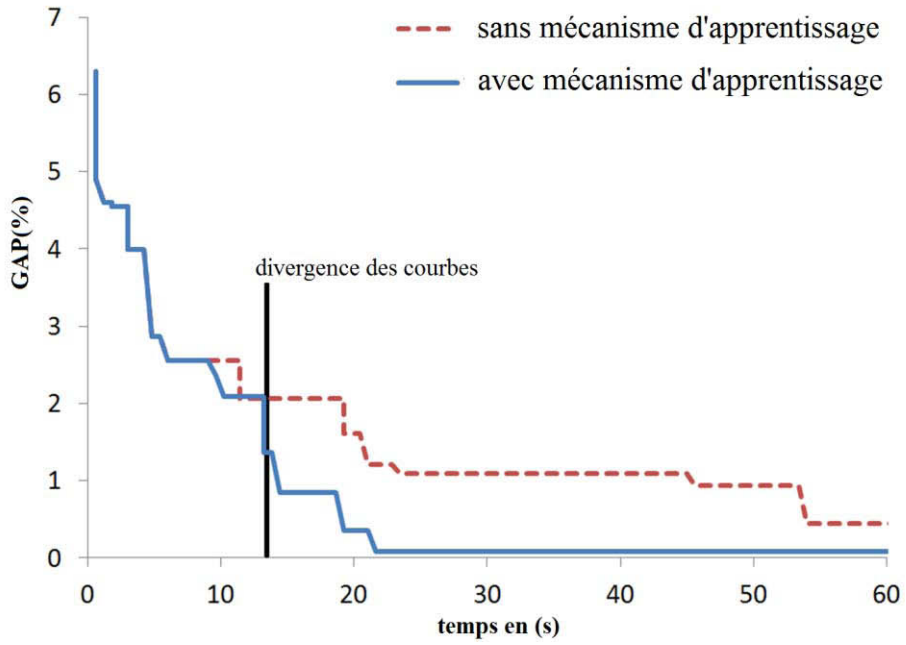


Figure 2.72 : Mise à jour des probabilités d'exécution des blocs en fonction des résultats obtenus

La figure 2.73 et la figure 2.74 illustrent l'évolution des probabilités d'exécution des différents blocs de la recherche locale au cours de la méthode pour deux instances différentes. Les figures montrent que les probabilités n'évoluent pas de la même manière sur ces deux instances. Notamment pour l'instance a7-84, le bloc 5 (2-opt* inter-tournée) est le plus efficace alors que pour l'instance r4b, ce sont les bloc 1(r-4-opt*) et 3 (worst_customer_ejection) qui sont les plus efficaces.

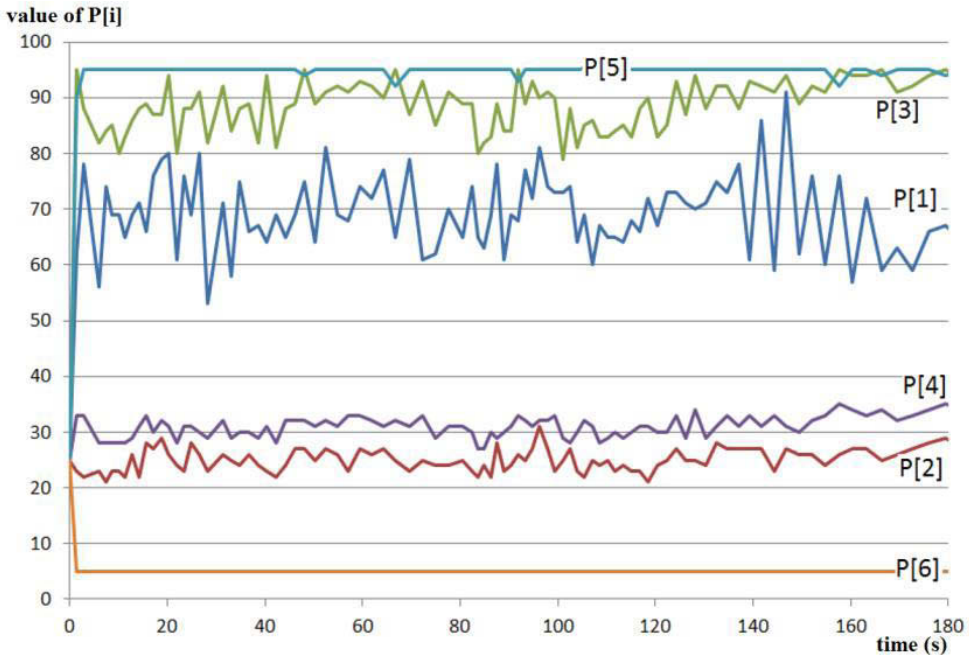


Figure 2.73 : Evolution de P sur l'instance a7-84 proposée par (Ropke et al., 2007)

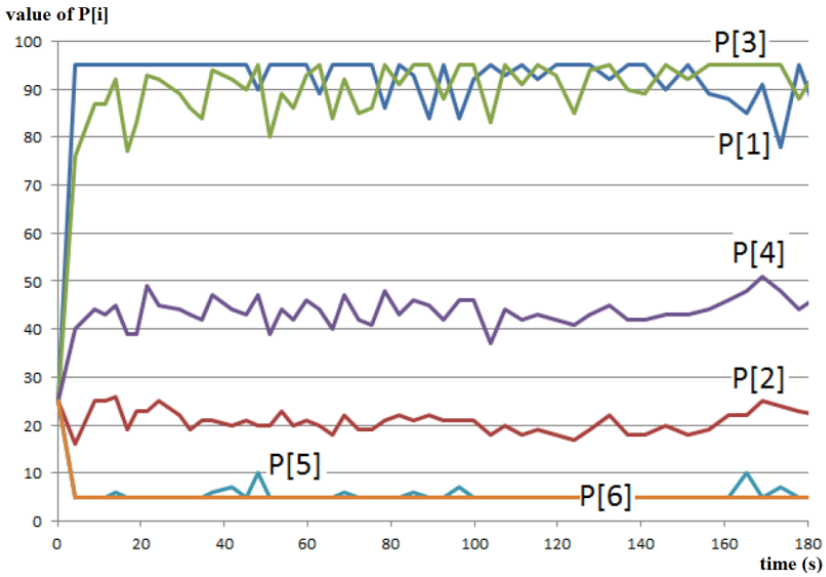


Figure 2.74 : Evolution de P sur l'instance r4b proposée par (Cordeau and Laporte, 2003)

Enfin, le tableau 2.3 montre les résultats obtenus sur les instances de Cordeau et Laporte avec à gauche la méthode sans apprentissage et à droite la méthode avec apprentissage. Ces résultats sont obtenus après 2 min d'exécution pour les deux méthodes. Les solutions initiales ainsi que les valeurs d'initialisation de l'aléatoire des deux méthodes sont identiques. Les

colonnes sont : *name* le nom de l'instance, *m* le nombre de véhicules disponibles, *n* le nombre de clients, *BKS* la meilleure solution connue.

Puis gap(%) représente l'écart entre la meilleure solution obtenue et la *BKS ; T** la date à laquelle la solution a été obtenue et *T* le temps total d'exécution de la méthode. Les résultats montrent que la méthode avec apprentissage a un gap moyen de 1.21% contre 1,57% pour la méthode sans apprentissage. La méthode avec apprentissage a un résultat plus proche de la BKS pour 15 des 19 instances.

Tableau 2.3:
résultats au bout de 2 min sur les instances proposées par (Cordeau and Laporte, 2003)

				sans apprentissage			**avec apprentissage**		
name	**m**	**n**	**BKS**	$\boldsymbol{gap.}$ **(%)**	**T***	**T**	$\boldsymbol{gap.}$ **(%)**	**T***	**T**
R1a	3	24	190.02	0.00	0	2.00	0.00	0.00	2.00
R2a	5	48	301.34	0.00	0.03	2.00	0.00	0.04	2.00
R3a	7	72	532.00	0.23	0.2	2.00	0.00	0.91	2.00
R4a	9	96	570.25	1.09	0.32	2.00	0.15	1.08	2.00
R5a	11	120	626.93	2.02	0.55	2.00	1.67	1.91	2.00
R6a	13	144	785.26	2.73	1.12	2.00	3.06	1.24	2.00
R7a	4	36	291.71	0.00	0	2.00	0.00	0.00	2.00
R8a	6	72	487.84	0.54	1.53	2.00	0.73	1.91	2.00
R9a	8	108	658.31	2.20	1.01	2.00	2.51	1.56	2.00
R10a	10	144	851.82	2.64	1.82	2.00	1.86	1.16	2.00
R1b	3	24	164.46	0.00	0	2.00	0.00	0.00	2.00
R2b	5	48	295.66	0.01	1.04	2.00	0.00	1.12	2.00
R3b	7	72	484.83	0.56	1.56	2.00	0.20	0.81	2.00
R4b	9	96	529.33	2.32	1.93	2.00	0.97	1.78	2.00
R5b	11	120	577.29	3.45	0.63	2.00	0.60	1.97	2.00
R6b	13	144	730.67	4.50	0.22	2.00	3.18	1.55	2.00
R7b	4	36	248.21	0.00	0.03	2.00	0.00	0.02	2.00
R8b	6	72	458.73	1.21	1.94	2.00	1.69	1.08	2.00
R9b	8	108	593.49	2.42	0.98	2.00	2.47	1.44	2.00
R10b	10	144	785.68	5.51	1.78	2.00	5.13	1.99	2.00
			Avg. Time		0.83	2.00		1.08	2.00
				$\overline{\boldsymbol{gap}}$	1.57		1.21		
			Nb. Best $\overline{\boldsymbol{gap}}$		9		15		

2.6. Le parcours de l'espace

2.6.1. La métaheuristique ELS (Evolutionary Local Search)

ELS est un schéma métaheuristique qui est une extension de la métaheuristique ILS (*Iterated Local Search*). ILS consiste à alterner des procédures de mutation et de recherche locale en partant d'une solution initiale. A chaque itération la solution obtenue est utilisée comme solution courante si et seulement si le coût de la nouvelle solution est inférieur à la solution précédente. Le schéma ELS a été introduit pour la première fois par (Wolf and Merz, 2007). Les phases de mutation et de recherche locale s'enchaînent de la même manière que dans ILS mais la différence est qu'à chaque itération une population de solutions voisines de la solution courante est générée. Chaque élément de cette population est noté S_i dans la figure 2.75. Sur

chacune de ces solutions une recherche locale est appliquée. Parmi l'ensemble des solutions générées, la meilleure solution S_{bv}devient la nouvelle solution courante S_{pere}. La solution courant évolue à chaque itération, même si les solutions générées sont de moins bonne qualité. La figure 2.75 est une représentation schématique de l'algorithme d'ELS.

Dans les parties précédentes, différentes étapes ont été présentées, comme les phases d'initialisations en (1) sur la figure et de recherche locale (rl). Dans la partie suivante, les phases de mutation (2) ainsi que les outils de diversifications mis en place sont abordés.

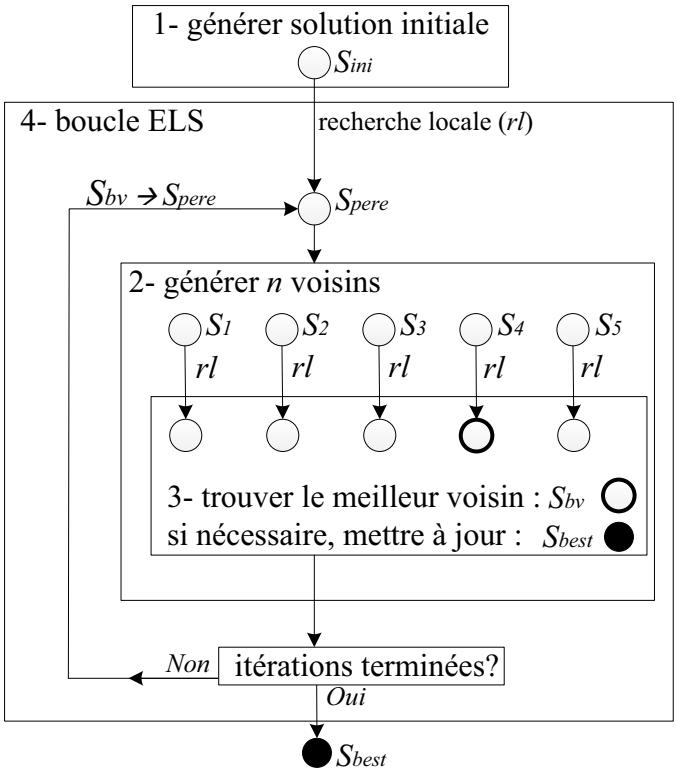


Figure 2.75: Schéma d'ELS

2.6.2. Les mutations

L'algorithme 16 est la représentation globale de la méthode ELS utilisée pour résoudre le DARP. Dans cet algorithme, la phase de mutation se situe de la ligne 20 à la ligne 25. Les mutations utilisées sont au nombre de deux. Chacune à une probabilité d'être exécutée.

Le premier type de mutation génère un voisin assez proche de la solution père (S_{pere}). Elle consiste à sélectionner un client choisi aléatoirement parmi tous les clients, à le retirer de la tournée à laquelle il appartient avant de le réinsérer dans une autre tournée. Les mutations ne tiennent pas compte de la qualité de la solution générée, c'est pourquoi pour insérer le client dans une nouvelle tournée, la fonction *insertion_client* randomisée est utilisée. Cette fonction a été présentée dans la partie recherche locale et elle est appelée avec comme paramètre $mc = \infty$, qui représente le détour maximal infini. Dans l'algorithme 16 cette mutation est la $mutation\_1$, appelée la ligne 22.

La seconde mutation utilise un autre mouvement de la recherche locale, le *cut_and_paste*. Le coût de la solution finale n'est pas pris en compte. Cette deuxième mutation est plus coûteuse que la première. Dans l'algorithme 16, cette mutation est la $mutation\_2$, appelée ligne 24.

Ces deux mutations n'ont pas la même probabilité d'être appelées. En ligne 21 de l'algorithme 16, la mutation 1 est appelée dans 70% des cas et la mutation 2 la plus coûteuse, est appelée dans les 30 % restant.

2.6.3. La diversification

Un défaut des métaheuristiques est la possibilité d'explorer plusieurs fois les solutions d'une même région de l'espace des solutions. Ces explorations sont improductives. Dans un schéma de type ELS, la génération de plusieurs voisins de la solution courante avant de leur appliquer une recherche locale augmente ce risque, même si les différents mouvements de la recherche locale sont randomisés.

Un système de détection de solutions identiques est mis en place. Comparer des solutions de manière exacte étant coûteux, un système basé sur les techniques de hachage a été mis en place (Cormen et al., 1990).

Lorsqu'une solution est rencontrée, la valeur de hachage $Hash$ est calculée. Ce calcul utilise plusieurs éléments de la solution qui sont : le coût, les numéros des clients situés en première et dernière position ainsi que la date de début du service sur le sommet situé au milieu de la tournée. Le coût de la solution est pondéré avec un paramètre α suffisamment grand.

$$Hash(S) = \left(S.coût * \alpha + \sum_{k=1}^{m} \left(S.T[k].S[1] \times S.T[k].S[n_k] + S.T[k].B\left[\left[\frac{n_k}{2}\right]\right] \right) \right)$$

Cette valeur $Hash$ est modulée avec une valeur M suffisamment grande, par exemple $M = 999999$. M correspond au nombre de cases dans un vecteur nommé $Clone$. Chaque case du vecteur $Clone$ est initialisée à 0.

Chaque fois que la valeur de hachage d'une solution est rencontrée la case correspondante est incrémentée. Ainsi le nombre de fois qu'une solution a été visitée peut être obtenue en $O(1)$. Si la valeur d'une case est supérieure à une constante prédéfinie (généralement 1 ou 2), l'exploration de la solution est abandonnée. Dans ELS, la détection des clones est réalisée lors de la génération de chaque voisin. La mise à jour de la table $Clone$ se fait à la fin de chaque recherche locale

2.6.4. L'algorithme général de ELS

L'algorithme 16 donne une présentation globale du processus d'optimisation. Il se compose de deux parties. La partie 1 qui correspond aux lignes 3 à 9 permet de générer des solutions initiales et la partie 2 comprise entre les lignes 12-34 correspond à l'ELS. Il faut noter, ligne 14, les appels périodes à la fonction $learning\_process()$ qui met à jour les probabilités des différents mouvements de la recherche locale.

2.7. Les résultats

Les résultats présentés dans cette partie ont été réalisés sur des instances classiques de la littérature. Dans la première partie, les instances sont présentées ainsi que les différentes méthodes de la littérature. Puis les résultats pour les différents groupes d'instances sont détaillés. Enfin, une troisième partie regroupe les résultats pour chaque instance sous la forme de tableaux.

Algorithme 16 Evolutionary Local Search

Local variables

S_{ini} : best initial solution
S_{pere} : solution for the neighbors generation
S_{bv} : best solution after an iteration
S : current solution
S' : current solution after the local search

Global variables

n_{ini} : number of initial solutions
mi : maximum number of iteration
mn : maximum number of neighbors
lpc : learning process counter
mc : number for clone detection

Out parameter

sol_{best} : best solution found

Begin

$S_{ini}.coût := +\infty$;
$i := 1$;
repeat
  $S := initilization()$
  If ($S_{ini}.cost > S.cost$) **then**
    $S_{ini} := S$
  end if
  $i = i + 1$
until $(i \leq n_{init})$
$S_{Pere} := local\_search\ (S_{ini})$
$S_{best} := S_{pere}; i := 1$
while ($i > mi$) **do**
  if ($i\ mod\ lpc = 0$) **then**
    P = call learning_process (succ, fail)
  end if
  $S_{bv}.coût := +\infty$
  for $j := 1\ to\ mn$ **do**
    repeat
      repeat
        $Alea := rand()$
        If ($Alea > 30$) **then**
          S = **call** mutation_1 (S_{pere})
        else
          S = **call** mutation_2 (S_{pere})
        end if
      until $\left((Clone[Hash(S)]) > mc\right)$
      $S' = local\_search\ (S)$
    until $(Clone[Hash(S')]) > mc)$
    If ($S_{bv}.cost > S'.cost$) **then**
      $S_{bv} := S$
    end if
  end for
  $S_{pere} := S_{bv}; i = i + 1$
end while
return $\{S_{best}\}$

end

2.7.1. Les instances de la littérature du DARP

Comme présenté dans l'état de l'art, il existe de nombreuses définitions du DARP. Mais il n'existe que deux jeux d'instances reconnus dans la littérature, un premier proposé par (Ropke et al., 2007) dont les résultats optimaux sont connus et le deuxième proposé par (Cordeau and Laporte, 2003) où seules les valeurs des meilleures solutions trouvées sont connues (BKS). Les caractéristiques de ces instances sont détaillées dans le tableau 2.4. On constate que les instances les plus grandes sont proposées par (Cordeau and Laporte, 2003).

Tableau 2.4:
synthèse des caractéristiques pour les différents jeux d'instance du DARP

Caractéristiques :	**Instances proposées par (Cordeau and Laporte, 2003)**	**Instance proposées par (Ropke et al., 2007)**	**Instance proposées par (Ropke et al., 2007)**
n min / max	24/144	16/96	16/96
m min / max	*3/13*	2/8	2/8
q_i min / max	1/1	1/1	1/6
valeur de d	10	égale à la charge	égale à la charge
valeur de T_k	480	480/720	480/720
valeur de L_i	90	30	45

Le tableau 2.5, regroupe les différentes méthodes de la littérature pour résoudre le DARP. Ces méthodes sont au nombre de 6, mais les résultats de ces méthodes ne sont pas disponibles pour la totalité des instances. Par exemple la méthode proposée par (Masson et al., 2014) n'a été testée que sur les instances de (Cordeau and Laporte, 2003).

Tableau 2.5:
instances sur lesquels les différentes méthodes ont été testées

	(Cordeau and Laporte, 2003)	**(Ropke et al., 2007) groupe A**	**(Ropke et al., 2007) groupe B**
(Cordeau and Laporte, 2003)	X		
(Ropke et al., 2007)		X	X
(Parragh et al., 2010)	X		
(Parragh and Schmid, 2013)	X	X	X
(Masson et al., 2014)	X		
(Braekers et al., 2014)	X		X
ELS	X	X	X

Dans les tableaux de résultats, deux temps sont donnés : le temps publié par les méthodes dans les articles ainsi que les temps normalisés par rapport à la machine utilisée pour ce chapitre. Cette normalisation est basée sur le nombre de Mflop/s des différents processeurs utilisés. Pour trouver le nombre de Mflop/s, les travaux publiés de (Dongarra and Luszczek, 2011) ainsi que les travaux qu'ils ont réalisé sur le LINPACK publié dans (Dongarra et al., 1979) ont été utilisés.

Malheureusement ces données ne sont pas disponibles pour (Masson et al., 2014) et (Braekers et al., 2014). Le tableau 2.6 regroupe les noms des différents processeurs ainsi que le *speed factor* qui correspond au coefficient de multiplication pour normaliser le temps correspondant à chaque méthode.

Tableau 2.6:
normalisation des temps de caluls par rapport au nombre de MFLOP/S
Relative performances of computers (http://www.roylongbottom.org.uk/linpack%20results.htm)

	(Cordeau and Laporte, 2003)	(Ropke et al., 2007)	(Parragh et al., 2010)	(Parragh and Schmid, 2013)	(Masson et al., 2014)	(Braekers et al., 2014)	**ELS**
Processor	PIV 2 GHz	Opteron 2.4 GHz	Pentium D 3.2 GHz	Xeon 2.67 GHz	Core i3-530	2.6 GHz Core laptop	Core i7-3770 3.40 GHz
OS	-	Linux	-	-	Linux	-	Windows 7
Lang.	-	C++	C++	C++	C++	C++	C++
MFlops	533	1291	630	1144	?	?	2529
Speed factor	0.21	0.51	0.24	0.45	-	-	1

Les résultats fournis par la méthode ELS sont obtenus de manière à avoir un temps total d'exécution égal au temps normalisé utilisé par la méthode de (Parragh and Schmid, 2013). Par exemple, sur les instances proposées par (Cordeau and Laporte, 2003), le temps d'exécution moyen pour la méthode proposée par (Parragh and Schmid, 2013) est de 21.84 min. Le speed factor est d'environ 0.45 pour leur machine, soit un temps normalisé de 9.88 min.

2.7.2. Paramètres associés à la méthode de résolution.

Comme présenté dans les parties précédentes (2.6), la méthode de résolution nécessite de fixer un ensemble de paramètres. Ils ont une influence sur les résultats de la méthode de résolution. Parmi ces paramètres, certains sont fixes et déterminés par expérimentation sur un sous-ensemble réduit d'instances. D'autres sont dynamiques et leur valeur se met à jour automatiquement durant l'exécution, comme par exemple la probabilité d'exécution de chaque bloc à l'intérieur de la recherche locale.

Les paramètres fixes sont au nombre de 5. Le premier est le nombre de voisins générés à chaque itération de l'ELS. Il est fixé à 30 pour toutes les instances testées. Le second représente la probabilité d'exécution de chaque type de mutation utilisée. Comme expliqué dans la partie 2.7.2, deux types de mutation ont été développés. Lors de la génération d'une solution voisine, le premier type de mutation est exécuté dans 70% des cas et le second type dans les 30 % restants. Le troisième paramètre concerne la table de hachage. Lorsqu'une solution est évaluée, la clef correspondante est calculée et ajoutée à la table de hachage. La solution voisine générée est acceptée uniquement si sa clef n'a pas été ajoutée plus de 2 fois dans la table de hachage. Ceci évite ainsi de parcourir plusieurs fois le même espace de solutions.

Le quatrième paramètre correspond au nombre maximal d'itérations de la recherche locale, fixé à 1000 et au nombre d'itérations sans amélioration, limité lui à 10. Le dernier paramètre est le temps consommé dans l'ELS. L'ELS est autorisée à effectuer des itérations jusqu'à atteindre le temps de calcul de la méthode proposée par (Parragh and Schmid, 2013).

2.7.3. Étude comparative sur les instances proposées par (Cordeau and Laporte, 2003).

Les résultats illustrés dans le tableau 2.7 sont les résultats moyens sur toutes les instances de Cordeau et Laporte. Ces résultats sont obtenus après 5 exécutions. Il faut noter que les méthodes proposées par (Masson et al., 2014) et (Braekers et al., 2014) ne sont pas des

méthodes dédiées au DARP. Elles sont capables de traiter des problèmes plus généraux. La méthode proposée par (Masson et al., 2014) peut traiter le DARP avec des sommets de transfert et celle proposée par (Braekers et al., 2014) peut traiter le DARP avec une flotte de véhicules hétérogènes.

Les résultats de ELS sont inférieurs en temps et en écart moyen (colonne $\boldsymbol{gap}$) à toutes les méthodes sauf à celle de (Braekers et al., 2014). (Braekers et al., 2014) présente des résultats légèrement moins bons mais pour des temps non normalisés 7 fois plus petits. En l'absence de temps normalisé, il est difficile de comparer ces deux méthodes et de conclure sur leur efficacité relative.

Tableau 2.7:
résultats sur les instances de Cordeau et Laporte

méthodes	$\boldsymbol{gap}$	**Avg. Time**	**Scale Time**
(Cordeau and Laporte, 2003)	1.11	338.82	71.63
(Parragh et al., 2010)	1.58	133.30	33.24
(Parragh and Schmid, 2013)	1.58	21.84	9.88
(Masson et al., 2014)	1.15	40.12	-
(Braekers et al., 2014)	0.97	1.39	-
ELS	0.85	9.87	9.87

Les résultats détaillées, instance par instance, pour les 6 méthodes sont proposés sur le tableau 2.9. Ce tableau fait apparaître en couleur grise, les cas, où d'une part la méthode fournit des résultats en moyenne meilleurs (colonne $\boldsymbol{gap}$) et des résultats meilleurs en considérant la meilleur des réplications (colonne $\boldsymbol{gap_{best}}$.)

Le tableau 2.8 contient le nombre d'instances pour lesquelles le résultat fourni par une méthode est égal au meilleur résultat fourni par l'ensemble des méthodes. ELS fournit le meilleur résultat pour 12 des 20 instances de Cordeau et Laporte.

Tableau 2.8:
nombre de fois où la méthode est la meilleure sur les instances de Cordeau et Laporte

méthodes	Nb de meilleures solutions trouvées (colonne $\boldsymbol{gap}$)
(Cordeau and Laporte, 2003)	7
(Parragh et al., 2010)	3
(Parragh and Schmid, 2013)	2
(Masson et al., 2014)	2
(Braekers et al., 2014)	8
ELS	12

Tableau 2.9:
résultats sur les instances de (Cordeau & Laporte, 2003)

				(Cordeau and Laporte, 2003)			(Parragh et al., 2010)			(Parragh and Schmid, 2013)			(Masson et al., 2014)			(Braekers et al., 2014)			Our proposition			
				TS (1 run)			VNS (5 runs)			Hybrid LNS (5 runs)			ALNS (5 runs)			DA (5 runs)			ELS (5 runs)			
	m	n	BKS	$gap.$ (%)	gap_{best} (%)	T	$gap.$ (%)	gap_{best} (%)	T	$gap.$ (%)	gap_{best} (%)	T	$gap.$ (%)	gap_{best} (%)	T	$gap.$ (%)	gap_{best} (%)	T	$gap.$ (%)	gap_{best} (%)	T*	T
R1a	3	24	190.02	0.00	0.00	19.00	0.00	0.00	8.98	0.00	0.00	0.54	0.00	0.00	2.09	0.00	0.00	0.28	0.00	0.00	0.00	0.25
R2a	5	48	301.34	0.25	0.25	80.60	0.15	0.00	20.02	0.39	0.00	2.76	0.13	0.00	6.06	0.00	0.00	0.70	0.00	0.00	0.09	1.25
R3a	7	72	532.00	0.02	0.02	171.80	0.77	0.19	33.21	1.17	0.62	5.11	0.29	0.00	13.17	0.29	0.02	0.81	0.35	0.08	1.15	2.30
R4a	9	96	570.25	0.44	0.44	287.70	1.40	0.64	63.73	1.05	0.15	16.29	1.63	0.56	30.47	1.80	1.21	1.24	0.74	0.05	6.16	7.37
R5a	11	120	626.93	1.60	1.60	462.40	1.72	1.57	154.47	1.70	0.41	26.70	1.47	0.71	44.00	0.82	0.46	1.49	1.70	0.62	10.21	12.07
R6a	13	144	785.26	2.06	2.06	538.70	3.06	2.15	230.06	1.92	0.46	48.48	1.61	0.78	64.96	1.96	1.59	1.78	1.38	0.96	14.28	21.92
R7a	4	36	291.71	0.00	0.00	43.90	0.87	0.00	9.62	0.29	0.00	1.02	0.29	0.00	3.45	0.18	0.18	0.38	0.43	0.00	0.09	0.47
R8a	6	72	487.84	1.45	1.45	204.40	1.61	0.56	52.94	1.51	0.84	5.92	1.67	0.65	14.76	0.65	0.64	0.81	1.09	0.77	1.66	2.68
R9a	8	108	658.31	2.15	2.15	505.10	2.26	1.31	143.08	2.70	0.48	24.89	1.69	0.79	34.43	1.27	0.66	1.20	3.50	2.21	9.53	11.25
R10a	10	144	851.82	3.16	3.16	875.30	2.50	1.78	354.97	3.18	2.41	47.17	1.69	1.22	67.75	1.06	0.25	1.91	1.04	0.65	13.63	21.33
R1b	3	24	164.46	0.00	0.00	19.30	0.00	0.00	12.87	0.97	0.00	0.61	0.64	0.00	2.19	0.00	0.00	0.40	0.00	0.00	0.01	0.28
R2b	5	48	295.66	0.14	0.14	82.90	1.19	0.33	25.89	1.09	0.10	3.00	0.34	0.10	9.17	0.14	0.01	0.86	0.02	0.00	0.71	1.37
R3b	7	72	484.83	1.75	1.75	185.40	1.94	1.10	44.32	1.33	0.00	8.19	1.44	0.45	23.82	1.07	0.78	1.27	1.21	0.86	2.43	3.70
R4b	9	96	529.33	1.24	1.24	311.80	2.11	0.72	127.43	2.24	1.04	22.58	1.18	0.68	51.26	2.20	1.07	1.95	0.50	0.33	6.99	10.20
R5b	11	120	577.29	2.16	2.16	543.30	1.94	1.01	273.99	2.24	1.80	44.09	1.33	0.19	91.54	1.22	0.72	2.59	0.51	0.20	14.35	19.93
R6b	13	144	730.67	1.77	1.77	737.00	2.86	1.59	341.00	1.78	1.00	71.50	2.47	1.51	128.36	2.26	1.76	3.01	0.81	0.08	26.64	32.32
R7b	4	36	248.21	0.00	0.00	42.30	0.00	0.00	16.81	0.00	0.00	1.30	0.00	0.00	5.06	0.45	0.45	0.57	0.00	0.00	0.08	0.58
R8b	6	72	458.73	0.86	0.86	228.60	2.23	0.82	56.96	2.51	1.08	9.54	1.44	0.58	23.63	0.80	0.66	1.35	0.80	0.54	2.53	4.32
R9b	8	108	593.49	1.43	1.43	512.80	2.43	1.13	152.58	2.15	0.00	27.49	1.88	1.13	68.13	1.20	0.80	2.44	0.68	0.32	5.48	12.43
R10b	10	144	785.68	1.65	1.65	924.10	2.58	1.43	543.05	3.45	2.36	69.57	1.78	1.27	118.18	2.06	1.20	2.71	2.33	1.39	25.13	31.45
		Avg. Time				338.82			133.30			21.84			40.12			1.39			7.06	9.87
		Scale Time				71.63			33.24			9.88			/			/			7.06	9.87
			$\overline{gap}$	1.11			1.58			1.58			1.15			0.97			0.85			
			$\overline{gap}_{best}$		1.11			0.82			0.64			0.53			0.62			0.45		
		Nb. Best	$\overline{gap}$	7	4		3	5		2	10		2	7		8	5		12	10		

2.7.4. Étude comparative sur les instances de : (Ropke et al., 2007)(groupe A)

Les instances de (Ropke et al., 2007) peuvent être séparées en deux groupes pour simplifier la lecture des tableaux. Dans ce paragraphe, un premier groupe noté A est traité. Il est composé de la première moitié des instances dans lesquels la capacité des véhicules est limité avec $Q = 3$, et les demandes de transports concernent des individus seul, $\forall i$ un client $q_i = 1$. Seules les méthodes de (Ropke et al., 2007) et de (Parragh and Schmid, 2013) ont été appliquées dessus et les meilleurs résultats ont été produits par ces derniers.

Tableau 2.10:
résultats sur les instances de (Ropke et al. 2007)(groupe A)

méthodes	gap	Avg. Time	Scale Time
(Ropke et al., 2007)	0.00	63.30	13.38
(Parragh and Schmid, 2013)	0.17	2.26	1.02
ELS	0.07	1.02	1.02

Le tableau 2.10 montre les résultats moyens sur le groupe A de (Ropke et al., 2007). On constate que ELS fournit de meilleurs résultats que la méthode de (Parragh and Schmid, 2013) en temps et en écart. La méthode proposée par (Ropke et al., 2007), quant à elle, est une méthode exacte avec des temps de calcul 13 fois supérieurs.

Les résultats détaillés du tableau 2.11, mettent en évidence que l'écart (colonne **gap**) de 0.07% sur l'ensemble des instances est dû à 11 instances seulement et que pour les autres, la solution optimale est obtenue. Un commentaire similaire peut être fait pour la colonne $\boldsymbol{gab_{best}}$, qui possède une de $\boldsymbol{gab_{best}}$ <0.01 et où seules 3 instances (instances a7-84, a8-80, a8-96) ont une valeur $\boldsymbol{gab_{best}}$non nulle.

Tableau 2.11:
résultats sur les instances de (Ropke et al., 2007) groupe A

				(Ropke et al., 2007)		(Parragh and Schmid, 2013)			ELS			
				B&C		ALNS (5 runs)			ELS (5 runs)			
	m	n	Opt	$gap.$ (%)	T	$gap.$ (%)	gap_{best} (%)	T	$gap.$ (%)	gap_{best} (%)	T*	T
a2-16	2	16	294.25	0.00	0.01	0.00	0.00	0.12	0.00	0.00	0.00	0.05
a2-20	2	20	344.83	0.00	0.05	0.00	0.00	0.28	0.00	0.00	0.00	0.13
a2-24	2	24	431.12	0.00	0.12	0.00	0.00	0.35	0.00	0.00	0.00	0.15
a3-24	3	24	344.83	0.00	1.31	0.00	0.00	0.29	0.02	0.00	0.00	0.13
a3-30	3	30	494.85	0.00	15.48	0.08	0.00	0.50	0.25	0.00	0.05	0.23
a3-36	3	36	583.19	0.00	8.13	0.01	0.00	0.83	0.00	0.00	0.01	0.38
a4-32	4	32	485.50	0.00	0.12	0.04	0.00	0.55	0.00	0.00	0.01	0.25
a4-40	4	40	557.69	0.00	0.51	0.00	0.00	0.78	0.00	0.00	0.04	0.35
a4-48	4	48	668.82	0.00	1.01	0.03	0.00	1.62	0.00	0.00	0.19	0.73
a5-40	5	40	498.41	0.00	0.24	0.00	0.00	0.85	0.00	0.00	0.04	0.38
a5-50	5	50	686.62	0.00	2.67	0.04	0.00	1.60	0.01	0.00	0.32	0.72
a5-60	5	60	808.42	0.00	2.59	0.11	0.01	2.51	0.00	0.00	0.23	1.13
a6-48	6	48	604.12	0.00	0.90	0.07	0.00	1.14	0.00	0.00	0.08	0.52
a6-60	6	60	819.25	0.00	3.94	0.32	0.13	2.29	0.09	0.00	0.26	1.03
a6-72	6	72	916.05	0.00	8.16	0.41	0.07	4.43	0.10	0.00	1.11	2.00
a7-56	7	56	724.04	0.00	5.36	0.00	0.00	1.67	0.09	0.00	0.30	0.75
a7-70	7	70	889.12	0.00	6.13	0.49	0.05	2.88	0.08	0.00	0.67	1.30
a7-84	7	84	1033.37	0.00	29.43	0.68	0.27	7.04	0.18	0.02	1.51	3.18
a8-64	8	64	747.46	0.00	6.48	0.07	0.00	2.14	0.10	0.00	0.45	0.97
a8-80	8	80	945.73	0.00	26.14	0.60	0.31	5.73	0.14	0.01	2.02	2.58
a8-96	8	96	1229.70	0.00	1210.56	0.53	0.41	9.92	0.34	0.07	3.02	4.48
			Avg. Time		63.30			2.26			0.49	1.02
			Scale Time		13.38			1.02			0.49	1.02
			$\overline{gap}$	0.00		0.17			0.07			
			$\overline{gap}_{best}$	0.00			0.06			<0.01		
			Nb. Best $\overline{gap}$									

2.7.5. Étude comparative sur les instances de: (Ropke et al., 2007)groupe B)

Les solutions optimales des instances de (Ropke et al., 2007) du groupe B sont connues. Dans ce groupe la capacité des véhicules est plus importante que dans le groupe A avec $Q = 6$, de plus les demandes de transports concernent des groupes d'individus, $q_i \in [1; \dots; 6]$, $\forall i$ un client. Dans le tableau 2.12, on constate que ELS est en moyenne 10 fois plus rapide que la méthode exacte de (Ropke et al., 2007) qui a fourni les solutions optimales. Comparée aux autres méthodes, le gap moyen obtenu par ELS est le plus petit avec 0.05 % contre 0.12% pour (Braekers et al., 2014) et pour (Parragh and Schmid, 2013).

Tableau 2.12:
résultats sur les instances de(Ropke et al., 2007) (groupe B)

méthodes	gap	Avg. Time	Scale Time
(Ropke et al., 2007)	0.00	56.30	11.90
(Parragh and Schmid, 2013)	0.12	2.25	1.02
(Braekers et al., 2014)	0.12	0.51	/
ELS	0.05	1.02	1.02

Les résultats détaillés de chaque instance sont présentés sur le tableau 2.13, et mettent en évidence de très nombreuses instances pour lesquelles l'écart (colonne **gap**) est nul, indiquant que la solution optimale a été trouvée.

Tableau 2.13:
résultats sur les instances de (Ropke et al., 2007) groupe B

				(Ropke et al., 2007)		(Parragh and Schmid, 2013)			(Braekers et al., 2014)			ELS			
				B&C		ALNS (5 runs)			DA (5 runs)			ELS (5 runs)			
	m	n	Opt	$gap.$ (%)	T	$gap.$ (%)	gap_{best} (%)	T	$gap.$ (%)	gap_{best} (%)	T	$gap.$ (%)	gap_{best} (%)	T*	T
b2-16	2	16	309.41	0.00	0.04	0.00	0.00	0.15	0.00	0.00	0.21	0.00	0.00	0.00	0.07
b2-20	2	20	332.64	0.00	0.01	0.00	0.00	0.21	0.00	0.00	0.17	0.00	0.00	0.00	0.10
b2-24	2	24	444.71	0.00	0.06	0.03	0.00	0.40	0.03	0.00	0.28	0.00	0.00	0.00	0.18
b3-24	3	24	394.51	0.00	0.55	0.00	0.00	0.31	0.00	0.00	0.22	0.00	0.00	0.01	0.13
b3-30	3	30	531.44	0.00	3.20	0.00	0.00	0.48	0.00	0.00	0.30	0.00	0.00	0.01	0.22
b3-36	3	36	603.79	0.00	37.57	0.06	0.00	0.74	0.06	0.00	0.34	0.00	0.00	0.00	0.33
b4-32	4	32	494.82	0.00	75.42	0.00	0.00	0.43	0.00	0.00	0.24	0.00	0.00	0.00	0.20
b4-40	4	40	656.63	0.00	0.15	0.00	0.00	1.00	0.00	0.00	0.45	0.00	0.00	0.02	0.45
b4-48	4	48	673.81	0.00	0.63	0.17	0.00	1.73	0.17	0.00	0.58	0.16	0.06	0.04	0.78
b5-40	5	40	613.72	0.00	0.59	0.00	0.00	0.78	0.00	0.00	0.38	0.00	0.00	0.03	0.35
b5-50	5	50	761.40	0.00	0.92	0.09	0.00	1.49	0.09	0.00	0.49	0.04	0.00	0.25	0.67
b5-60	5	60	902.04	0.00	1.96	0.16	0.05	3.00	0.16	0.00	0.69	0.01	0.00	0.53	1.35
b6-48	6	48	714.83	0.00	0.27	0.00	0.00	1.07	0.00	0.00	0.42	0.01	0.00	0.21	0.48
b6-60	6	60	860.07	0.00	0.86	0.01	0.00	2.07	0.01	0.00	0.59	0.00	0.00	0.30	0.93
b6-72	6	72	978.47	0.00	127.71	0.18	0.12	4.43	0.18	0.00	0.86	0.03	0.01	1.37	2.00
b7-56	7	56	823.97	0.00	56.65	0.02	0.00	1.82	0.02	0.00	0.53	0.30	0.05	0.40	0.82
b7-70	7	70	912.62	0.00	4.66	0.26	0.05	3.70	0.26	0.00	0.66	0.00	0.00	0.59	1.67
b7-84	7	84	1203.37	0.00	11.51	0.85	0.66	6.72	0.85	0.04	0.90	0.18	0.00	1.64	3.03
b8-64	8	64	839.89	0.00	8.11	0.09	0.09	2.56	0.09	0.00	0.57	0.10	0.07	0.62	1.17
b8-80	8	80	1036.34	0.00	4.04	0.19	0.03	3.84	0.19	0.00	0.79	0.05	0.03	1.23	1.73
b8-96	8	96	1185.55	0.00	0.00	0.44	0.19	10.32	0.44	0.01	1.05	0.19	0.12	2.35	4.67
			Avg. Time		56.30			2.25			0.51			0.46	1.02
			Scale Time		11.90			1.02			/			0.46	1.02
			$\overline{gap}$	0.00		0.12			0.12			0.05			
			$\overline{gap}_{best}$				0.06			<0.01			0.02		

2.8. Conclusion

Une méthode de résolution efficace a été proposée pour résoudre le problème du DARP. Des résultats ont été obtenus sur les instances classiques de la littérature et ont été comparés avec les différentes méthodes publiées.

Les résultats obtenus avec la méthode ELS sont en moyenne meilleurs que les résultats de la littérature. L'écart moyen est de 0.85% pour des résultats obtenus après 10 min de temps de calcul sur les instances de (Cordeau and Laporte, 2003). Ces instances représentent l'un des jeux de données les plus difficiles à résoudre.

Pour obtenir de tels résultats, plusieurs points clés de la méthode ELS ont fait l'objet d'une attention particulière. Une fonction d'évaluation avec une complexité linéaire est utilisée. De plus une recherche locale qui adapte ses paramètres automatiquement au cours de la résolution a été proposée. Ce mécanisme d'apprentissage est basé sur les succès et les échecs des appels précédents de la recherche locale. Enfin une table de hachage a été introduite dans le schéma ELS pour diversifier au maximum les solutions considérées dans les phases de mutation et de recherche locale.

Développer cette méthode de résolution pour le DARP permet de mettre en évidence plusieurs points. Tout d'abord, il existe des liens entre les problèmes de transport et les problèmes d'ordonnancement. En particulier, l'évaluation d'une tournée repose sur un graphe disjonctif et utilise le même type d'algorithmes que ceux utilisés pour les problèmes d'ordonnancement. Mais les liens entre ces deux types de problèmes sont probablement plus importants et il conviendrait de les étudier davantage pour éventuellement identifier et porter des méthodes efficaces d'un problème à l'autre.

De plus, la mise en place de probabilité de sélection d'opérateurs évoluant au cours des itérations méritera une attention particulière dans le futur. Il a été montré que cette approche permet d'améliorer la convergence de la méthode vers les bonnes solutions. Ainsi, même si elle n'améliore pas les résultats, ceux-ci sont obtenus plus rapidement. S'il semble évident qu'une méthode adaptative possède un plus fort potentiel qu'une méthode classique, il reste encore à identifier les points sur lesquels opérer la boucle de rétroaction. Bien que cette problématique se soit déjà posée dans la littérature, aucune réponse satisfaisante n'a pour le moment été apportée et n'a été adoptée par la communauté.

Un autre point concerne enfin le calcul des dates de début de service sur les sommets clients. Comme mis en évidence dans la partie 2.3, ces dates déterminent les caractéristiques de la tournée. Or la valeur de ces dates dépend de la capacité à connaître le temps de trajet nécessaire au véhicule pour rejoindre les clients. Dans le DARP classique, ces temps de trajet sont supposés constants. Or ce n'est pas le cas dans un réseau routier car des fluctuations apparaissent au cours de la journée. Ces fluctuations peuvent remettre en question les caractéristiques des tournées mais aussi et surtout leur validité. Prendre en compte ces variations est l'axe de recherche exploré dans les deux chapitres suivants.

Références

Atallah, M.J., Chen, D.Z., Lee, D.T., 1995. An optimal algorithm for shortest paths on weighted interval and circular-arc graphs, with applications. Algorithmica 14, 429–441. doi:10.1007/BF01192049

Braekers, K., Caris, A., Janssens, G.K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transp. Res. Part B Methodol. 67, 166–186. doi:10.1016/j.trb.2014.05.007

Brinkmann, K., Neumann, K., 1996. Heuristic procedures for resource—constrained project scheduling with minimal and maximal time lags: the resource—levelling and minimum project—duration problems. J. Decis. Syst. 5, 129–155. doi:10.1080/12460125.1996.10511678

Caumond, A., Lacomme, P., Tchernev, N., 2008. A Memetic Algorithm for the Job-shop with Time-lags. Comput Oper Res 35, 2331–2356. doi:10.1016/j.cor.2006.11.007

Cheng, R., Gen, M., Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. Comput. Ind. Eng. 30, 983–997. doi:10.1016/0360-8352(96)00047-2

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. Part B Methodol. 37, 579–594. doi:10.1016/S0191-2615(02)00045-0

Cormen, T.H., Leiserson, C.E., Rivest, R.L., 1990. Algorithms. MIT Press, Cambridge, Massachusetts.

Dongarra, J.J., Bunch, J.R., Moler, C.B., Stewart, G.W., 1979. LINPACK Users' Guide. SIAM.

Dongarra, J., Luszczek, P., 2011. Linpack benchmark, in: Encyclopedia of Parallel Computing. Springer, pp. 1033–1036.

Firat, M., Woeginger, G.J., 2011. Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. Oper. Res. Lett. 39, 32–35. doi:10.1016/j.orl.2010.11.004

Gabow, H.N., Tarjan, R.E., 1985. A linear-time algorithm for a special case of disjoint set union. J. Comput. Syst. Sci. 30, 209–221. doi:10.1016/0022-0000(85)90014-5

Grefenstette, J., Gopal, R., Rosmaita, B., Van Gucht, D., 1985. Genetic algorithms for the traveling salesman problem, in: Proceedings of the First International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum, New Jersey (160-168), pp. 160–168.

Hunsaker, B., Savelsbergh, M., 2002. Efficient feasibility testing for dial-a-ride problems. Oper. Res. Lett. 30, 169–173. doi:10.1016/S0167-6377(02)00120-7

Jorgensen, R.M., Larsen, J., Bergvinsdottir, K.B., 2007. Solving the dial-a-ride problem using genetic algorithms. J. Oper. Res. Soc. 58, 1321–1331.

Lin, S., Kernighan, B.W., 1973. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Oper. Res. 21, 498–516. doi:10.1287/opre.21.2.498

Masson, R., Lehuédé, F., Péton, O., 2014. The Dial-A-Ride Problem with Transfers. Comput. Oper. Res. 41, 12–23. doi:10.1016/j.cor.2013.07.020

Parragh, S.N., Doerner, K.F., Hartl, R.F., 2010. Variable neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 37, 1129–1138. doi:10.1016/j.cor.2009.10.003

Parragh, S.N., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 40, 490–497. doi:10.1016/j.cor.2012.08.004

Potvin, J.-Y., Rousseau, J.-M., 1995. An Exchange Heuristic for Routeing Problems with Time Windows. J. Oper. Res. Soc. 46, 1433–1446. doi:10.1057/jors.1995.204

Prins, C., 2009. A GRASP × Evolutionary Local Search Hybrid for the Vehicle Routing Problem, in: Pereira, F.B., Tavares, J. (Eds.), Bio-Inspired Algorithms for the Vehicle Routing Problem, Studies in Computational Intelligence. Springer Berlin Heidelberg, pp. 35–53.

Ropke, S., Cordeau, J.-F., Laporte, G., 2007. Models and a branch-and-cut algorithm for pickup and delivery problems with time windows. Networks 49, 258–272.

Roy, B., Sussmann, B., 1964. Les problemes d'ordonnancement avec contraintes disjonctives. Note Ds 9.

Stein, D.M., 1978. Scheduling Dial-a-Ride Transportation Systems. Transp. Sci. 12, 232.

Tang, J., Kong, Y., Lau, H., Ip, A.W.H., 2010. A note on "Efficient feasibility testing for dial-a-ride problems." Oper. Res. Lett. 38, 405–407. doi:10.1016/j.orl.2010.05.002

Wolf, S., Merz, P., 2007. Evolutionary Local Search for the Super-Peer Selection Problem and the p-Hub Median Problem, in: Bartz-Beielstein, T., Aguilera, M.J.B., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (Eds.), Hybrid Metaheuristics, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–15.

CHAPITRE 3

Temps de trajet dépendant du temps

Objectifs du chapitre :

Dans la modélisation des problèmes de tournées de véhicules, les temps de trajet entre les sommets sont le plus souvent considérés comme déterministes. Or le temps de trajet entre deux positions dans un réseau routier n'est pas constant au cours de la journée. Il varie en fonction de nombreux facteurs comme, par exemple, le nombre d'usagers sur la route ou encore les conditions météorologiques. L'objectif de ce chapitre est d'intégrer des temps de trajet qui varient selon la date de départ du véhicule afin de modéliser par exemple les phénomènes d'embouteillage qui se produisent dans un réseau routier. L'hypothèse est que la congestion due à ces d'embouteillages peut être estimée au cours de la journée et donc être intégrée à la modélisation. Le problème traité reste dans un contexte statique où les données sont connues initialement. La possibilité d'attendre sur les sommets pour éviter les pics de congestion est également prise en compte.

Une approche de résolution en deux phases de ce problème est proposée. Elle consiste, premièrement, à résoudre dans une approche monocritère le problème où le véhicule n'est pas autorisé à attendre sur les sommets. Puis, dans la seconde phase, une approche de résolution est proposée pour intégrer les temps d'attente. Les solutions obtenues dans la première phase sont utilisées pour construire une population initiale de la seconde phase. Cette seconde phase traite le problème de manière bi-critère, les critères étant le temps de conduite des chauffeurs et la durée totale des tournées.

3.1. Présentation du problème

3.1.1. Contexte de l'étude

Le problème considéré dans cette partie n'est pas un problème de transport à la demande. C'est un problème de tournées de véhicules moins contraint, pour lequel on va considérer d'autres types de contraintes. L'objectif est de développer une méthode de résolution pour un problème de transport dans lequel des temps de trajet dépendent du temps. Les méthodes développées pourront alors être adaptées pour résoudre le problème du transport à la demande dépendant du temps.

Une autre spécificité du problème considéré, concerne le fait d'autoriser le véhicule à prendre des pauses durant sa tournée. Le modèle fait la distinction entre le temps de conduite du chauffeur et son temps de travail. Ces pauses sont obligatoires dans la législation du temps de travail. Elles peuvent être positionnées pour permettre, dans une certaine mesure, d'éviter à un camion de s'engager dans un axe avec une forte densité de trafic. En cas de congestion, le camion parcourt une faible distance pendant un laps de temps important. Cette situation a des conséquences sur le temps de conduite et sur le temps de travail et doit être évité le plus possible.

En gérant les temps de pause au cours de la journée, il est possible de contrôler les dates de passage (arrivée) sur les arcs du réseau routier et ainsi de chercher des solutions avec un temps de conduite plus faible mais un temps de travail plus grand, la différence reposant dans les temps d'attente. Comme les temps de conduite sont sensiblement plus petits que les temps de travail, ceci à un sens pratique important.

Il faut mettre ce type de recherche dans le contexte global d'une compagnie de transport. Planifier des transports sans prendre en compte les variations de temps de trajet peut avoir des conséquences sur le planning des conducteurs. Une problématique est d'éviter de se retrouver les soirs et veilles de weekend avec des conducteurs et camions (éventuellement chargés) dans l'incapacité de rejoindre les clients ou le dépôt pour cause de contrainte de temps de conduite. Cette situation est très dommageable pour la société de transport qui se retrouve alors devant un choix difficile : soit payer des indemnités d'éloignement de son domicile au chauffeur, soit envoyer, lorsque c'est possible, un deuxième chauffeur ramener le camion. Le deuxième cas se heurte aussi à la législation en vigueur et en particulier aux interdictions de circuler le weekend et jours fériés.

Dans le cas de transport longue distance, les phénomènes transitoires qui ralentissent le trafic routier (pensons aux congestions sur le périphérique par exemple) peuvent avoir une faible durée par rapport à la durée d'une tournée. Il est évident que la réduction du temps de conduite par l'ajout de pause supplémentaire dans la tournée permet d'éviter ou réduire le cas dommageable d'un camion dans un bouchon.

3.1.2. Problème de tournées de véhicules (VRP – Vehicle Routing Problems)

Le VRP est une extension du problème du voyageur de commerce (TSP) dans lequel une flotte homogène de véhicules est considérée. La flotte doit livrer un ensemble de clients modélisés par les sommets d'un graphe. C'est un problème de type $One\text{-}to\text{-}Many$ où un seul type de produit est considéré. Chaque sommet (client) du graphe doit être livré d'une certaine quantité de produits. Ce problème est aussi appelé problème de tournées de véhicules avec capacité (CVRP – *Capacitated Vehicle Routing Problem*), car un véhicule ne peut transporter qu'une quantité limitée de produit. Dans ce chapitre, la notation VRP est utilisée.

Pour les instances de tailles assez réduites, il existe dans la littérature des algorithmes exacts de résolutions traitant jusqu'à une centaine de clients (Baldacci et al., 2007). Pour les instances de plus grande taille, les méthodes développées sont de types heuristique et métaheuristique. Un état de l'art de ces différentes méthodes est disponible dans le livre de (Toth and Vigo, 2001).

3.1.3. Problèmes de tournées dépendant du temps (modèle étudié)

Les problèmes de tournées de véhicules avec temps de trajet dépendant du temps sont encore assez peu étudiés dans la littérature. Pourtant, en zone urbaine, le temps de trajet des véhicules entre deux points du réseau routier n'est pas constant et il dépend de multiples facteurs. Certains facteurs sont connus comme la distance qui sépare ces deux points ou la vitesse maximale autorisée pour le véhicule. En revanche, d'autres facteurs sont moins prévisibles, par exemple :

- les accidents ;
- les conditions météorologiques ;
- les travaux ;
- le nombre d'usagers sur la route ;
- les conditions de circulation aux intersections ;
- etc…

Parmi cette seconde catégorie de phènomènes, il n'est pas possible de connaître l'impact exact qu'ils ont sur le temps de trajet. Par contre, avec suffisament de données sur les conditions de circulations passées, il est possible de les estimer assez précisément. Ces données peuvent être disponibles grâce au développement des nouvelles technologies, comme les appareils de guidage GPS et la connection 3G. Il y a donc de plus en plus de capacité d'information sur les temps de trajet en temps réel ainsi que sur les évolutions au cours de la journée, de la semaine et du mois. Il est donc possible d'obtenir des informations plus précises que la simple vitesse moyenne ou le temps de trajet moyen entre deux clients. Comme l'illustre la figure 3.1, il est possible d'obtenir des vitesses moyennes en fonction du jour de la semaine et de l'heure avec le service Google Maps par exemple. Les couleurs des arcs representent les conditions de circulations habituelles dans la ville de Clermont-Ferrand à une certaine date.

Pour l'instant, dans les problèmes de tournée de véhicules de la littérature, la vitesse des véhicules est souvent modélisée par une constante spécifique à chaque arc du graphe. Le temps de trajet est alors calculé en divisant la distance entre les deux clients par cette vitesse moyenne. Une telle modélisation est de nature à engendrer des retards dans les livraisons des clients parce qu'elle ne prend pas en compte la variabilité. Elle peut donc conduire à une impossibilité pour le chauffeur de réaliser sa tournée lorsqu'elle est mise en application dans un réseau routier.

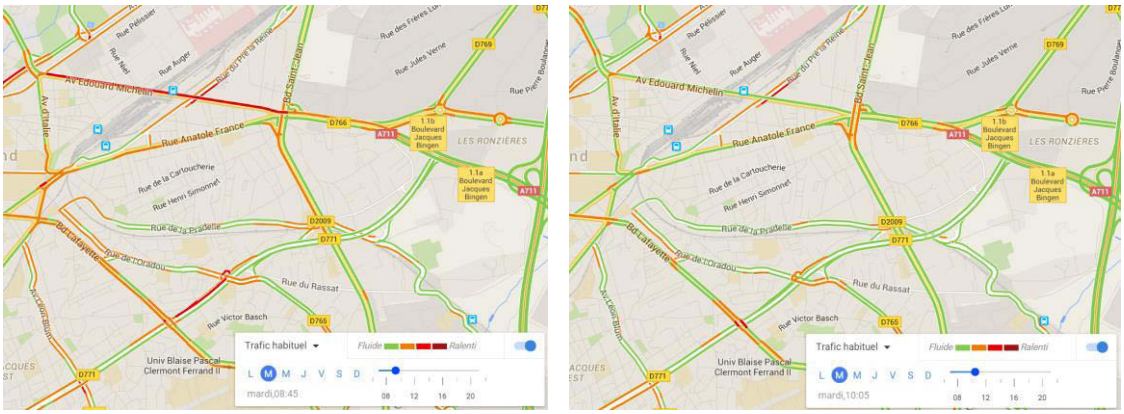


Figure 3.1 : Evolution du trafic à Clermont-Ferrand (63) entre 8h45 et 10h05 un jour de semaine. Cartes obtenues avec Google Maps

Le problème considéré dans ce chapitre est un VRP dans lequel la journée est découpée en un ensemble fixe de périodes. Les temps de trajet dans le réseau routier évoluent de manière significative suivant la période. Les vitesses moyennes du véhicule durant les différentes périodes sont supposées connues et permettent d'obtenir les temps de trajet. Ces vitesses sont différentes pour chacun des arcs du réseau considéré. Le problème est statique dans le sens où toutes les vitesses sont supposées connues avant l'optimisation. Les phénomènes ponctuels comme les accidents ou les routes en travaux, qui ont des conséquences certaines sur le temps de trajet, ne sont pas pris en compte explicitement. Les solutions obtenues doivent permettre de servir les clients en tenant compte de l'évolution des temps de trajet. Dans la littérature certains articles traitent des problèmes de tournées de véhicules dépendant du temps (TDVRP), comme par exemple (Malandraki and Daskin, 1992). Plus récemment (Wen and Eglese, 2015) traitent un problème dans lequel les temps de trajet varient pour modéliser les embouteillages.

Une autre particularité du modèle proposé est la prise en compte de pauses pour les véhicules. La pause est un élément obligatoire dans la législation sur les temps de conduite des chauffeurs. Non seulement elle peut être considérée dans un problème de VRP mais elle devrait normalement être prise en compte dans la conception de solution. Les législations

varient d'un secteur à un autre (conducteur de camion, chauffeur de bus, …) et sont soumises à beaucoup d'exceptions. Des travaux ont été menés sur les temps de pause dans les problèmes de tournées de véhicules. Des explications détaillées ainsi que des exemples ont été proposés dans la thèse de Vidal (Vidal, 2013).

La nouveauté dans ce modèle est que, pour pouvoir profiter de l'évolution constante du réseau, il est possible d'inclure des pauses durant la tournée. Ces pauses permettent notamment au véhicule d'attendre que l'état du réseau évolue avant de repartir pour livrer un nouveau client. Ce type de réflexion est couramment mené par les automobilistes en région parisienne à l'approche du périphérique. Ainsi à 7h30 du matin, on peut se trouver devant la problématique suivante :

- emprunter le périphérique et courir le risque de rester bloqué à 10km/h pendant 1h
- attendre sur une aire de repos de 7h30 à 8h30 pour emprunter le périphérique à partir de 8h30, après l'heure de pointe et ne passer alors que 30 min pour traverser Paris en empruntant le périphérique.

Ce qui n'est pour un usager qu'un simple problème d'optimisation lié à son confort de conduite devient un critère économique pour une entreprise. Le chauffeur passant une heure sur le périphérique consomme une partie de son temps de conduite de la journée alors que celui attendant 1h sur une aire de repos ne fait qu'utiliser un temps de pause décompté de son temps de travail.

Une tournée peut donc être découpée en trois éléments différents : le temps de conduite, le temps de livraison et le temps de pause. L'introduction de pauses dans le modèle peut permettre d'améliorer la qualité des solutions proposées sur plusieurs points. Premièrement, du point de vue du chauffeur, l'ajout de pauses durant la tournée améliore ses conditions de travail. De plus, en positionnant les pauses à des moments stratégiques, elles peuvent permettre d'attendre des périodes plus propices pour se déplacer. Le conducteur passe ainsi moins de temps dans les embouteillages, qui sont les périodes les plus éprouvantes. Deuxièmement, du point de vue de l'entreprise, le prix d'un véhicule en pause n'est pas le même que le prix du véhicule en mouvement. Il peut être préférable d'accorder une pause d'une heure à un chauffeur si cela lui permet de passer 30 minutes sur la route plutôt que de rester bloqué une heure dans un embouteillage à l'entrée de la ville.

La solution recherchée ne cherche donc pas à minimiser la distance parcourue par la flotte comme pour le VRP. Elle vise plutôt à minimiser à la fois le temps de conduite (rt) de la flotte et la durée totale des tournées (td). Ce problème peut donc être vu comme un problème bi-critère (rt / td).

Il faut noter que le fait que les temps de transport évoluent différemment les uns des autres implique que le plus court chemin entre deux points du graphe (réseau routier) évolue au cours de la journée. Par conséquent, deux solutions avec des dates de départ différentes sur le dépôt peuvent être amenées à traiter les clients dans des ordres très différents. Pour mettre ces différents points en évidence, il n'est pas possible de travailler uniquement sur le graphe complet composé des plus courts chemins entre les différents clients comme cela est souvent fait pour le VRP. Le problème et les solutions sont exprimés sur un graphe "routier". Ce graphe se compose d'arcs orientés, des sommets clients et également des sommets intermédiaires qui constituent les points de passage possibles pour que les véhicules puissent passer d'un client à un autre.

Dans le modèle, des simplifications ont été effectuées. Les législations ne sont pas prises en compte explicitement et les pauses introduites doivent uniquement respecter une durée minimale afin que le chauffeur ait un temps suffisant pour se garer et pour se reposer avant de

repartir. Afin d'obtenir des solutions réalistes, lorsque la date de départ sur un sommet est choisie, l'heure de départ est arrondie au quart d'heure supérieur. Une telle approche basée sur des quarts d'heure, permet d'obtenir des solutions réalistes susceptibles d'être appliquées par un chauffeur. On ne peut pas imaginer fixer une heure de départ "fractionnaire" telle que 17h43 ou 17h31 à un chauffeur. En tout état de cause, elle serait arrondie de la propre initiative du chauffeur. Il semble aussi raisonnable de fixer l'heure de départ d'une tournée selon des critères "raisonnables" permettant, par exemple, d'éviter que les tournées s'effectuent de nuit.

3.2. Définitions formelles et notations

3.2.1. Définition et notations du VRP

Le VRP est modélisé par un graphe non-orienté $G_{VRP} = (V, E)$ avec $V = \{1, \dots, n\}$ l'ensemble des sommets associés aux clients et $E = \{(i,j) |\ i,j \in V\}$ l'ensemble des arrêtes. A l'ensemble de sommets V est ajouté un sommet 0 qui représente le dépôt sur lequel la flotte de véhicule est positionnée initialement. Un ensemble de notations est introduit concernant les clients et les véhicules :

- n : nombre de clients ;
- m : nombre de véhicules ;
- $\mathrm{lt_i}$: temps de service nécessaire pour livrer le client i ;
- TD : durée totale maximale d'une tournée ;
- $\mathrm{q_i}$: demande du sommet – client i, cette charge est positive (sauf pour le dépôt $q_0 = 0$);
- Q_k : charge maximale du véhicule k, comme le problème présenté ici est homogène, on note $Q = Q_k \; \forall \; k \in \{1, \dots, m\}$;
- $\mathrm{c_{ij}}$: coût nécessaire pour aller du sommet i au sommet j, *i.e.* coût de l'arrête (i,j) ;
- t_{ij} : temps de trajet sur l'arrête (i,j).

Le VRP est un problème *One-to-Many* avec un seul type de produit et les requêtes sont uniquement composées des sommets de livraison avec $q_i \geq 0, i = 1 \dots n$. De plus le coût d'utilisation d'un l'arrête (i,j) est fixe et vaut c_{ij} quel que soit le véhicule. La figure 3.2 donne un exemple d'une solution d'un VRP à 8 clients et deux véhicules.

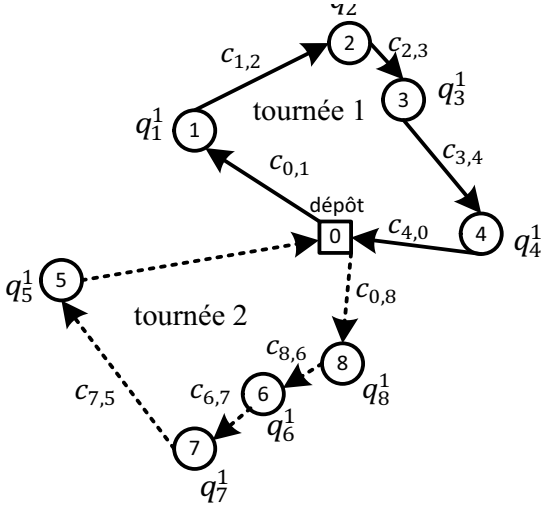


Figure 3.2 : Une solution associée à un problème de VRP

Les contraintes du problème sont :

- tous les clients reçoivent la quantité demandée ;
- la capacité maximale du véhicule est respectée en tout point de sa tournée ;
- le nombre de tournées est inférieur ou égal à m (nombre de véhicules disponibles) ;
- la durée totale des tournées est inférieure à TD_{max}.

Généralement le VRP est résolu dans un graphe complet non orienté et utilise la distance euclidienne entre les sommets comme coût. Le temps de trajet sur une arrête est constant et sa valeur est le plus souvent égale ou proportionnelle à la distance euclidienne. La fonction objectif associée au VRP consiste à minimiser le coût, $i.e.$ la somme des coûts des arrêtes empruntées par les véhicules. Enfin, les dates de passage sur les sommets n'ont pas d'importance puisqu'ils n'ont pas d'influence sur le coût de la solution et qu'il n'y a pas de fenêtre de temps.

3.2.2. Définition et notations du TDVRP

Le TDVRP se modélise par un graphe orienté qui représente par exemple un réseau routier. Ce graphe est noté $G_{VRPTD} = (V_2, E)$ avec $V_2 = V \cup V'$ l'union des sommets $V = \{1, \dots, n\}$ représentant les clients et des sommets intermédiaires $V' = \{x_1, \dots, x_{n'}\}$. Ces derniers permettent aux véhicules d'aller d'un sommet à un autre sur le réseau routier. Comme pour le VRP, à cet ensemble de sommets V est ajouté un sommet 0 qui représente le dépôt où la flotte de véhicules est positionnée initialement. Ces différents sommets apparaissent dans la figure 3.3.

L'ensemble $E = \{(i,j) |\, i,j \in V_2\}$ est l'ensemble des arcs. A chaque arc est associé une longueur et un ensemble de vitesses. L'ensemble des vitesses correspond aux vitesses moyennes du véhicule sur l'arc pour différentes périodes de temps qui composent la journée. Contrairement au VRP, le coût d'un arc correspond à un temps de trajet et dépend de la date de départ d. Il est notée t_{ij}^{d}. Le problème est considéré sur un horizon de 24 heures avec une granularité de l'ordre de la minute pour les temps de trajet entre les sommets. De plus la journée est découpée en périodes de durée identique. De nouvelles notations sont introduites :

- K : nombre de périodes ;
- k : $k^{ème}$ période de la journée ;
- n': nombre de sommets intermédiaires ;
- s_{ij}^{p} : vitesse du véhicule sur l'arc (i,j) associée à la période p ;
- d_{ij} : longueur de l'arc (i,j) ;
- W_{min} : durée minimale de la pause ;
- nb_start : nombre de début de tournées possibles pour le chauffeur ;
- $start_i$: $i^{\text{ème}}$ date de début possible de journée sur dépôt.

La figure 3.3 est un exemple de graphe associé au TDVRP dans lequel le dépôt est représenté par un carré numéroté 0 ; les sommets - clients sont représentés par des cercles numérotés 1, 2 et 3 ; les sommets intermédiaires sont symbolisés par des carrés noirs. Dans ce graphe les arcs sont orientés et contrairement au VRP aucun coût ne leur est associé. En effet le calcul du coût dépend de la date de départ du véhicule.

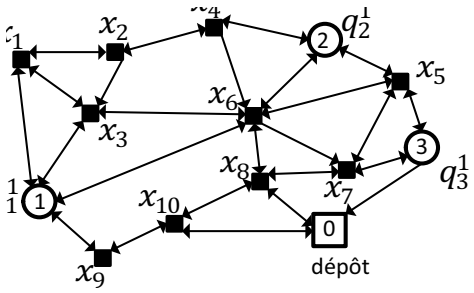


Figure 3.3 : Exemple de graphe G_{VRPTD}

Les contraintes du VRP s'appliquent au TDVRP et deux contraintes supplémentaires concernant les pauses sont ajoutées au problème. Il s'agit de prendre en compte les éléments suivants :

- les pauses sur un sommet sont d'une durée supérieure à W_{min} ;
- après une pause, le départ du véhicule s'effectue à une date arrondie au quart d'heure supérieur (xh00, xh15, xh30, xh45).

3.2.3. Définition d'une solution du TDVRP

Pour le VRP, une solution s se compose d'un ensemble de tournées. A chaque solution est associée un coût $c(s)$ égal à la somme des coûts de chaque tournée : $c(s) = \sum_{t=1}^{m} c(t)$ avec $c(t)$ le coût d'une tournée. Une tournée se caractérise, pour le VRP, par une suite ordonnée de sommets qui commence et finit au dépôt 0, suite ordonnée qui est notée λ. $\lambda(i)$ correspond au sommet en position i dans la tournée. Le coût d'une tournée est égal à la somme des coûts des arcs empruntés par le véhicule : $c(t) = \sum_{i=1}^{|\lambda|-1} c_{\lambda(i),\lambda(i+1)}$, avec $|\lambda|$ le nombre de sommets dans λ. De plus, à une tournée est associée une charge transportée : $q(\lambda) = \sum_{i=1}^{|\lambda|} q_{\lambda(i)}$. La figure 3.4. donne une représentation d'une solution obtenue à partir de $\lambda = \{0,1,2,3,4,0\}$.

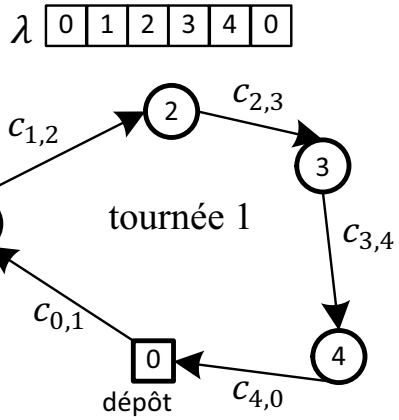


Figure 3.4 : Suite λ associée à 4 clients et la tournée correspondante pour le VRP

Pour le TDVRP, une solution est aussi composée d'un ensemble de tournées mais, par contre, ces tournées ne sont pas uniquement caractérisées par une suite λ comme pour le VRP. Pour illustrer cela, un exemple simple est proposé sur la figure 3.5. Le graphe G_{VRPTD} est composé uniquement d'un client et d'un sommet intermédiaire et la tournée est imposée par le sens des arcs. Pour cet exemple, supposons que deux dates de départ sont possibles : soit le véhicule commence la journée à 8h00 soit il commence à 10h00. Donc $nb\_start = 2$ et $start = \{8h00, 10h00\}$.

Pour qu'une solution soit possible, la charge q_1 doit être inférieure à la charge maximale transportable par le véhicule. A chaque arc sont affectés une distance et un ensemble de K vitesses, avec dans l'exemple $K = 5$. Ceci signifie qu'on considère que la journée se compose de trois périodes. Dans le modèle, pour simplifier, t_{ij}^{d} est supposé égal à la distance divisée par la vitesse du véhicule pour la période k associée à la date de départ d.

arc E:	**dépôt → x_1**	**x_1 → 1**	**1 → dépôt**
distance d_{xy} :	25km	25km	40km

	vitesses s_{xy}^{k} :		
périodes k	**dépôt → x_1**	**x_1 → 1**	**1 → dépôt**
P_1[8h ; 9h]	50km/h	25km/h	30km/h
P_2[9h ; 10h]	50km/h	40km/h	30km/h
P_3[10h ; 11h]	25km/h	50km/h	40km/h
P_4[11h ; 12h]	25km/h	25km/h	40km/h
P_5[12h ; 13h]	25km/h	30km/h	40km/h

G_{VRPTD} :

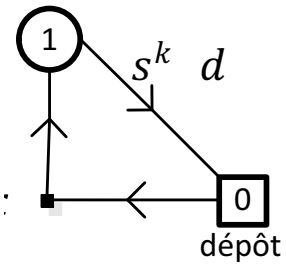


Figure 3.5 : Un exemple de TDVRP

Trois solutions sont illustrées dans la figure 3.6. La légende est la suivante : $start$ est la date de début de travail du chauffeur ; a est la date d'arrivée sur un sommet ; d est la date de départ du sommet ; t est le temps de trajet sur l'arc associé ; c est la durée du chargement sur le sommet; p est la durée de la pause sur le sommet.

La solution figure 3.6 (a) correspond au cas où le véhicule n'attend jamais avec un départ du dépôt à 8h00. A l'heure du départ, la vitesse sur l'arc du dépôt vers le sommet x_1 est 50km/h et $d_{0x_1} = 25$ km. Le véhicule arrive donc sur le sommet x_1 à 8h30 et repart vers le sommet suivant. Il arrive sur le sommet client 1 à 9h30 et il décharge le véhicule pendant 30 minutes avant de repartir. La tournée se termine à 11h35 et le chauffeur n'a pas effectué de pause. Il a donc conduit pendant 3h05.

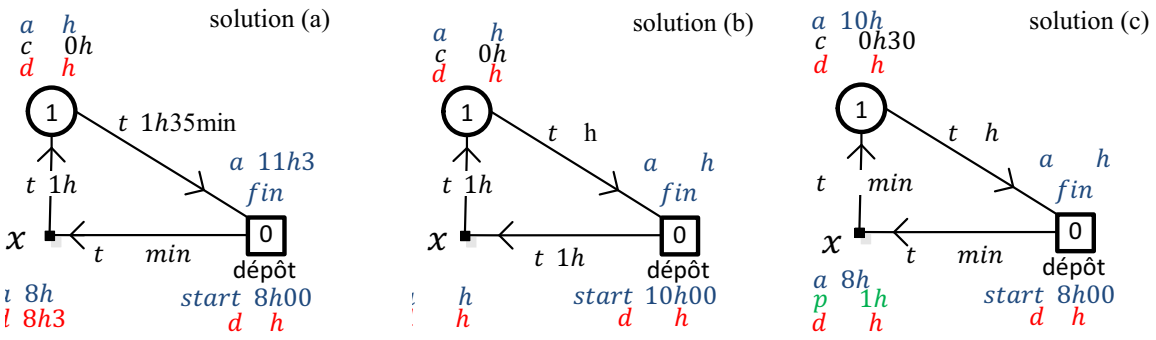


Figure 3.6 : Trois solutions associées à l'exemple

La solution figure 3.6 (b) correspond à un autre cas où le véhicule n'attend pas. Mais cette fois, le départ du dépôt a lieu à 10h00. La longueur des arcs est toujours la même, par contre les vitesses ont changé. La vitesse sur le premier arc est passée de 50km/h à 25km/h et la vitesse sur l'arc du client A au dépôt est passée de $30km/h$ (à 8h) à 40km/h pour toutes les

périodes après 10h. La tournée se termine alors à 13h30. Comme la tournée précédente, le chauffeur n'a pas effectué de pause et il a mis 30 minutes pour décharger le véhicule sur le client 1. Il a donc conduit pendant 3h.

La dernière solution, illustrée figure 3.6 (c), est une solution dans laquelle de l'attente est introduite. La tournée commence comme pour la solution (a) avec un départ du dépôt à 8h00. Le véhicule arrive sur le sommet x_1 à 8h30 mais cette fois le chauffeur effectue une pause avant de repartir. La pause dure 1h30 afin que le départ soit réalisé durant la période P_3 où la vitesse sur l'arc suivant est la plus intéressante (50km/h). Le véhicule arrive donc à 10h30 sur le sommet client où il charge la cargaison pendant 30 minutes et repart. Cette tournée se termine à 12h00. La durée totale de travail du chauffeur est de 4h mais le chauffeur conduit seulement 2h.

On constate que, bien que les tournées soient identiques sur l'ordre de visite des clients, elles sont différentes pour le problème du TDVRP. Pour pouvoir caractériser totalement une tournée du TDVRP, il faut donc prendre en compte plusieurs éléments :

- les chemins entre les clients ;
- les dates de départ de chaque sommet.

1. Les chemins entre les clients

Une première distinction avec le VRP est que les plus courts chemins entre les clients ne sont pas figés dans le temps. A cause des variations de vitesses associées aux arcs, les plus courts chemins en temps varient pendant la journée.

Un nouvel exemple présenté sur la figure 3.7 permet de mettre en évidence ces changements. Dans cet exemple seuls deux chemins sont possibles pour aller du sommet 0 (le dépôt) au client 1 : soit le véhicule utilise l'arc (0,1), soit il passe par le sommet intermédiaire x_1 en utilisant successivement les arcs $(0, x_1)$ puis $(x_1, 1)$.

On suppose seulement deux horaires de départ $start = \{8h, 9h\}$. Si le véhicule part à 8h, alors le plus court chemin consiste à passer par x_1 avec $t_{0x_1}^{8h}$ = 30 min auquel s'ajoute $t_{x_1 1}^{8h30} = 30$ min, soit 60 min contre t_{01}^{8h} = 80 min. Par contre, si le véhicule part à 9h, alors il est préférable d'emprunter l'arc (0,1). En effet t_{01}^{9h} = 60 min puisque la vitesse sur l'arc en deuxième période a augmenté, que $t_{0x_1}^{9h}$ = 30 min est identique et que $t_{x_1 1}^{9h30} = 1$ h, ce qui donne un temps total de trajet 1h30 pour le chemin passant par x_1.

arc E:	**dépôt → x_1**	**x_1 → 1**	**dépôt → 1**
distance d_{xy} :	25km	25km	40km
	vitesses s_{xy}^k :		
périodes k	**dépôt → x_1**	**x_1 → 1**	**1 → dépôt**
P_1[8h ; 9h]	50km/h	50km/h	30km/h
P_2[9h ; 10h]	50km/h	25km/h	40km/h

G_{VRPTD} :

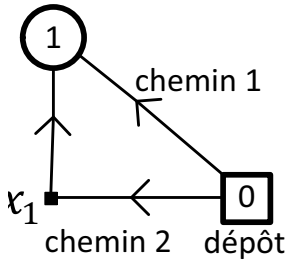


Figure 3.7 : Un exemple de pour le calcul d'un chemin

Il existe donc plusieurs chemins entre les clients. La suite λ' associée à un problème de TDVRP doit contenir l'ordre de passage du véhicule sur les sommets clients mais aussi l'ordre de passage sur les sommets intermédiaires pour rejoindre les clients. Un exemple de tournée

associée à une suite λ' est illustré dans la figure 3.8.

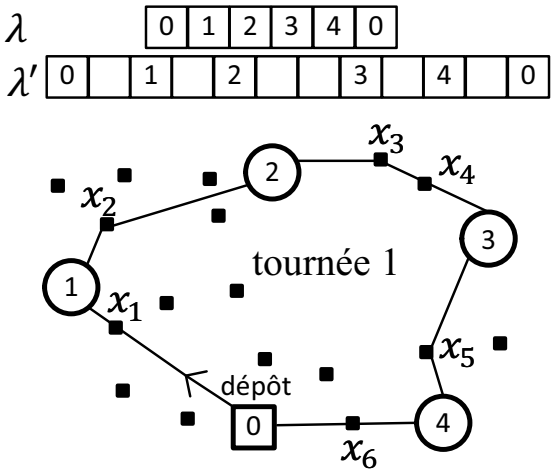


Figure 3.8 : Suite λ' associé à 4 clients et tournée correspondante pour le TDVRP

2. Les dates de départ des sommets

Comme les temps de trajet entre les sommets varient et que le véhicule est autorisé à attendre sur les sommets, il s'agit de définir si le véhicule attend sur chaque sommet (et combien de temps) ou s'il repart immédiatement. En plus de la suite λ, une tournée est donc caractérisée par une suite T qui contient les dates de départ du véhicule sur les sommets. Les dates de départ suffisent pour déterminer toutes les autres variables de la tournée comme les temps d'attente et les dates d'arrivée. Les temps de trajet peuvent être calculés si les dates de départ sont connues ainsi que le temps de chargement lt_i sur un sommet client i.

La figure 3.9 illustre un chemin emprunté par le véhicule pour aller du client 2 au client 3 : le véhicule passe par les sommet x_3 et x_4 pour aller du client 2 au client 3. Le vecteur T associé contient donc 4 dates qui correspondent aux départs du véhicule de chaque sommet du chemin qui sont 8h pour le sommet 2, 8h10 pour le sommet x_3, etc… x_5 x_4

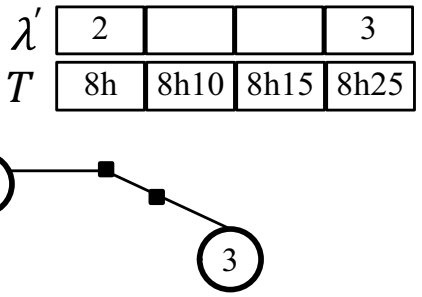


Figure 3.9 : Stockage des dates de départ sur les sommets

Dans la définition du problème, il est précisé que le véhicule doit commencer sa journée à une date précise qui correspond à une des valeurs de $start$. Ces dates sont réparties de manière régulière sur toute la journée comme illustré dans la figure 3.10. Elles correspondent au moment à partir duquel le temps d'attente est comptabilisé. Cette date peut être différente de la date de départ du véhicule mais elle n'est pas nécessaire pour définir la tournée puisque si T(0) la date de départ du dépôt est connue alors le début de la tournée correspond à $\{\max(x) \mid x \in start, x < T(0)\}$. Cette date est notée $start(i)$ dans la suite du chapitre.

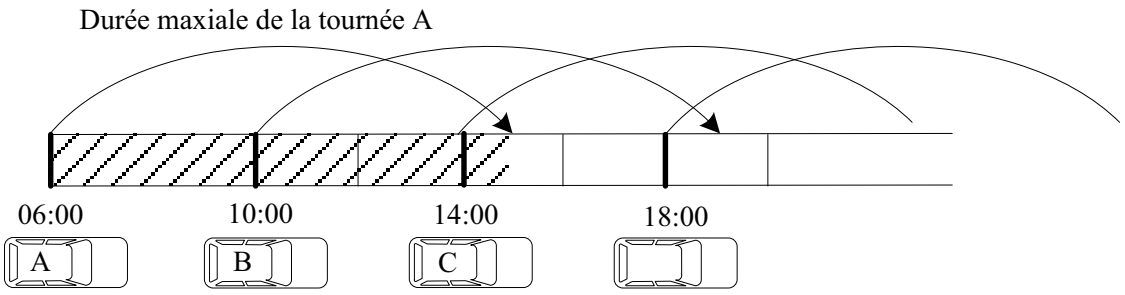


Figure 3.10 : Débuts de tournée possibles pour le véhicule

La solution à 8 clients et deux véhicules du VRP présentée dans la figure 3.2 pourrait être représentée pour un problème de TDVRP par la figure 3.11.

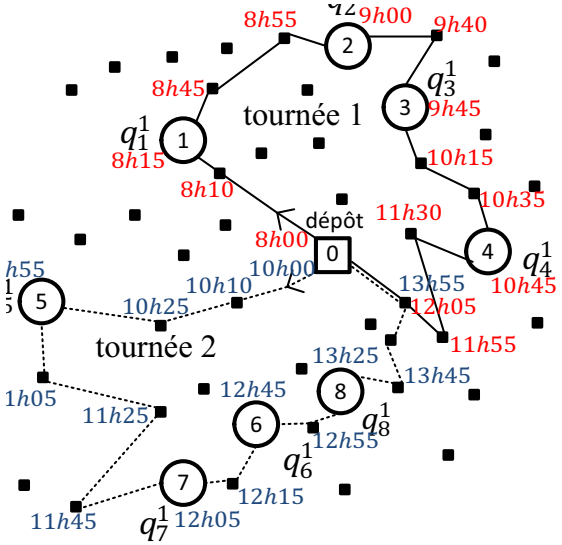


Figure 3.11 : Une solution du TDVRP à 8 sommets et deux véhicules

Dans le modèle, les véhicules peuvent passer plusieurs fois sur le même arc pour aller d'un client à un autre. De plus les sommets clients sont aussi des sommets qui peuvent être utilisés dans le chemin du véhicule pour relier deux clients.

3.2.4. Coût d'une solution et fonction objectif bi-critères du TDVRP.

Le coût associé à une solution est composé de deux critères qui sont la somme de la durée des tournées notée $td(s) = \sum_{i=1}^{m} td(t_i)$ et la somme des temps de conduite notée $rt(s) = \sum_{i=1}^{m} rt(t_i)$ avec t_i une tournée de s. Ce coût est noté $c(s) = \big(rt(s)\ ;\ td(s)\big)$.

Les valeurs $td(t)$ et $rt(t)$ sont les deux critères qui définissent le coût de la tournée t. Ils sont calculés avec les formules suivantes :

$$rt(t) = \sum_{i=1}^{|\lambda|-1} t^{T(i)}_{\lambda(i),\lambda_t(i+1)}$$

$$td(t) = T(|\lambda|) - start(t)$$

Le temps de trajet dépend de la date de départ $T_t(i)$ pour la tournée t du sommet i. La valeur du temps de trajet dépend de la période k à laquelle appartient $T_t(i)$. $T(|\lambda|)$ correspond à la date de retour. La somme des temps d'attentes $wt(t)$ est égale à : $wt(t) = td(t) - rt(t) -$

$st(t)$ avec $st(t)$ la somme des temps de traitement sur les clients contenus dans la tournée qui ne sont pas comptabilisés dans le temps de conduite.

Le coût d'une solution est donc caractérisé par un label composé de deux critères. Ce label regroupe plusieurs critères $L.i$ avec $i \in \{1, \dots, x\}$ où x est le nombre de critères. Une règle de dominance stricte " $\prec$ " peut être définie de la manière suivante :

Définition 3.1: Un label L_1 domine un label L_2 au sens de (Pareto, 1896), $L_1 \prec L_2$, si et seulement si

$$\forall i \in \{1, \dots, x\}\ L_1.i \leq L_2.i \text{ et } \exists j \in \{1, \dots, x\} L_1.j < L_2.j$$

Comme illustré sur la figure 3.12, dans un problème où les deux critères (rt) et (td) sont à minimiser, on obtient une partition de l'espace entre les solutions dominées et les solutions incomparables.

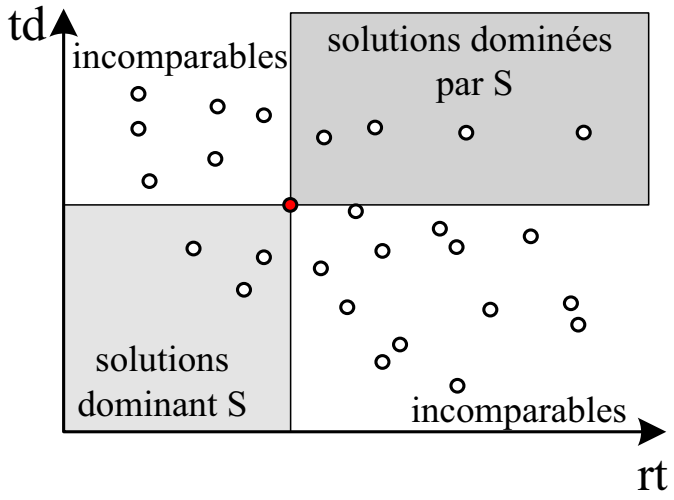


Figure 3.12 : Les relations entre les solutions

Définition 3.2 : un **front de Pareto** est un ensemble de labels tel qu'aucun label n'est dominé par un autre label du front. On parle alors d'ensemble de **labels non-dominés**.

Définition 3.3: le **front de Pareto optimal** est constitué de l'ensemble des labels qui ne sont dominés par **aucun** autre label.

Les labels sont utilisés tout au long de la méthode de résolution avec différents critères. C'est pourquoi ces définitions concernent les labels et pas seulement des labels à deux critères comme ceux associés à une solution. Dans le TDVRP, les deux critères qui composent le coût de la solution sont optimisés de manière à considérer :

- la minimisation du temps de conduite total des tournées, $rt(s)$;
- la minimisation de la durée totale des tournées, $td(s)$.

L'objectif consiste donc à trouver l'ensemble Pareto-optimal. Dans un problème de minimisation, l'ensemble Pareto-optimal peut être représenté sur un graphe identique à celui de la figure 3.13 à gauche. A un front de Pareto est associé une mesure appelé l'hypervolume qui représente la qualité du front, illustrée à droite de la figure 3.13. Cet indice a été introduit par (Zitzler, 1999) pour pouvoir comparer les fronts de Pareto entre eux. Il nécessite de définir des valeurs maximales pour chaque critère.

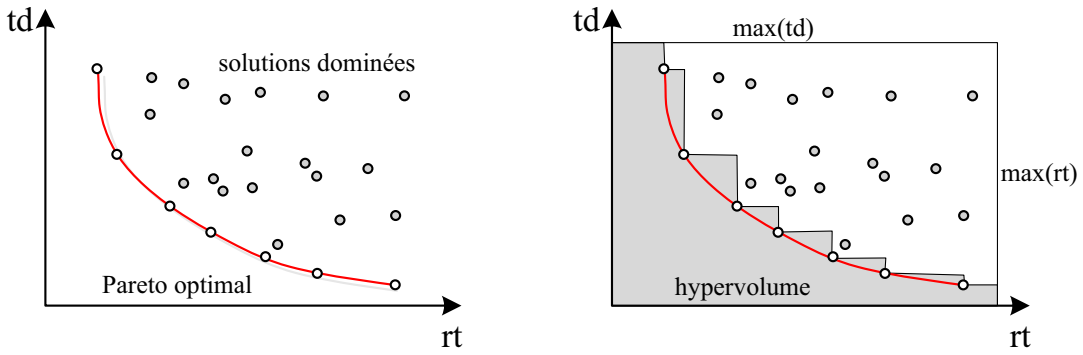


Figure 3.13 : Exemple de front de Pareto (à gauche) et du calcul de l'hypervolume (à droite)

La valeur de max(td) (resp max(rt)) doit être supérieure à l'ensemble des valeurs de td (resp. de rt) pour les solutions qui composent les fronts à comparer.

3.3. Résolution du plus court chemin dans le TDVRP

Contrairement au plus court chemin dans le problème du VRP, il n'y a pas de précalcul simple qui permette de stocker facilement le meilleur chemin entre deux sommets. Ce précalcul est difficile à réaliser en pratique car le meilleur chemin entre deux clients évolue en fonction des dates de départ sur les sommets. Ceci oblige à stocker l'ensemble de tous les plus courts chemins entre deux sommets pour toutes les dates possibles, ce devient qui en pratique rapidement volumineux.

Dans cette partie, les algorithmes utilisés pour calculer le plus court chemin entre deux clients pour une date de départ fixe sont étudiés en considérant deux cas :

- interdire l'attente sur les sommets ;
- autoriser l'attente sur les sommets.

Ce chemin entre deux clients est un chemin dans lequel aucun autre client n'est servi. Avant de présenter les algorithmes, il est nécessaire de détailler le calcul des temps de trajet.

3.3.1. Modélisation des temps de trajet sur les arcs

Dans le modèle, une vitesse est associée à chaque arc et à chaque période. La longueur de l'arc est connue. Un exemple est illustré sur la figure 3.14, pour un arc de 1 à 0. L'arc mesure 25 km et trois périodes sont considérées pour lesquelles la vitesse est définie dans le tableau P_1, P_2, P_3.

Pour calculer la durée du trajet entre les deux sommets du graphe pour une date de départ t, l'idée la plus simple est d'obtenir la période associée à t puis, en utilisant la distance et la vitesse s_{10}^k associée à k, de calculer la durée. Le résultat pour l'exemple est représenté sur la figure 3.15. L'évolution du temps de trajet est alors représentée avec un trait rouge. Dans la figure, il apparaît alors des discontinuités entre les périodes. Ces discontinuités engendrent des problèmes dans le modèle qui peuvent être mise en évidence avec la figure 3.15. Si un premier véhicule part à la date 89, le temps de trajet associé à l'arc est alors de 60 minutes, puis si un second véhicule part deux minutes après, alors son temps de trajet n'est plus que de 30 minutes et donc le second véhicule arrive avec un peu moins de 30 min d'avance par rapport au premier.

Dans un réseau réel, il peut arriver que deux véhicules se doublent durant un trajet et que, par conséquent, le second véhicule arrive avant le premier. Cependant ce cas est considéré comme

marginal dans la mesure où la vitesse limite imposée par la réglementation est identique pour tous les usagers (hors véhicules prioritaires) de même type et dans l'hypothèse où tous les usagers circulent à cette vitesse limite.

distance d_{xy} :	25km

	vitesses s_{xy}^k :
périodes k	**dépôt → x_1**
P_1[0 ; 90]	25km/h
P_2[90 ; 180]	50km/h
P_3[180 ; 270]	30km/h

G_{VRPTD} :

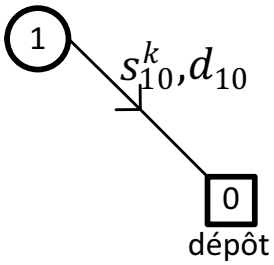


Figure 3.14 : Exemple de vitesses associées à un arc

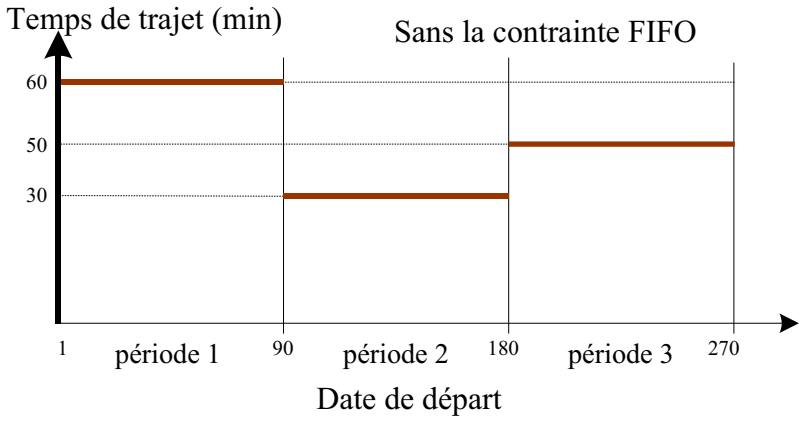


Figure 3.15 : Evolution du temps de trajet d'un l'arc

Pour ne pas prendre en compte les possibilités de dépassement dans le modèle, les temps de trajet doivent respecter une propriété dite "First In First Out" (FIFO). Cette propriété impose que si deux véhicules utilisent un même arc entre deux sommets, le véhicule qui part le premier est aussi celui qui arrive en premier sur le sommet de destination. Il n'y a donc pas de possibilité de dépassement.

Ce type de propriété imposé aux modèles est courant dans la littérature. On peut citer par exemple l'article publié par (Ichoua et al., 2003) où une fonction a été proposée pour calculer des temps de trajet respectant la contrainte FIFO sur un arc dont les vitesses sont constantes et connues par périodes. L'algorithme 1 représente le pseudocode associé à cette fonction. Il prend en paramètres d'entrée une date de départ pour le véhicule ainsi que les sommets de début et de fin de l'arc. Il retourne le temps de trajet noté tt. Parmi les variables globales, il y a la distance de l'arc d_{ij} ainsi que l'ensemble des vitesses s_{ij}^k pour les différentes périodes k considérées.

Le déroulement de l'algorithme est le suivant. A l'aide de la fonction $period(t)$ l'algorithme obtient la période k à laquelle correspond la date t (ligne 3). La vitesse du véhicule à cette période est donc connue grâce à la variable s_{ij}^k. L'algorithme calcule alors la date t_{end} à

laquelle le véhicule arrive s'il voyage à vitesse constante s_{ij}^k. Si cette date est supérieure à la fin de la période k alors le véhicule a changé de période durant le trajet et donc sa vitesse a aussi changé. L'algorithme rentre alors dans la boucle "while" qui va de la ligne 5 à la ligne 10. Tant que la date d'arrivée t_end n'est pas plus petite que la fin de la période courante k, il soustrait la distance parcourue sur l'arc durant le temps passé sur la période courante k à la distance que le véhicule doit parcourir d' (qui initialement est égale à d_{ij}). La période courante devient alors la période $k+1$.

L'algorithme calcule ensuite la nouvelle date de fin t_{end} en prenant compte du changement de vitesse. L'opération est répétée jusqu'à obtenir une date de fin dans la période courante.

Le temps de trajet est arrondi à la minute supérieure à la ligne 11 de l'algorithme.

Algorithme 1 travel_time_calculation_procedure (Ichoua et al., 2003)

Input/Output parameters

t : departure date
i, j : starting and ending nodes

Global parameters

d_{ij} : distance from node i to j
s_{ij}^k : speed of the vehicle between i and j during the period k

Variable parameters

t_{start} : departure date with a specific speed
t_{end} : arrival date with a specific speed
d' : remaining distance

Output parameters

tt : travel time between i and j for the departure time t

Begin

1 $t_{start} := t$
2 $d' := d_{ij}$
3 $k := period(t_{tps})$
4 $t_{end} := t + d_{ij}/s_{ij}^k$
5 **while** ($t_{end} > end\_period(k)$) **then**
6 | $d' := d' - s_{ij}^k\ (endPeriod(k) - t_{start})$
7 | $t_{start} := end\_period(k)$
8 | $k := k + 1$é
9 | $t_{end} := t_{start} + d'/s_{ij}^k$
10 **end while**
11 $tt := \lceil t_{end} - t \rceil$
12 **return** $\{tt\}$
End

La figure 3.16 illustre le résultat obtenu par la fonction proposée par (Ichoua et al., 2003) pour le même exemple que celui de la figure 3.15. Les temps de trajet calculés sont alors représentés par une fonction continue sur la journée. La valeur du temps de trajet est constante sur le début de la période puis diminue ou augmente linéairement pour atteindre la valeur prévue au début de la période suivante.

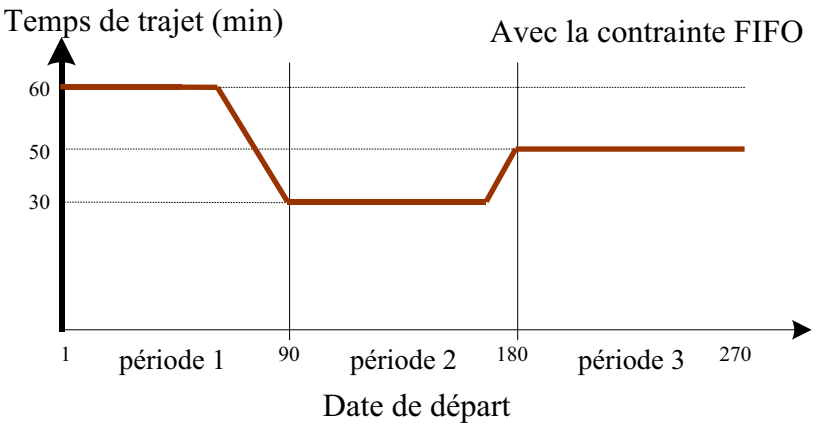


Figure 3.16 : Evolution du temps de trajet d'un l'arc respectant la contrainte FIFO

Cet algorithme est utilisé dans tous les algorithmes qui sont présentés dans la suite de ce chapitre. Pour simplifier les notations, une variable t_{ij}^{x} est utilisée plutôt que de noter l'appel à la fonction "$travel\_time\_calculation\_procedure(x,i,j)$" afin d'exprimer le temps de trajet entre les sommets i et j pour un départ du véhicule à la date x.

3.3.2. Calcul du plus court chemin en interdisant l'attente sur les sommets

Pour le TDVRP un premier calcul du plus court chemin peut être distingué des autres. Il concerne le cas où l'attente sur les sommets du graphe est interdite. Dans cette situation, le véhicule n'a qu'une unique date de départ possible lorsqu'il arrive sur un sommet. Elle correspond à sa date d'arrivée. Le calcul du plus court chemin dans un graphe respectant la propriété FIFO est alors assez proche du calcul de plus court chemin pour un problème dont les temps de trajet ne dépendent pas du temps. La principale différence est que les coûts associés aux arcs ne sont connus qu'au fur et à mesure que l'algorithme avance dans les itérations.

Comme le véhicule ne peut pas attendre et qu'aucun client ne peut être servi entre les deux sommets à relier, la durée totale (td) du chemin est égale à son temps de conduite (rt). Son coût $c(PCC_{uv}^{t}) = td(PCC_{uv}^{t})$ = rt(PCC_{uv}^{t}). Dans cette situation, le problème n'est plus bi-critère et le plus court chemin recherché est unique. C'est le plus court chemin qui minimise le critère (td).

Concernant les notations utilisées, D_u représente la date de départ sur le sommet client u. Le plus court chemin entre u et v est alors PCC_{uv}^{t}. Les variables suivantes sont utilisées :

D_i : date de départ du véhicule sur le sommet i ;

$A_{i\to j}$: date d'arrivée du véhicule sur le sommet j en provenant du sommet i ;

t_{ij}^{x} : temps de trajet sur l'arc (i, j) avec un départ à la date x. Cette valeur est obtenue à l'aide de la fonction "*travel_time_calculation_procedure()*" proposée par (Ichoua et al., 2003).

Pour calculer ce plus court chemin, l'algorithme de Dijkstra peut être utilisé. Comme l'attente est interdite, la valeur de départ D_j sur un sommet j doit être égale à une des dates d'arrivées $A_{i\to j}$. Ensuite, comme la contrainte FIFO implique que plus le véhicule part tôt plus il arrive tôt sur le sommet de destination de l'arc alors la valeur du départ D_j qu'il faut choisir pour minimiser le (td) du chemin est égale à la valeur minimale des $A_{i\to j}$. Comme le temps de trajet du véhicule sur un arc est positif, l'algorithme traite les sommets successivement et l'ordre de traitement des sommets consiste à prendre à chaque itération, le sommet qui a la date d'arrivée minimale la plus faible.

La durée (td) associée au plus court chemin est la différence entre la date d'arrivée au plus tôt sur le sommet v et la date de début sur le sommet u :

$$td = \min_{\forall\, i \in \delta^-} (A_{i \to v}) - D_u$$

La figure 3.17 représente les 3 premières étapes de l'algorithme. Dans la figure de gauche, le graphe initial est représenté. A cette étape les coûts associés aux arcs ne sont pas connus. Ensuite la date de départ sur le sommet u est définie. Dans l'exemple cette date vaut 50. Les temps de trajet des arcs qui ont pour origine u peuvent alors être calculés et les dates d'arrivée sur les sommets successeurs de u sont définies.

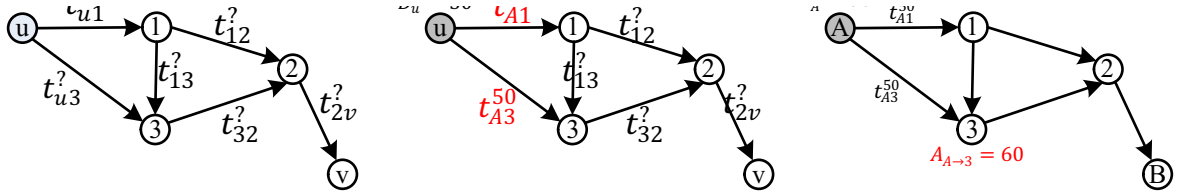


Figure 3.17 : Graphe initial, date de départ de u fixée, propagation des dates d'arrivée sur δ^+de u (1 et 3)

L'étape suivante est illustrée sur la figure 3.18. Le sommet dont la valeur d'arrivée est la plus faible est alors propagée. Dans l'exemple il s'agit du sommet 3. Les coûts des arcs sortant sont alors connus, ce qui permet d'obtenir une date d'arrivée sur le sommet 2.

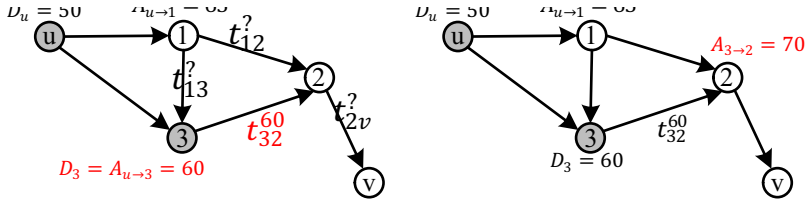


Figure 3.18 : Le sommet avec la date d'arrivée le plus tôt est propagé

L'algorithme continue de propager les valeurs sur les sommets (figure 3.19). Comme dans l'algorithme de Dijkstra, les valeurs des sommets déjà propagées ne peuvent pas être modifiées et seule la plus petite date d'arrivée sur un sommet est conservée.

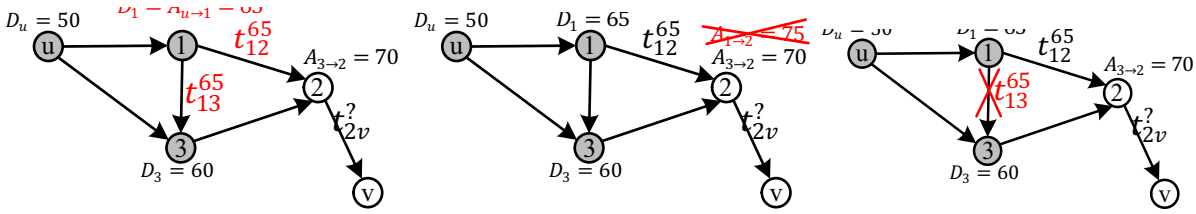


Figure 3.19 : Le sommet 1 est propagé vers les successeurs du sommet 1

Lorsque tous les prédécesseurs du sommet final v ont été traités ou que la date de début du sommet courant est plus grande que la date d'arrivée sur le sommet v, l'algorithme s'arrête, comme illustré dans la figure 3.20.

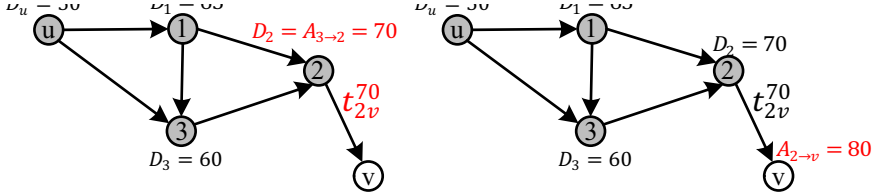


Figure 3.20 : Le sommet 2 est propagé, une valeur d'arrivée sur u est obtenue

Quelle que soit la date t de départ considérée, il existe un unique plus court chemin noté PCC_{uv}^{t}. Son coût est composé d'un unique critère : $c(PCC_{uv}^{t}) = (td)$.

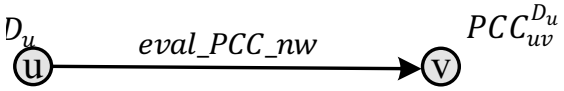


Figure 3.21 : La fonction d'évaluation associe un unique chemin à chaque date de départ

A ce chemin est aussi associée une liste de sommets qui correspond au trajet du véhicule. Comme expliqué dans la partie 3.2.3.1, pour deux dates de départ différentes t et t', les deux plus courts chemins obtenus peuvent être différents en temps et aussi en séquence de sommets. La fonction qui calcule le plus court chemin en temps est appelée dans la suite de ce chapitre $eval\_PCC\_nw(u, v, D_u)$ où nw signifie no-wait.

3.3.3. Calcul du plus court chemin en autorisant l'attente sur les sommets

Le calcul du plus court chemin dans le TDVRP implique de prendre en compte les pauses que le véhicule est autorisé à faire durant le parcours. Autoriser le véhicule à attendre peut être un réel avantage car il permet d'attendre les moments plus propices pour se déplacer dans le réseau et donc de diminuer le temps de conduite. Dans cette situation, il faut donc faire la distinction entre le temps de conduite (rt) et la durée totale (td). Il n'existe donc pas un plus court chemin unique en multicritère mais un ensemble de chemins qui constituent un front. Un chemin est noté C_{uv}^{t}. La différence entre ces deux valeurs est le temps d'attente (wt) puisque, comme pour le cas précédent, aucun client n'est servi sur le chemin. Cette attente peut se situer sur tous les sommets, y compris sur le sommet initial. A un chemin sont donc associés deux critères : le temps de conduite (rt) et la durée totale (td).

$$c(C_{uv}^{t}) = (rt, td)$$

Dans la littérature, le calcul du plus court chemin bi-objectif et multi-objectif a été bien étudié. Plusieurs approches ont été proposées parmi lesquelles des algorithmes avec propagation de labels. Un des articles les plus récents est (Sedeño-Noda and Raith, 2015) qui généralise l'algorithme de Dijkstra au cas bi-objectif. Cet algorithme n'a pas été utilisé dans la méthode de résolution car lorsqu'un label est propagé, un unique label est généré sur le sommet suivant. Or dans le modèle considéré, un label peut être propagé à plusieurs dates possibles ce qui génère un ensemble de labels non-dominés sur les sommets successeurs. Un algorithme spécifique de propagation de label a donc été mise en place.

Le principe de fonctionnement de l'algorithme développé pour calculer le plus court chemin en autorisant l'attente sur les sommets est le suivant : le graphe G_{TDVRP} est initialisé avec une liste de labels initiaux qui sont positionnés sur le sommet u et une date $start_i$ qui correspond à la date à partir de laquelle la tournée du chauffeur commence. Tous les labels initiaux doivent donc correspondre à la même date $start_i$. L'algorithme propage alors les labels dans le graphe en suivant la règle de propagation expliquée dans le paragraphe suivant. Sur chaque sommet du graphe, seuls les labels non-dominés sont conservés et l'algorithme utilise une

règle de dominance définie dans la suite de ce paragraphe. Lorsque l'exécution est terminée, chaque label présent sur le sommet final représente un chemin non-dominé entre u et v.

La figure 3.22 illustre l'appel de cette fonction et les différentes étapes qui la composent. L'ensemble des labels L_v^i correspond à un ensemble de chemins non-dominées C_{uv}^A avec u le sommet d'origine du chemin, v le sommet final du chemin et A la date à partir de laquelle la tournée a commencé pour les labels L_u.

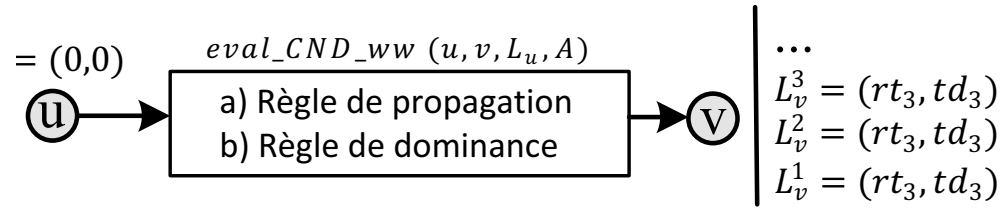


Figure 3.22 : La fonction d'évaluation associe à chaque date de début un ensemble de chemins

Les règles de propagation et de dominance sont les suivantes :

Règle de propagation :

Soit un arc (i, j) auquel est associé un label $L_i(rt, td)$. Pour propager ce label sur le sommet j plusieurs variables sont nécessaires. Premièrement la date de disponibilité A_i^L associée à L_i est calculée. Cette date correspond à la date d'arrivée sur le sommet et correspond aussi à la date à partir de laquelle le véhicule peut quitter le sommet i. La formule suivante est utilisée : $A_i^L = A + td$ avec A la date associée au graphe G_{TDVRP}. Il n'est donc pas nécessaire d'intégrer la date A_i^L dans le label L_i puisqu'elle peut être calculée à partir de td et de A.

Le label L_i peut donc être propagé directement à la date A_i^L où le chauffeur peut effectuer une pause avant de repartir. La pause de durée wt doit respecter certaines contraintes associées :

- wt doit avoir une durée au moins égale à W_{min} ;
- après une pause, le départ du véhicule se fait à une date arrondie au quart d'heure supérieur (xh00, xh15, xh30, xh45), donc $(A_i^L + wt) \bmod 15 = 0$.

Si le chauffeur repart sans faire de pause, alors $D_0 = A_i^L$. Si le chauffeur effectue une pause alors la date de départ au plus tôt D_1 qui respecte les contraintes est égale à :

$$D_1 = arrondi\ au\ quart\ d'heure\ supérieur(A_i^L + W_{min})$$

Puis les dates de départs suivantes sont toutes séparées d'un quart d'heure :

$$D_x = D_1 + 15(x - 1)$$

La durée de la pause effectuée sur le sommet i est égale à $t_x = D_x - A_i^L$. Après que l'ensemble de ces variables soit défini, pour chaque date de départ D_x avec $x \in \{0,1,2, \dots\}$ un nouveau label peut être généré sur le sommet j. La valeur de ce label est égale à :

$$L_j(rt + t_{ij}^{D_x}; td + t_{ij}^{D_x} + wt_x)$$

avec $t_{ij}^{D_k}$ le temps de trajet sur (i, j). Le nombre de départs possibles est fini puisque seules les dates de départ qui respectent les contraintes sur la durée maximale de la tournée sont générées. Sur la figure 3.23, un label L_i^1 est propagé sur l'arc (i, j). La variable A_i^k qui représente la date de disponibilité de ce label est calculée : $A_i^k = A_u + td_k$. Puis les dates de départ possible D_i sont calculées. Pour chacune de ces dates, le temps de trajet sur l'arc $t_{ij}^{D_x}$, est obtenu avec la fonction présentée dans la partie 3.3.1. Un ensemble de labels est alors propagé sur le sommet j.

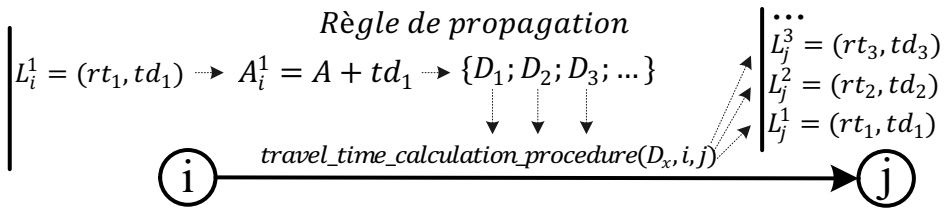


Figure 3.23 : Propagation d'un label

Règle de dominance :

Parmi les labels propagés sur le sommet j, une règle de dominance est appliquée pour ne conserver que les labels qui ont des critères (rt) et (td) non-dominés au sens de (Pareto, 1896). Donc il n'existe pas de label L^1 et L^2tel que :

$$(L^1.rt \geq L^2.rt \text{ et } L^1.td > L^2.td \text{) ou (} L^1.rt > L^2.rt \text{ et } L^1.td \geq L^2.td \text{)}$$

Lorsque plusieurs labels sont présents sur le sommet i, chaque label conduit à la création d'un ensemble de labels sur le sommet j comme illustré sur la figure 3.24. La règle de dominance est appliquée même si la totalité des labels sont issus de la propagation d'un même label. Enfin si des labels sont déjà présents sur le sommet j, alors ces labels sont pris en compte lorsque la règle de dominance est appliquée.

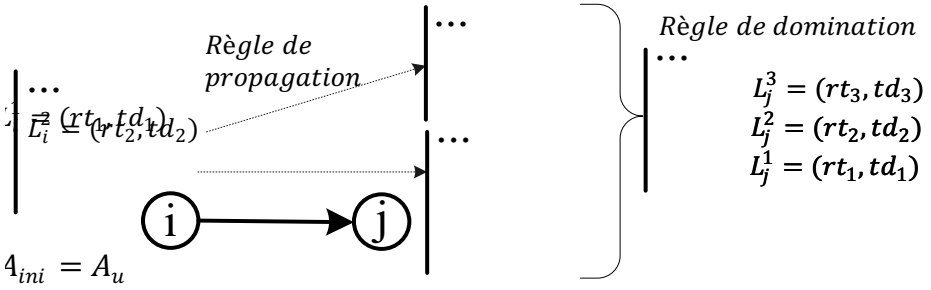


Figure 3.24 : Propagation des labels et utilisation de la règle de dominance sur le sommet j

Une règle de dominance dans un algorithme de programmation dynamique doit permettre de prendre des décisions à l'étape i de l'algorithme de telle sorte que les choix soient optimaux. Ici la règle de dominance définie sur un sommet ne vérifie pas cette contrainte. En effet, il est possible qu'un label L dominé par un autre label P sur un sommet i puisse propager des labels qui par la suite pourraient être meilleurs que certains labels obtenus à partir de P.

Pour illustrer le problème, considérons deux sommets représentés dans la figure 3.25. Au sommet i sont associés deux labels L_i^1et L_i^2 avec L_i^1 (55;180) qui domine L_i^2 (56;190). Cela entraîne en principe la suppression de L_i^2. Rappelons que la durée minimale d'une pause est de 15 minutes. Pour le label L_i^1 les départs possibles sont aux dates : 180, 195, 210, etc. Pour L_i^2 ces choix correspondent au cas où il repart tout de suite à la date 190 et au cas où il fait une pause d'au moins 15 minute et repart au quart d'heure supérieur, c'est-à-dire à 210. La première date de départ 190 pour L_i^2 n'est pas accessible pour le premier label L_i^1. Si le temps de trajet pour cette date est plus intéressant que la différence de (rt) entre L_i^1 et L_i^2 alors un label non-dominé peut être créé. C'est le cas dans l'exemple avec le label $L_j^1(66{,}200)$.

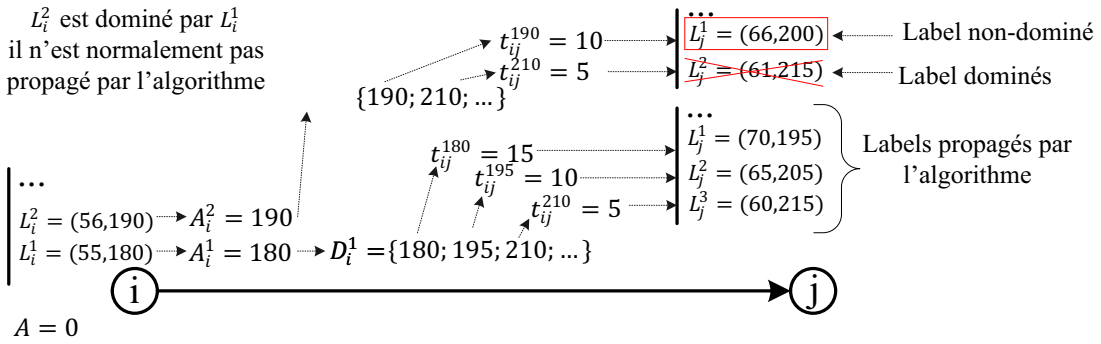


Figure 3.25 : Exemple de propagation et de domination avec perte de labels non-dominés

Le déroulement de l'algorithme consiste à propager les labels d'un sommet courant i vers les sommets successeurs en utilisant la règle de propagation. Les labels propagés sont marqués car un sommet peut être traité plusieurs fois. Uniquement les labels non-dominés sont conservés sur un sommet successeur j et la règle de dominance est utilisée.

Et si un label est ajouté alors le sommet j correspondant est ajouté dans une structure de Tas Minimum. Cette structure est triée par rapport à la plus petite valeur de (rt) des labels non marqués j, notée rt_j^{min}. La condition d'arrêt de l'algorithme est :

- soit le Tas Minimum est vide ;
- soit la valeur de (rt) minimale sur tous les sommets est plus grande que le (rt) correspondant au chemin sans attente. Cela qui justifie l'utilisation du Tas Minimum trié selon le critère rt_j^{min}.

L'algorithme a besoin pour fonctionner d'un graphe. A chaque sommet i de ce graphe est associé un ensemble E_i de labels ainsi que deux entiers rt^{min} et rt^{max} qui représentent les valeurs minimales et maximales associées aux critères (rt) des labels présents dans E_i. Les labels stockés dans E_i représentent les coût des chemins non-dominés entre u et i.

L'algorithme fonctionne en deux étapes et commence par une phase d'initialisation dans laquelle on affecte un label à chaque sommet i du graphe. Il correspond au label sans attente. Ces labels sont obtenus grâce à l'algorithme de Dijkstra introduit dans la section précédente.

A cause de la contrainte de FIFO la solution sans attente est celle qui minimise la durée totale du chemin. Elle ne peut donc pas être dominée par une autre solution. Elle fait donc partie du front recherché.

La méthode de calcul des chemins non-dominés est décrite par l'algorithme 2 et elle nécessite de définir un certain nombre de procédures:

- $initialise\_RT^{max}(G, E, rt^{max}, rt^{min})$: procédure qui affecte sur chaque sommet un label représentant le cas sans attente. Il procède comme pour le calcul de plus court chemin sans attente. Lorsqu'un sommet est propagé, le sommet est ajouté dans un tas ;
- $generate\_labels(L_i, E_j, rt_j^{max}, td_{max})$: à partir d'un label L_i cette fonction génère l'ensemble des labels qui doivent être propagés sur l'arc (i, j). Les labels générés doivent avoir $(rt) < rt_j^{max}$ et $td < td_{max}$;
- $dominance\_rules(L_p, E_i)$: cette fonction utilise les règles de dominance sur les labels pour ajouter si nécessaire le label L_p à l'ensemble E_i. Si le label est ajouté alors la fonction retourne $true$;

- $Push\ (i, rt_i^{min}), Pop(), heap\ is\ empty$: les méthodes associées à une structure de Tas Minimum qui permet de stocker les sommets à traités par l'algorithme. Le Tas Minimum est trié par rapport à la date minimale (rt) des labels qui sont contenus sur les sommets.

Algorithme 2 ***eval_CND_ww***

Input/Output parameters

A_{ini} : Initial departure date
u , v : starting and ending nodes
E_{ini} : set of initial starting time
rt_i^{max} : Maximal riding time value accepted on node i
rt_i^{min} : Minimal riding time value in E_i
td_{max} : Maximal total duration for the path between i and j

Global parameters

G : graph
δ_i^+ : successor of i

Variables

$heap$: Structure heap of nodes, sort by the value rt_i^{min}

Output parameters

E_v : set of non-dominated labels

Begin

stop:= $false$
$E_u := E_{ini}(t)$
$initialiser\_rt^{max}(G, E, rt^{max}, rt^{min}, A_{ini}, heap)$
$k := u$
while (stop = $false$) **then**
  $add := false;$
  for all L **in** δE_k **do**
    for all i **in** δ_k^+ **do**
      $P := generate\_label(L, rt_i^{max}, td_{max}, A_{ini})$
      for all L_p **in** P **do**
        res := $dominance\_rules(L_p, E_i)$
        if (res = true) **then**
          if ($rt_i^{min} < L_p.rt$) **then**
            $rt_i^{min} := L_p.rt$
          end if
          add: = true
        end if
      end for
      if (add = true) **then**
        $Push\ (i, rt_i^{min})$
      end if
    end for
  end for
  if ($heap\ is\ empty\ \boldsymbol{OR}\ rt_i^{min} \geq rt_v^{max}$) **then**
    $stop := true$
  else
    k := $Pop()$
  end if
end while
return {E_v}
End

L'algorithme 2 reçoit en paramètre u et v (deux sommets) et une date de départ. Il retourne un ensemble de labels. Chacun de ces labels correspond au coût d'un chemin non-dominé. Les principales étapes de l'algorithme sont détaillées dans la suite de ce paragraphe.

A la ligne 3, on exécute l'algorithme de plus court chemin sans attente : celui-ci permet d'obtenir un label sur chaque sommet. Le résultat de cet algorithme est illustré sur la figure 3.26 pour le même exemple que celui utilisé pour l'algorithme de plus court chemin sans attente (paragraphe 3.3.2). Ces labels, notés labels sans attente, représentent le label non-dominé ayant le plus grand temps de conduite possible sur chaque sommet.

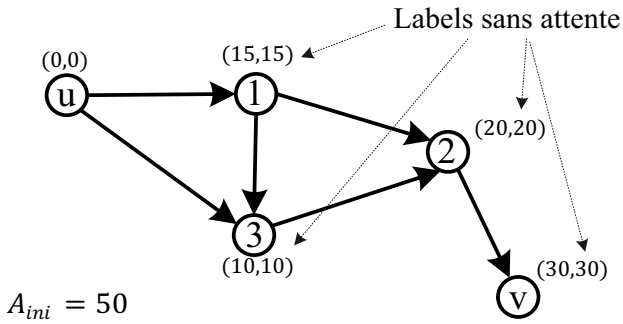


Figure 3.26 : Graphe après initialisation avec les labels sans attente

Le sommet u à gauche de la figure 3.26 devient le sommet courant k. Chaque label contenu dans l'ensemble E_k associé à un sommet k est propagé vers les sommets successeurs notés δ_k^+. Pour cela, la règle de propagation est utilisée et cette étape est correspond à la fonction $generate\_labels(L_i, E_j, rt_j^{max}, td_{max})$ appelée par l'algorithme en la ligne 9.

Cette fonction génère les labels dont le critère (rt) est inférieur à rt_j^{max} et (td) est inférieur à td_{max}. La figure 3.27 illustre la propagation des labels du sommet u vers le sommet 1. La fonction "Domination_rules" est appelée pour chaque label généré (ligne 11). Si au moins un label est ajouté, $add \leftarrow true$ et le sommet successeur noté i est ajouté au tas une fois la totalité des labels ajouté (ligne 20).

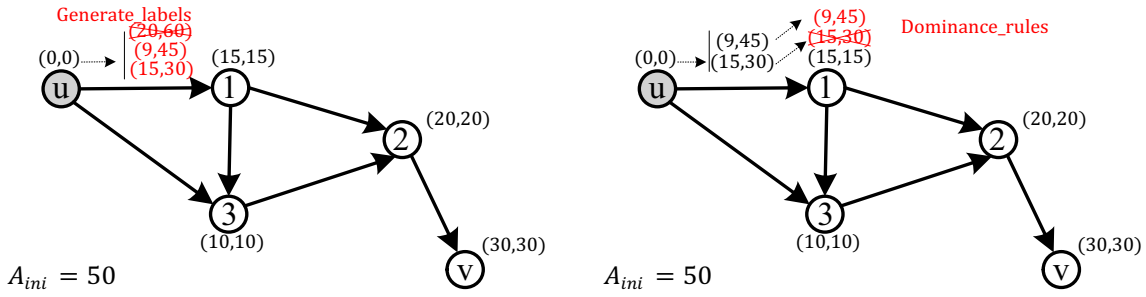


Figure 3.27 : Propagation des labels de u vers 1

Les autres successeurs sont traités sur le même principe et sont ajoutés si nécessaire dans la structure de Tas Minimum

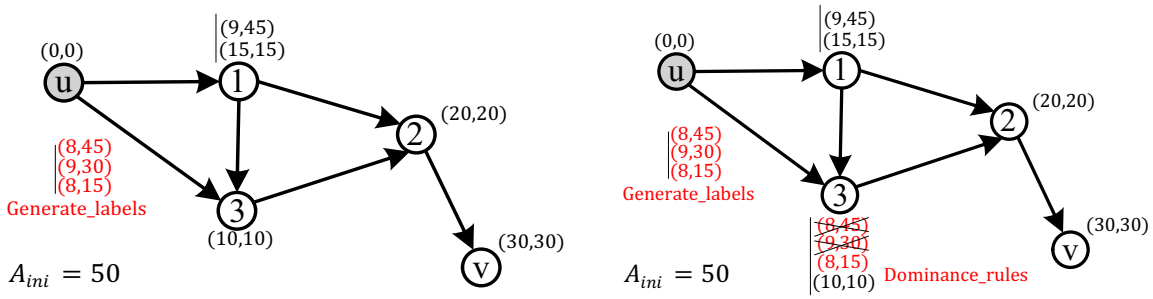


Figure 3.28 : Propagation des labels de u vers 3

Le sommet suivant est celui qui contient la plus petite valeur de (rt). Dans l'algorithme il est obtenu avec la fonction $pop()$ (sommet 3 sur la figure 3.29).

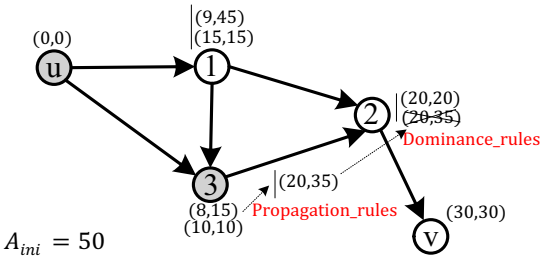


Figure 3.29 : Propagation des labels de 3 vers 2

Dans la figure 3.30 les labels du sommet 1 sont propagés vers le sommet 3 qui a pourtant déjà été traité. Contrairement à $evaluer\_PCC\_NW$, il n'existe pas d'ordre topologique et les labels d'un sommet déjà propagé peuvent être modifiés. Le sommet est alors réinséré dans le Tas Minimum. Lorsqu'il sera traité une seconde fois seuls les labels qui n'ont pas été propagés le seront.

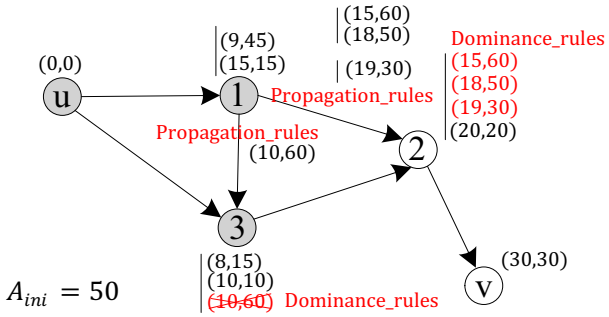


Figure 3.30 : Propagation des labels de 1 vers 2 et 3

L'algorithme se termine quand le Tas Minimum est vide ou quand la valeur minimale du critère (rt) des labels associés au sommet courant est supérieure à la valeur du (rt) maximale sur le sommet final v. Chaque label sur le sommet final correspond à un chemin entre u et v. Pour pouvoir obtenir le chemin associé à un label il est important de conserver la provenance des différents labels.

Cette fonction est utilisée notamment dans l'évaluation des tournées dans les parties suivantes. Elle est appelée $eval\_CND\_ww\ (u, v, L\_u, A)$ où ww signifie "with waiting-time".

3.4. Schéma général de résolution du TDVRP

3.4.1. Une approche en deux étapes

L'approche utilisée pour résoudre le TDVRP se compose de deux étapes illustrées sur la figure 3.31 :

- étape 1 : le cas sans attente du TDVRP est résolu. Dans cette étape, une métaheuristique de type GRASP×ELS est utilisée ;
- étape 2 : on utilise les solutions générées dans la première étape pour construire une population de solutions. Elle sert de point de départ à la résolution du TDVRP par une métaheuristique. Cette seconde autorise des attentes possibles sur les sommets du graphe.

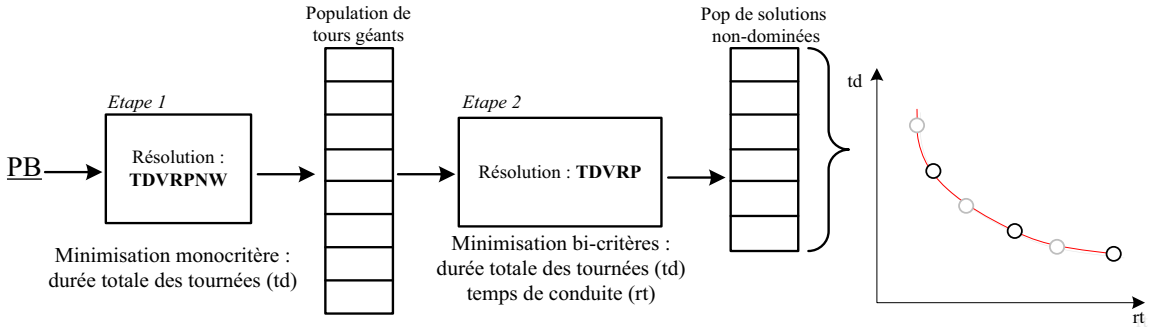


Figure 3.31 : Schéma général de résolution du TDVRP avec attente

Comme l'attente n'est pas autorisée dans la première étape, ce premier problème peut être traité de manière monocritère, ce qui simplifie l'optimisation. De plus une solution sans attente de bonne qualité est intéressante car :

Propriété : étant donnée un ensemble E de solutions non-dominées, il existe une solution sans attente S_{nw} qui appartient au front. Si ce n'est pas le cas, il est possible de la construire à partir des autres solutions du front.

Preuve : Soit un ensemble E de solution non-dominées qui ne contient pas de solution sans attente. Pour construire une solution sans attente S_{nw}, il faut utiliser la solution $S \in E$ qui a la plus petite valeur $td(S)$. Pour chacune des tournées $t \in S$ le vecteur λ est conservé ainsi que la date de début de la tournée. Une nouvelle tournée t' est construite dans laquelle le véhicule n'est pas autorisé à attendre. Comme la contrainte FIFO est respectée par les temps de trajet, le véhicule repart plus tôt en supprimant l'attente sur les sommets. Il arrive donc plus tôt au dépôt final. Soit t' la nouvelle tournée, la valeur $td(t')$ est inférieure à $td(t)$. Par conséquent, la nouvelle solution S'est associée à un critère $td(S') < td(S)$. Comme S est la solution du front avec la plus petite valeur de td(S) et que $td(S') < td(S)$ alors S' est une solution non-dominée de l'ensemble initial.

■

Ceci signifie que si la solution optimale est trouvée pour le cas sans attente, alors cette solution fait aussi partie du front de solutions recherché pour le problème général TDVRP avec attente.

L'étape 1, illustrée par la figure 3.32 à gauche, consiste à trouver une population de solutions qui minimisent le critère (td) et le critère (rt). Durant cette étape l'attente n'est pas autorisée et donc quelque soit la solution on a $(td) = (rt) + cst$, avec cst une constante qui dépend

uniquement du temps de chargement sur les sommets. Ce temps de chargement est le même pour toutes les solutions puisque tous les clients doivent être traités. L'étape 2 cherche à améliorer une population de solutions non-dominées. La population est générée à partir des solutions obtenues à l'étape 1. Cette population est ensuite améliorée comme illustré sur la figure 3.32 à droite. Parmi les solutions de cette population, une solution est une solution sans attente : il s'agit de la solution la plus à droite en bas.

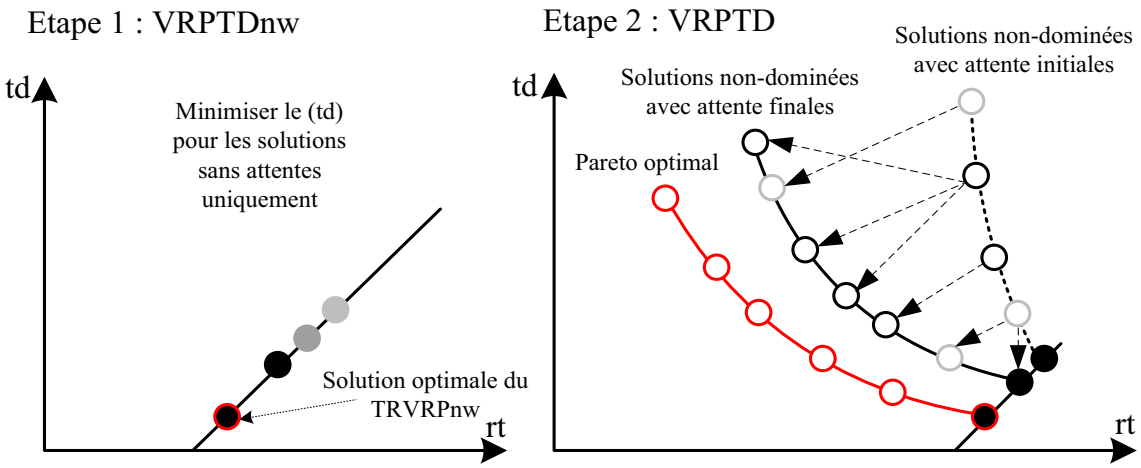


Figure 3.32 : Les solutions qui composent les populations pour deux étapes

La population finale obtenue à la fin de la deuxième étape contient des solutions non-dominées du TDVRP.

3.5. Résolution du TDVRP sans autoriser l'attente

3.5.1. Espace de codage et parcourt de l'espace des solutions

Comme pour la résolution du DARP, l'approche proposée se base sur un espace de codage indirect. Cet espace de codage contient des éléments β constitués d'une suite ordonnée des n clients du problème. Le codage utilisé est un codage de type n-to-1, selon la définition de (Cheng et al., 1996). C'est un espace de codage dans lequel plusieurs éléments β peuvent être associés à une même solution comme illustré dans la figure 3.33. Un élément β ne stocke que les sommets clients. Pour associer un élément de codage à une solution, une fonction dite fonction d'évaluation est définie. L'opération inverse consiste à passer d'une solution S à un élément de codage β : c'est la concaténation.

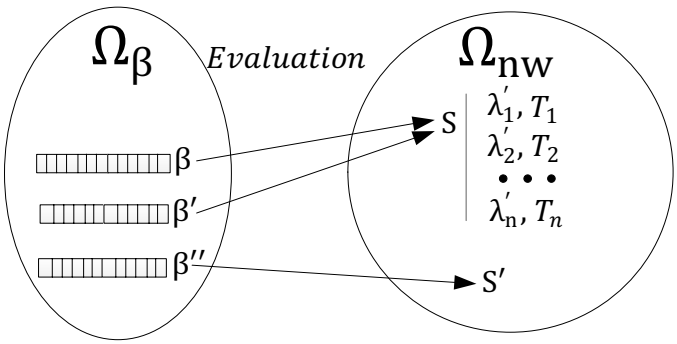


Figure 3.33 : Espace de codage Ω_β associé à l'espace des solutions Ω

Pour parcourir l'espace des solutions, une méthode GRASP×ELS est utilisée. C'est une méthode hybride entre une métaheuristique GRASP (*Greedy Randomized Adaptive Search Procedure*) introduite par (Feo and Resende, 1995) et une métaheuristique ELS (*Evolutionary Local Search*) déjà définie et utilisée dans le chapitre 2 pour la résolution du DARP.

Cette méthode hybride GRASP×ELS a été introduite pour la première fois par (Prins, 2009) pour un problème de VRP. Le schéma est présenté sur la figure 3.34 : à chaque itération du GRASP, une solution initiale est générée à l'aide d'une heuristique randomisée qui crée un élément β suivi d'une première phase d'évaluation et de recherche locale. La solution initiale est ensuite utilisée par la phase ELS. Dans la phase ELS, une solution est concaténée en un élément de codage β. A partir de β un ensemble d'éléments "voisins" β' sont générés à l'aide d'une fonction de mutation. Ces éléments voisins sont évalués afin d'obtenir des solutions S de l'espace Ω sur lesquelles est appliquée une recherche locale. La recherche locale retourne des solutions S'. Les solutions S' sont concaténées en des éléments β''. La population qui contient les meilleurs solutions rencontrées est mise à jour si nécessaire puis l'élément β'' qui correspond à la solution de plus faible coût devient la nouvelle solution courante. La phase ELS est répétée ainsi pendant ne itérations et la phase GRASP est répétée np itérations. Le nombre de voisins générés est nn.

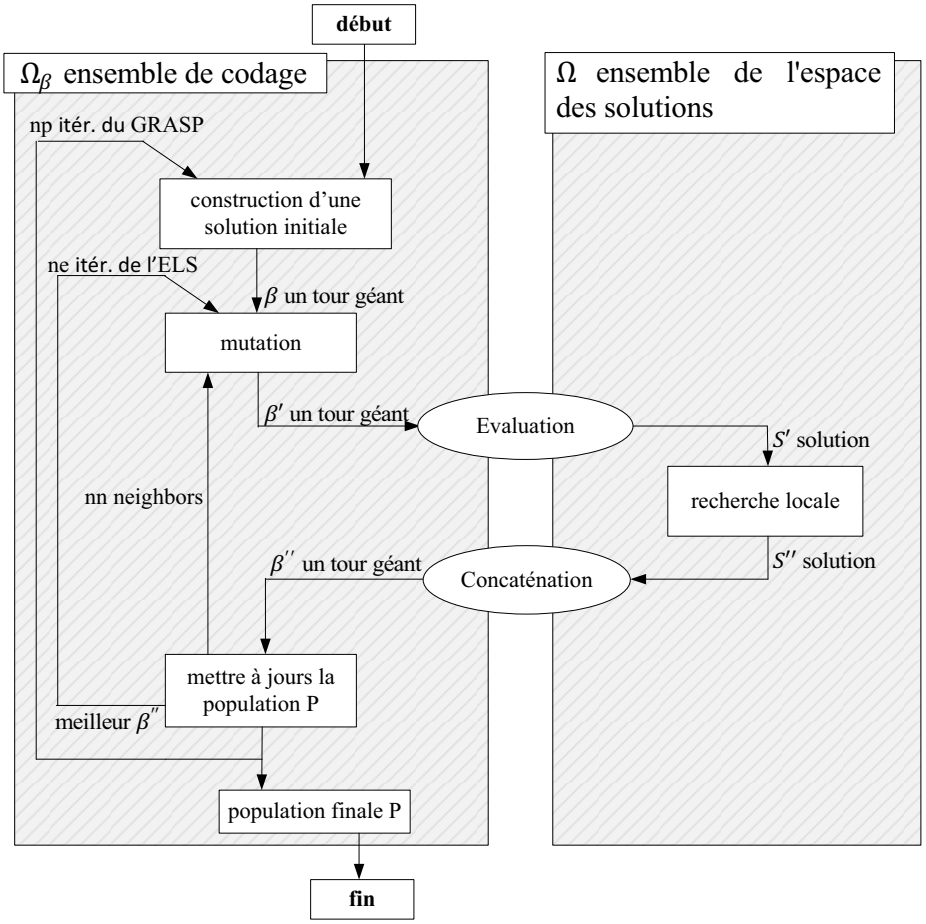


Figure 3.34 : Schéma de la méthode de résolution GRASP×ELS

Les deux phases de GRASP×ELS ont des objectifs spécifiques : la phase ELS explore les voisinages des solutions courantes, la phase GRASP maintient la diversité dans l'exploration globale de l'espace des solutions. Les résultats obtenus par (Prins, 2009) illustrent les

performances d'une telle approche sur le problème du VRP. C'est pourquoi elle a été utilisée sur le TDVRP. Dans la figure 3.34, les points-clé de la méthode sont identifiés :

- les fonctions d'évaluation et de concaténation ;
- la méthode de construction de solutions initiales ;
- la recherche locale ;
- la phase de mutation.

Ces différents points-clé sont présentés dans la partie suivante.

3.5.2. La méthode d'évaluation

Dans la littérature du VRP, une fonction nommée *split* a été proposée pour la première fois par (Beasley, 1983). Cette fonction était utilisée en deuxième partie d'une heuristique *route-first cluster-second* pour résoudre le VRP. La méthode introduite par Beasley permet de découper un tour géant en tournées et ceci de manière optimale par rapport au coût. Il faut noter que cette méthode est restée peu utilisée jusqu'en 2001 où elle a été introduite pour la première fois dans un schéma global d'optimisation. De ce point de vue, le premier article publié est (Lacomme et al., 2001).

Le principe du split consiste à construire un graphe et à associer un coût correspondant à une tournée construite avec une sous-séquence de β. Il est alors possible d'obtenir le découpage optimal à l'aide d'un calcul de plus court chemin dans ce graphe.

Un principe identique est utilisé pour le TDVRP-nw (*no wait*), où l'ensemble des tours géants contenant tous les clients du problème constitue l'espace de codage Ω_β. A partir d'un élément β on construit le graphe associé puis, avec un algorithme de plus court chemin adapté, on obtient le découpage optimal. Pour le TDVRP-nw, sans attente, ce découpage optimal est unique. Le split permet d'évaluer un ensemble de solutions appartenant toutes à l'espace des solutions Ω_{nw}, comme représenté dans la figure 3.35. La fonction split fait appel à une procédure nommée $eval\_trip\_nw()$ qui est présentée dans la partie suivante. Cette procédure fait elle-même appel à la fonction $eval\_PCC\_nw()$ qui a été présentée dans la partie 3.3.2.

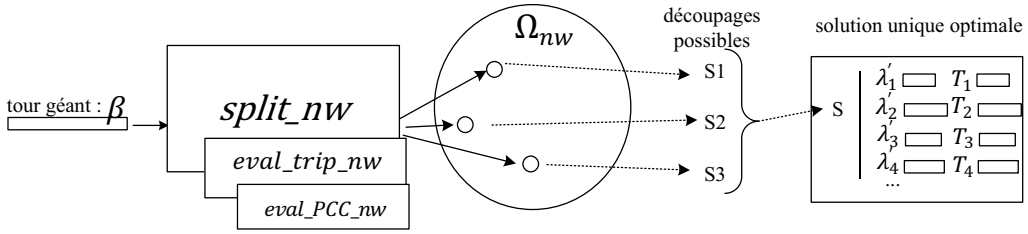


Figure 3.35: Evaluation d'un tour géant sans autoriser l'attente avec la procédure SPLIT

Le graphe G_{split} utilisé par la fonction split se compose de $n + 1$ nodes numérotés de 1 à n et d'un sommet 0 qui sert de point de départ à la méthode. Un arc (i, j) correspond à un λ composé des clients de la sous-séquence entre $\beta(i + 1)$ et $\beta(j)$. Par exemple figure 3.36, l'arc entre 0 et 2 correspond à $\lambda = [0,1,2,0]$. Seul les λ qui respectent la charge maximale des véhicules sont présents dans G_{split}. A un arc λ est associé un coût $c(\lambda)$ qui correspond au coût de la tournée : pour le TDVRP-nw ce coût est composé d'un seul critère.

Le split consiste à propager des labels d'un sommet initial vers les autres sommets du graphe G_{split} . Les labels associés à un sommet i sont composés de deux critères (vu, c) avec vu le nombre de véhicules utilisés et c_i le coût associé au plus court chemin entre 0 et le sommet i. Sur un sommet, seuls les labels non-dominées sont conservés. Comme illustré sur la figure 3.36, le label sur le sommet final avec la plus petite valeur c_n est conservé. Ce label

correspond à la valeur du chemin optimal associé à β.

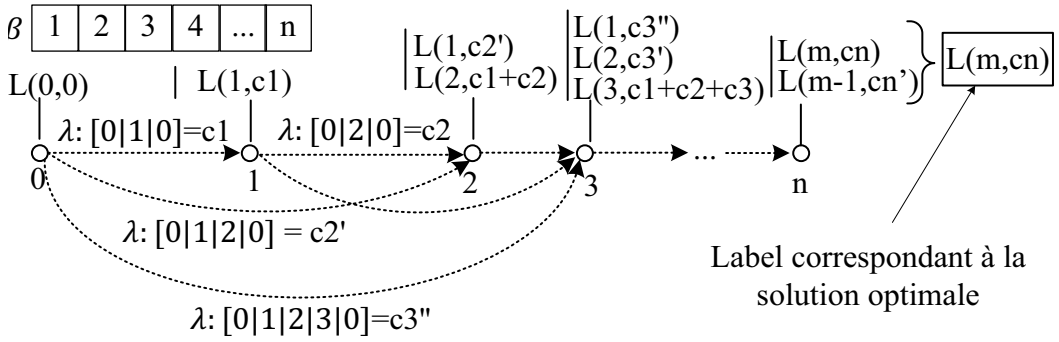


Figure 3.36: Schéma du SPLIP

Comme pour la propagation de label dans le calcul du plus court chemin, une règle de propagation est définie ainsi qu'une règle de dominance :

Règle de propagation :

Soit un sommet i auquel est associé un label $L_i(vu, c)$, soit (td) le coût associé à l'arc (i, j). Le label propagé sur le sommet j est :

$$L_j(vu + 1\ ;\ c + td)$$

Règle de dominance :

Parmi les labels propagés sur le sommet j, une règle de dominance est appliquée pour ne conserver que les labels qui ont des critères (vu) et (c) non-dominés au sens de (Pareto, 1896). Donc il n'existe pas de label L^1 et L^2tel que :

$$(L^1.vu \geq L^2.vu \text{ et } L^1.c > L^2.c\ ) \text{ ou } (L^1.vu > L^2.vu \text{ et } L^1.c \geq L^2.c\ )$$

Dans le split le critère vu associé à un label correspond au nombre de véhicules utilisés. Seuls les labels avec une valeur $vu < m$ peuvent donc être propagés. Le split est décrit dans l'algorithme 3. Il nécessite de définir un certain nombre de procédures :

- $initialiser\_label$: initialise les ensemble $E_i \forall i \in \{1, \dots, n\}$ à vide ;
- add_{client} : à partir d'un vecteur λ, ajoute un client en avant-dernière position, juste avant le dépôt final, ex. : $\lambda = [0; 1; 2; 3; 0], add\_client(\lambda, 4) \rightarrow \lambda = [0; 1; 2; 3; 4; 0]$;
- $eval\_trip\_nw$: associe un coût unique à l'évaluation d'un λ. Cette procédure retourne -1 s'il n'existe pas de tournée valide ;
- $Propagate\_split$: propage les labels vers le sommet successeur en respectant la règle de propagation et la règle de dominance des labels.

L'algorithme 3 prend en entrée un élément β et retourne une solution si au moins un label est présent dans l'ensemble des labels associé au sommet final. L'ensemble des labels associé au sommet i est noté E_i. E représente l'ensemble des E_i avec $i \in \{1, \dots, n\}$. Le graphe qui sert de support aux explications dans la partie précédente n'est pas construit explicitement dans la fonction. L'algorithme de la ligne 5 à la ligne 21 traite les sommets de β successivement de 1 jusqu'à i. De la ligne 8 à la ligne 20 on effectue la propagation des labels jusqu'à ce qu'une ou plusieurs contraintes du problème ne soient plus vérifiées. Ces contraintes sont la quantité transportée par le véhicule et la durée totale de la tournée. Lorsque la valeur de la tournée associée aux sommets allant de i à j est obtenue à la ligne 15, les labels de E_i sont propagés vers E_j en suivant la règle de propagation. Seuls les labels non-dominés sont conservés dans E_j.

Algorithme 3 split_nw

Input/Output parameters
β : giant tour

Output parameters
S : a solution

Begin
$L \coloneqq (0,0)$
$initialiser\_label(\beta, E)$
$E_0 \coloneqq insert(L)$
$i \coloneqq 0$
while $(i < n)$ **then**
| $load \coloneqq 0$
| $j \coloneqq i$+1
| **repeat**
| | load = load + q_j
| | **if** ($j = i + 1$) **then**
| | | $\lambda \coloneqq [0; \beta[j]; 0]$
| | **else**
| | | $\lambda \coloneqq add\_client(\lambda, \beta[j])$
| | **end if**
| | $C_\lambda \coloneqq eval\_trip\_nw(\lambda)$
| | **if** ($C_\lambda > 0$) **then**
| | | $Propagate\_split\ (E_i, C_\lambda, E_j)$
| | **end if**
| | $j \coloneqq j + 1$
| **until** $(j > n)$ **OR** $(load + q_{j+1} > Q_k)$ **OR** $(|C_\lambda| = 0)$
end while
if ($|E_n| \geq 1$) **then**
| $S \coloneqq create\_solution\ (\beta, E)$
end if
return {S}
End

1. Evaluation d'un vecteur λ associée à une tournée

Un vecteur λ se compose d'une suite ordonnée de clients commençant et finissant au dépôt. L'objectif de l'évaluation consiste à calculer le coût associé à cette tournée en déterminant la date de départ du camion pour chaque sommet de la tournée (cela revient à déterminer l'attente au niveau de chaque sommet). Dans le cas sans attente (TDVRP-nw) le coût associé à la tournée est composé du critère unique (td).

Dans le TDVRP-nw, l'évaluation consiste à calculer la valeur (td) de la tournée pour chaque date de départ possible $départ_i$ et, parmi toutes les valeurs de (td), à sélectionner le chemin de coût minimal. La fonction $eval\_PCC\_nw()$ est appelée pour calculer le coût du plus court chemin entre les clients consécutifs de λ. Comme le plus court chemin sans attente est unique par date de départ dans le TDVRP, la date d'arrivée sur le sommet suivant est égale à D_{i-1}à laquelle est ajoutée la durée du trajet $tt_{i-1,i}$. L'attente est interdite. Cependant, comme un temps de chargement est imposé sur chacun des clients, il faut le prendre en compte pour calculer la date de départ d'un sommet $i : D_i = A_i + lt_{\lambda(i)}$. Ces étapes sont répétées jusqu'à arriver sur le dernier sommet de λ qui correspond au dépôt. Le coût de la tournée est alors calculé et il correspond à la durée totale de la tournée (td), c'est-à-dire à la différence entre la date de départ sur $\lambda(0)$ et la date d'arrivé sur le sommet $\lambda(|\lambda|)$. Cette fonction est illustrée sur la figure 3.37.

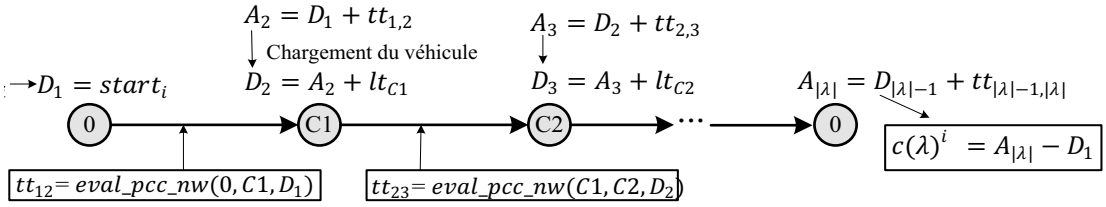


Figure 3.37: Evaluation d'un λ sans autoriser l'attente pour une date de départ

Le véhicule ne peut pas attendre. Par contre, il existe plusieurs dates de début possibles sur le dépôt initial. La fonction $evaluer\_trip\_nw()$ évalue donc le (td) associé à chacune des dates de départ puis sélectionne celui de valeur minimale, comme illustré dans la figure 3.38.

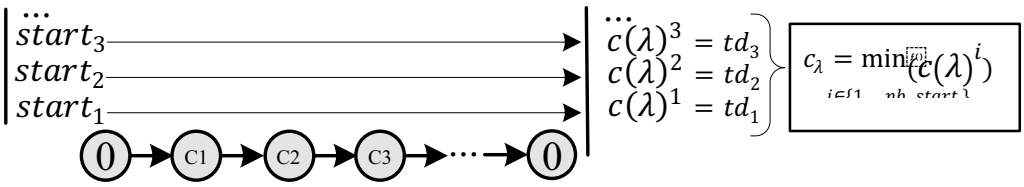


Figure 3.38: Evaluation générale d'un λ sans autoriser l'attente

L'algorithme 4 correspond au pseudo-code associé à cette fonction. Il traite chaque date de départ successivement à l'aide de la boucle "for" qui commence à la ligne 2 et qui se termine à la ligne 17. Pour chaque date il calcule le coût du trip associé, à l'aide d'une boucle "while" de la ligne 7 à la ligne 13. Cette boucle est stoppée si le coût de la tournée est supérieur au coût d'une des tournées précédemment calculée et dont la valeur est dans C_λ. Après avoir évalué toutes les tournées possibles, la fonction retourne le coût minimal contenu dans C_λ.

Algorithme 4 eval_trip_nw

Input/Output parameters
λ : list of clients

Output parameters
C_λ : cost value of the trip

Begin
1 $C_\lambda \coloneqq inf$
2 **for** i **from** 1 **to** nb_start **do**
3 | $t \coloneqq start_i$
4 | $v \coloneqq \lambda(1)$ **// depot_node**
5 | $i := 2$
6 | $C_{temps} \coloneqq 0$
7 | **while** $(i \leq |\lambda|)$ ***and*** $(C_\lambda < C_{temps})$ **then**
8 | | $u \coloneqq v$
9 | | $v \coloneqq \lambda(i)$
10 | | $C_{temps} \coloneqq C_{temps} + eval\_PCC\_nw(u, v, t) + lt(v)$
11 | | $i \coloneqq i + 1$
12 | | $t \coloneqq start_i + C_{temps}$
13 | **end while**
14 | **if** $(C_\lambda > C_{temps})$ **then**
15 | | $C_\lambda \coloneqq C_{temps}$
16 | **end if**
17 **end for**
18 **return** $\{C_\lambda\}$
End

Dans le cas de la résolution du problème sans autoriser l'attente, à chaque solution est associé un coût unique. Ce coût représente la durée minimale de la tournée λ pour toutes les dates de départs possibles sur le dépôt.

3.5.3. Les phases de mutation et de recherche locale

Les phases de mutation et de recherche locale se succèdent dans la phase ELS. Comme illustré sur la figure 3.39, les phases de mutations sont effectuées dans l'espace de codage Ω_β alors que la phase de recherche locale, elle, se déroule dans l'espace des solutions Ω_{nw}. Pour passer d'un espace à l'autre ; les fonctions d'évaluation et de concaténations sont utilisées.

La fonction d'évaluation a été présentée dans la partie précédente. La fonction de concaténation permet de passer d'une solution S à un élément β. Elle procède en deux étapes : un ordre de traitement des tournées qui composent la solution est d'abord choisi aléatoirement puis, selon cet ordre, un tour géant est construit en recopiant les clients dans l'ordre ou ils sont traités dans la tournée.

Pour générer des éléments voisins de l'élément β, on permute deux sommets clients choisis aléatoirement dans β. Ces éléments voisins β' sont alors évalués.

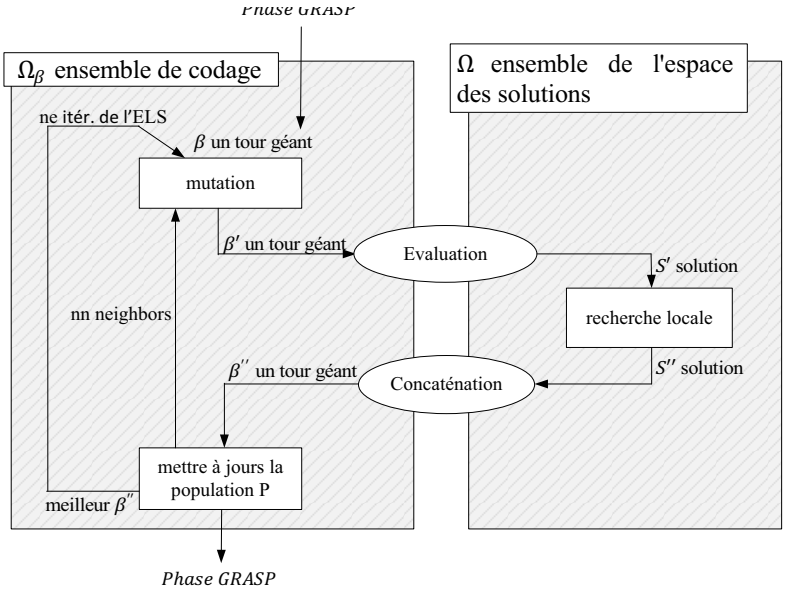


Figure 3.39 : Schéma de la phase ELS de la méthode

La recherche locale est appliquée sur une solution : elle se compose d'opérateurs classiques utilisés pour le VRP, qui sont décrit en détail dans l'article de (Prins, 2009) :

- 2-opt intertournées ;
- 2-opt intratournées ;
- 3-opt intratournées.

Ces mouvements sont effectués en permutant l'ordre de traitement des clients dans la tournée. La différence par rapport aux mouvements appliqués dans le VRP est que l'évaluation d'un mouvement ne peut pas être réalisée en temps constant car il faut réévaluer une partie de la tournée. En effet, modifier une date d'arrivée sur un sommet entraîne la modification de toutes les dates de passage sur les sommets suivants et par conséquent des plus courts chemins, comme illustré dans la figure 3.40.

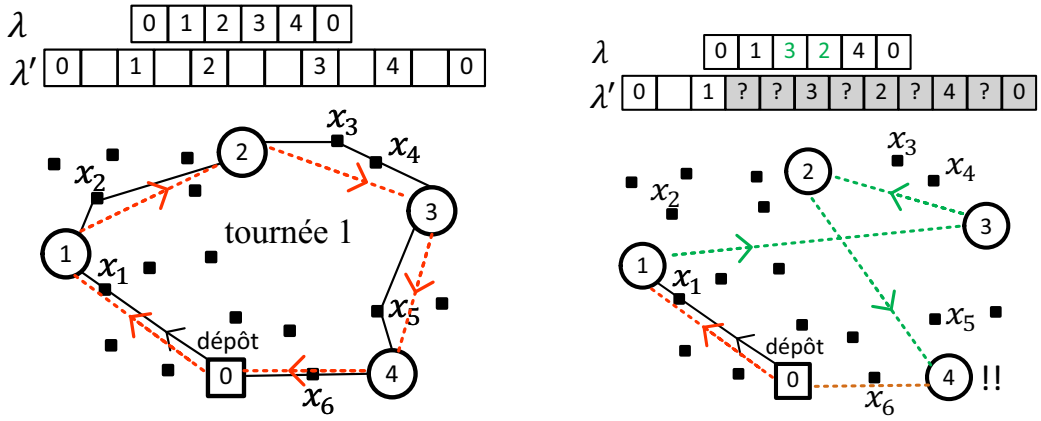


Figure 3.40: La modification de l'ordre des clients implique la modification dans la tournée

Sur la figure 3.40, en modifiant l'ordre de passage de λ pour obtenir [0 1 3 2 4 0], il faut réévaluer en partie la tournée. Les dates de départ sur les sommets qui précèdent la modification sont conservées ainsi que les plus courts chemins entre ces sommets. Sur la figure 3.40, les chemins entre le dépôt et le client 1 dans les deux tournées restent identiques. Ensuite les plus courts chemins sont évalués successivement en utilisant la fonction $évaluate\_PCC\_nw()$ jusqu'à ce que le véhicule arrive sur le premier sommet non modifié dans λ. Sur la figure 3.40 il s'agit du client 4. Si la date d'arrivée sur ce client est supérieure à la date précédente, l'évaluation du mouvement est arrêtée. La contrainte FIFO sur les arcs assure que la date d'arrivée sur les clients suivants est supérieure ou égale à la date précédente, donc le coût n'est pas amélioré. Par contre si la date d'arrivée est inférieure alors le mouvement est améliorant et la fin de la tournée est évaluée. La recherche locale est donc plus coûteuse que pour le VRP, où l'on peut vérifier en $O(1)$ si le mouvement est améliorant.

3.5.4. Création de solutions initiales

Un algorithme de plus proche voisin randomisé est utilisé pour la génération de solutions initiales. Cet algorithme construit un élément β en ajoutant à chaque itération un client à la fin de β. Ce client est choisi selon un critère. Dans le cas du TSP c'est la distance entre le dernier sommet de β et les autres sommets du graphe. Dans le cas du TDVRP plusieurs critères peuvent être envisagés :

- la distance entre les clients ;
- une borne inférieure ou supérieure sur les temps de trajet entre les clients. Ce temps de trajet peut être calculé en utilisant uniquement les plus grandes ou les plus petites vitesses sur les arcs ;
- le temps de trajet minimal entre les deux clients sans autoriser l'attente ;
- le temps de trajet minimal entre les deux clients en autorisant l'attente (ce critère est le plus coûteux en temps de calcul).

Dans la méthode, on a choisi le plus court chemin en termes de distance. Ce critère ne varie pas durant la journée et donne une information sur le temps de trajet entre les clients puisque la distance intervient dans le calcul du temps de trajet sur un arc.

3.6. Résolution du TDVRP

3.6.1. Espace de codage et méthode de résolution

Pour résoudre le TDVRP dans lequel l'attente est autorisée (TDVRP-ww), il est possible d'utiliser le même espace de codage que celui utilisé pour le TDVRP-nw : il s'agit de

l'ensemble des tours géants Ω_β. Une nouvelle fonction d'évaluation a été développée, basée elle aussi sur Split. Elle permet d'associer une solution de Ω à un élément de cet espace de codage, comme le montre la figure 3.41. Comme une solution du TDVRP-nw est aussi une solution du TDVRP, l'espace des solutions du TDVRP-nw est un sous-ensemble de l'espace des solutions du TDVRP-ww.

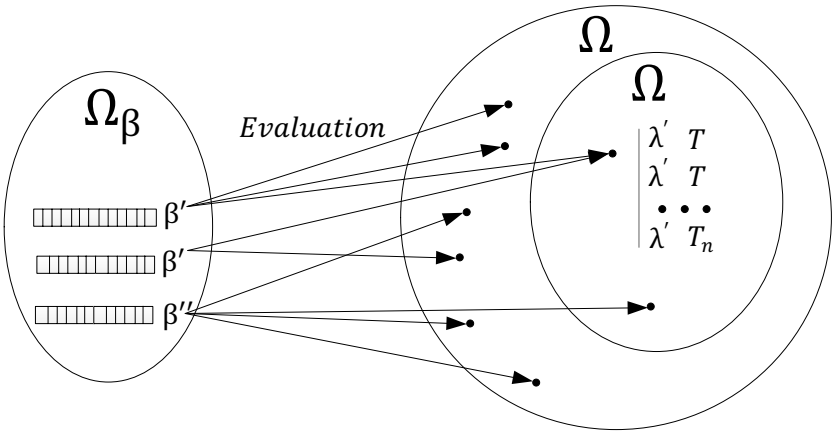


Figure 3.41 : Espace de codage Ω_β associé à l'espace des solutions Ω

L'efficacité de la méthode de résolution du TDVRP-ww repose essentiellement sur une procédure efficace de parcours de l'espace de recherche (temps nécessaire pour générer et évaluer une solution) mais aussi en terme de diversité pour parcourir des régions de l'espace qui sont susceptibles de contenir des solutions de bonne qualité. Le parcours de l'espace des solutions se fait de manière indirecte en parcourant l'espace Ω_β via une procédure de type *Path Relinking*.

2. La procédure Path Relinking

Les approches de type Path Relinking (PR) sur un problème de VRP ont été proposées pour la première fois par (Glover et al., 1993) pour diversifier les solutions dans un algorithme de recherche tabou et un état de l'art est publié par (Ho and Gendreau, 2006). Une stratégie de type PR consiste à générer une nouvelle solution à partir de deux solutions en parcourant une liste de solutions intermédiaires.

L'originalité de l'approche introduite ici réside dans la définition d'une stratégie de type PR entre deux éléments de codage β′ et β′′ et pas entre deux solutions. Les éléments intermédiaires β générés sont des éléments de codage obtenus par permutation de sommets dans β. Puis ces éléments sont évalués à l'aide la nouvelle fonction split qui est présentée dans la partie suivante. Cette fonction renvoie un ensemble de solutions non-dominées pour chaque β. Le résultat du PR est donc lui-aussi un ensemble de solutions non-dominées issues des évaluations, comme illustré sur la figure 3.42.

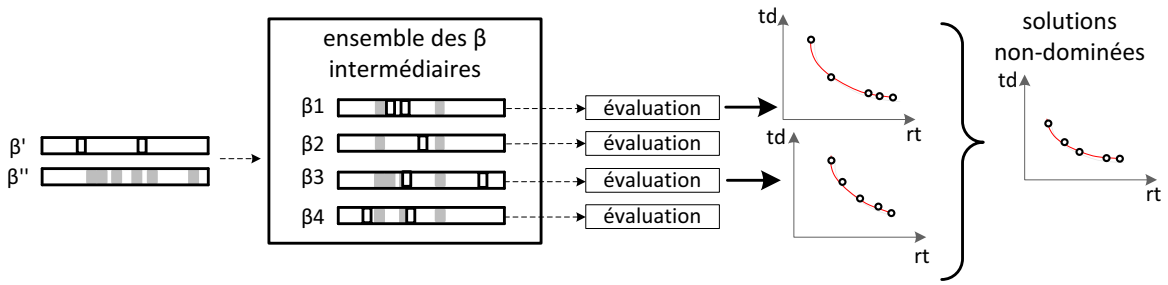


Figure 3.42 : Approche du Path Relinking

La notion de chemin entre deux éléments β repose sur une distance entre les éléments de codage. Elle est expliquée dans le paragraphe suivant.

3. La distance entre les éléments β

Pour pouvoir appliquer le PR, il faut être capable de construire des "chemins" entre deux éléments β' et β'' de l'espace de codage Ω_β. Cette notion de chemin suppose qu'il existe un ensemble de mouvements élémentaires qui permet, en partant de β', d'obtenir β''. Le mouvement élémentaire choisi dans cette approche correspond à une permutation entre deux clients dans β.

Afin d'obtenir l'ensemble des permutations nécessaires, l'algorithme développé par (Zhang et al., 2005) est utilisé. Un algorithme détaillé est proposé dans (Lacomme et al., 2015). Il permet d'obtenir en plus du nombre de permutations minimales, une liste de permutations à appliquer pour passer de β' à β''. A partir de cette liste, il est donc possible de générer un ensemble d'éléments intermédiaires comme illustré dans la figure 3.43.

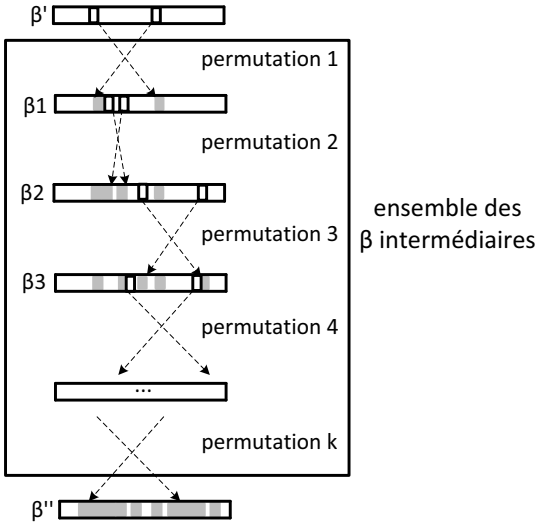


Figure 3.43 : Nombre de permutations minimales

Il faut souligner que l'algorithme de (Zhang et al., 2005) génère une liste de permutations possibles dont le nombre est minimal. Il existe cependant d'autres listes de permutations. Un exemple de cette situation est mis en évidence dans la figure 3.44. A gauche, les permutations pour passer de β' à β''sont (1,2) puis (3,4) et à droite ce sont (3,4) puis (1,2). Si la liste de permutations n'est pas la même, la liste des β intermédiaires est également différente et donc le Path Relinking n'explore pas le même espace de codage. Par exemple dans la figure pour passer de β' à β'', β_1= [2,1,3,4] est exploré dans le premier cas alors que dans le second cas c'est β_2= [1,2,4,3].

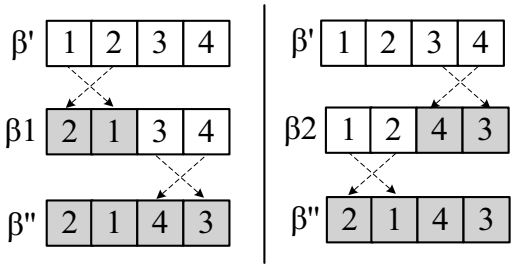


Figure 3.44 : L'ordre des permutations influence l'espace des solutions parcourut par les β

Dans l'algorithme du PR mis en place sur ce problème, l'ordre choisi correspond à celui obtenu par l'algorithme de (Zhang et al., 2005) .

4. Schéma général de résolution

Le Path Relinking (PR) est inclus dans une méthode globale de résolution et il est combiné à une recherche locale afin d'explorer l'espace des solutions. Comme pour GRASP×ELS, le parcours de l'espace des solutions est réalisé en passant par l'espace de codage dans PR et en partie dans l'espace des solutions avec la recherche locale, (figure 3.45). La méthode commence avec un ensemble d'éléments de codage β qui permet de générer une population initiale de solutions. Cette population est sauvegardée afin de pouvoir modifier ces éléments sans modifier la population en cours de traitement. Dans cette population, deux solutions S' et S'' sont sélectionnées. Plusieurs choix peuvent être envisagés pour effectuer cette sélection. Ces deux solutions sont concaténées pour obtenir deux éléments β' et β''. La fonction de concaténation est identique à celle utilisée dans la partie 3.5 où les solutions sans attente sont considérées. La fonction procède en deux étapes : premièrement un ordre de traitement des tournées qui composent la solution est choisi aléatoirement ; deuxièmement en suivant l'ordre imposé, un tour géant est construit en recopiant les clients dans l'ordre où ils sont traités dans la tournée. Sur β' et β'' est ensuite appliqué un Path Relinking. Les solutions intermédiaires sont alors générées et successivement évaluées. La population de solutions non-dominées obtenues est alors insérée dans la population sauvegardée en conservant uniquement les solutions non-dominées. Lorsque les PR sur la population sont terminés la population courante est remplacée par la population sauvegardée qui a été mise à jour. Puis de nouvelles itérations de PR sont réalisées sur la nouvelle population courante.

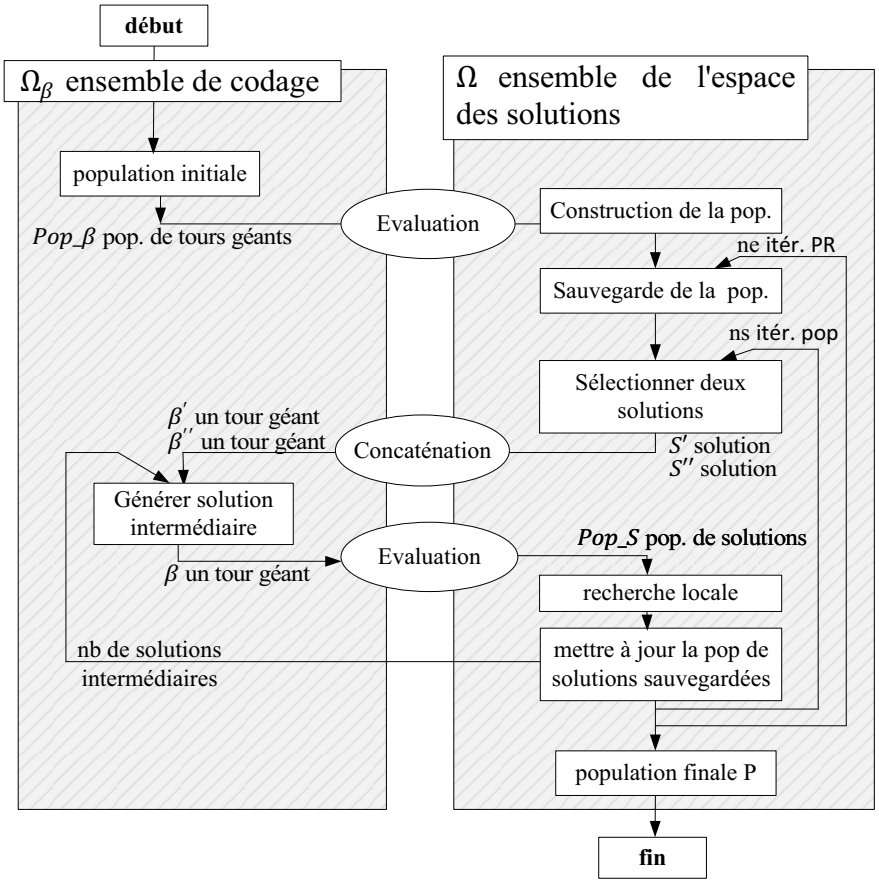


Figure 3.45 : Schéma de la méthode de résolution utilisant le Path Relinking

Concernant la sélection des deux solutions sur lesquelles sont appliqués le Path Relinking, le choix consiste à toujours prendre la solution qui a le plus faible temps de conduite (rt) pour construire l'élément β' puis à faire varier à chaque exécution la solution associée à l'élément β''comme illustré dans la figure 3.46.

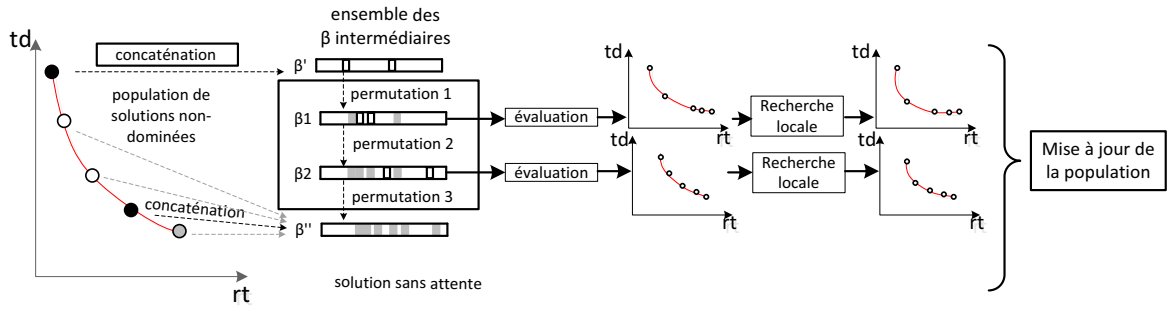


Figure 3.46 : Schéma de la méthode de résolution du TDVRP

Dans la figure 3.45 et la figure 3.46, les points-clé de la méthode de résolution sont :

- la construction de la population initiale ;
- les fonctions d'évaluation et de concaténation ;
- la méthode de construction de la population initiale de solutions ;
- la génération des β intermédiaires, présentée dans la partie précédente ;
- la recherche locale.

3.6.2. La méthode d'évaluation

Dans le cas sans attente, qui a été présenté dans le paragraphe 3.5.2, l'évaluation retourne un découpage optimal unique parmi tous les découpages possibles de l'élément β. Ce découpage correspond à la solution avec le plus petit coût td. Mais pour le TDVRP avec attente, il n'existe plus un découpage optimal unique sur les solutions générées à partir de β. Le problème est multicritère car, à un élément β, est associé un ensemble de solutions non-dominées. Certaines solutions peuvent être obtenues à partir des mêmes découpages. Elles sont alors constituées des mêmes λ avec des dates de passages différentes. Mais elles peuvent aussi être composées de λ différents et donc de découpage β différents, comme le montre la figure 3.47.

Parmi cet ensemble de solutions non-dominées retournées par la fonction split, une seule solution provient de l'espace des solutions sans attente. Cette solution est la même que celle obtenue avec l'évaluation sans attente de β.

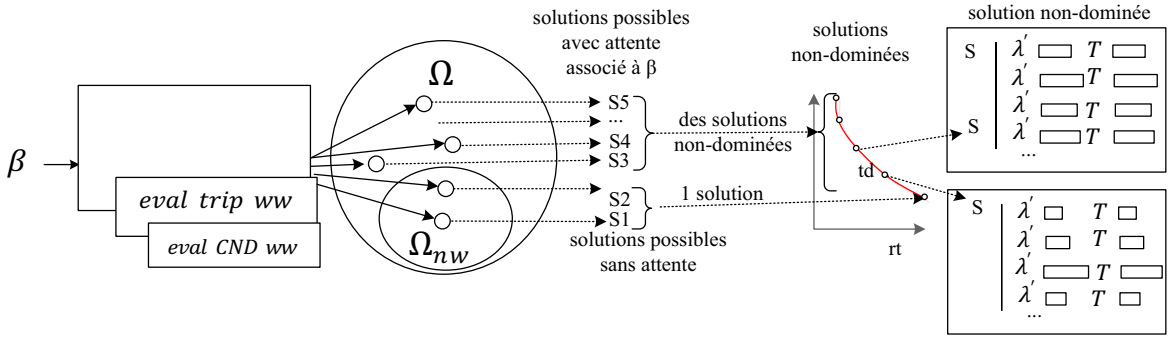


Figure 3.47 : Evaluation d'un élément β en autorisant l'attente avec la procédure SPLIT

La méthode split a déjà été adaptée pour évaluer les solutions sans attente. Elle est généralisée

au cas avec attente. Pour cela il faut prendre en compte les deux critères (td) et (rt). Il faut aussi prendre en compte que l'évaluation d'un vecteur λ donne un ensemble de tournées. Chacune de ces tournées a un label non-dominé. La fonction $split\_ww()$ utilise la fonction $eval\_trip\_ww$ qui permet l'évaluation d'une tournée. Elle est présentée dans la partie suivante et, comme illustré sur la figure 3.47, elle utilise la fonction d'évaluation des chemins non-dominés présentée dans la partie 3.3.3.

Comme pour le split classique, un graphe auxiliaire G_{split} est construit. Dans ce graphe, à chaque client de β est associé un sommet. Les arcs sont associés aux vecteurs λ et à ces arcs est associé un ensemble de coûts non–dominés qui correspondent au coût des tournées. Sur chacun de ces sommets est associé un ensemble de labels qui sont composés de trois critères: vu est le nombre de véhicules utilisés, x est la somme des temps de conduite, y est la somme des durées totales. Ces labels sont propagés de sommet en sommet grâce à une règle de propagation.

Règle de propagation :

Soit un arcs (i, j) dans le graphe du split auquel est associé un ensemble de labels E_i. Soit un ensemble A_i de coûts correspondant aux coûts des tournées non-dominées associées à l'arc (i,j). La règle de propagation est: pour chaque label $L(vu, x, y) \in E_i$ et chaque tournée $\in A_i$ de coût (td, rt), un label est créé en utilisant la formule suivante :

$$L'(vu + 1\ ;\ y + rt;\ x + td)$$

Une fois les labels générés, seuls les labels qui ne sont pas dominés sont conservés. La règle de dominance avec 3 critères est alors appliquée. Cette règle est définie dans le paragraphe 3.2.4 (définition 3.1). La figure 3.48 représente la propagation des labels dans le graphe G_{split} entre le sommet i et le sommet k.

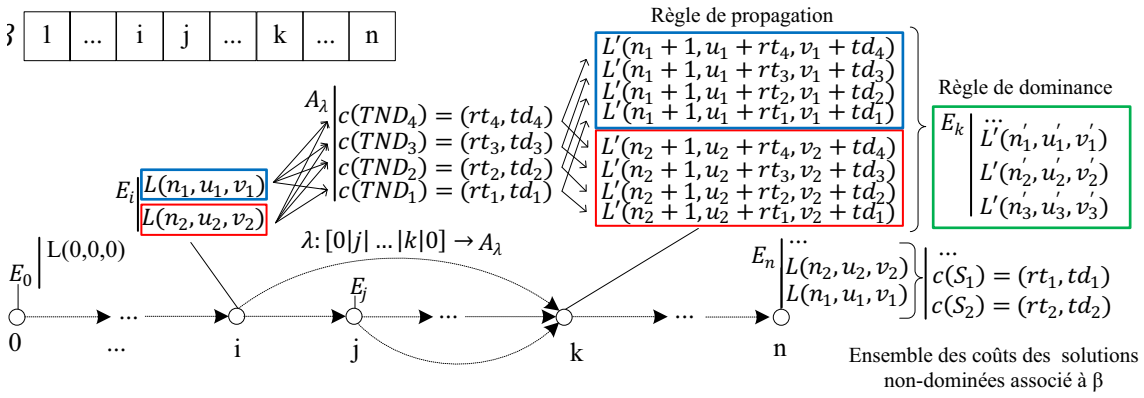


Figure 3.48: Schéma du split pour le TDVRP avec attente

L'algorithme split reste le même et seules les procédures qui le composent diffèrent :

- $eval\_trip\_nw()$ est remplacée par la fonction $eval\_trip\_ww()$: cette fonction est définie dans la partie suivante et est associée à l'évaluation d'un λ. Elle retourne le nombre de tournées non-dominées obtenues ;
- $Propagate\_split()$ est remplacée par la fonction $Propagate\_split\_ww()$: cette fonction propage les labels vers le sommet suivant en respectant la règle de propagation et la règle de dominance des labels qui a été présentée dans cette partie.

```
15 | | C_λ:= eval_trip_nw(λ)                15 | | A_λ:= eval_trip_ww(λ)
16 | | if (C_λ > 0) then                    16 | | if (|A_λ| > 0) then
17 | | | Propagate_split (E_i, C_λ, E_j)    17 | | | Propagate_split_ww (E_i, A_λ, E_j)
18 | | end if                               18 | | end if
```

Figure 3.49: Modification dans le code du split présenté dans l'algorithme 3 de la partie 3.5.2

Les labels associés à un sommet k du graphe G_{split} sont indépendants des sommets suivants. Donc si deux éléments $\beta 1$ et $\beta 2$ sont identiques jusqu'au $k^{ème}$ sommet, alors entre les deux évaluations, les éléments sur les sommets de 0 à k dans le graphe G_{split} peuvent être conservés entre les deux évaluations. Lors de l'évaluation des éléments β, ceci permet de réduire le temps de calcul.

5. Evaluation d'un vecteur λ associé à une tournée

L'évaluation d'un vecteur λ est réalisée par la procédure $eval\_trip\_ww(\lambda, E)$. La principale différence par rapport au cas sans attente réside dans le fait que, pour une même date de départ $start$, le véhicule peut générer un ensemble de labels $E^{i}_{|\lambda|}$ (figure 3.50). Chacun de ces labels respecte l'ordre de traitement des clients mais est différent en termes de temps de conduite et de temps total associé à la tournée. Il est aussi différent en ce qui concerne les pauses et les chemins suivis par les chauffeurs entre les clients. Cette fonction fait appel à la fonction de calcul des chemins non-dominés avec attente qui a été présentée dans la partie 3.3.3.

Les sommets du vecteur λ représentent les sommets clients traités par le véhicule. Lorsque le véhicule arrive sur un sommet client, il doit attendre que les marchandises soient déchargées avant de pouvoir repartir. Sur un sommet k du vecteur λ, le temps de chargement est donc ajouté à tous les labels $L(rt, td)$ qui compose l'ensemble E_k. Comme le temps de chargement est compté dans (td) mais qu'il n'est pas considéré comme du temps de conduite, $L(rt, td)$ devient $L(rt, td + lt(k))$. L'ensemble E'_k qui correspond à E_k après l'ajout du temps de chargement est composé uniquement de labels non-dominés, puisque la même valeur est ajoutée à tous les labels.

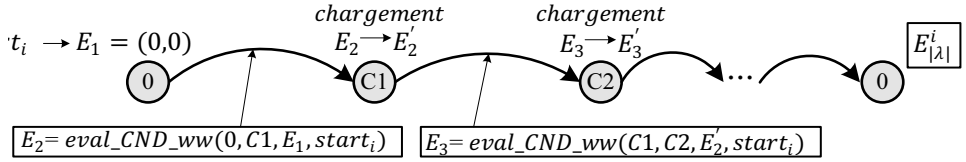


Figure 3.50: Evaluation d'un λ en autorisant l'attente

Comme pour l'évaluation sans attente, pour chaque date de début de travail possible $start_i$ avec $i \in \{1, \dots, nb_{start}\}$ le vecteur λ est évalué de manière indépendante. Une règle de dominance est appliquée sur le sommet final pour ne garder que les tournées de coûts non-dominés, comme illustré sur la figure 3.51. Cette règle de dominance concerne les deux critères (td) et (rt) associés à la tournée.

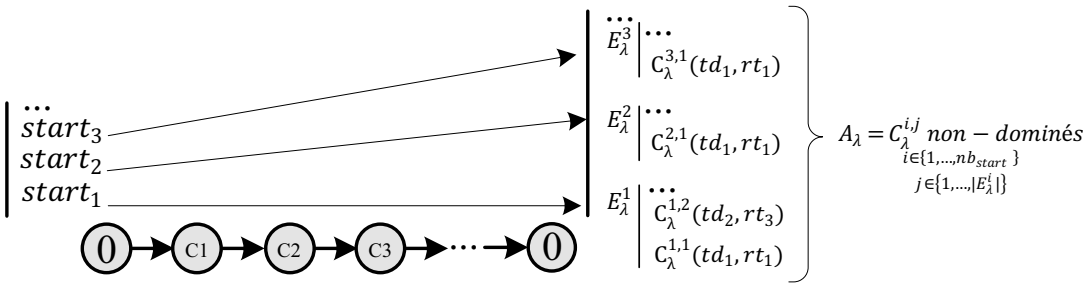


Figure 3.51: Evaluation d'un λ en autorisant l'attente

L'algorithme 5 représente le pseudo-code associé à la fonction. Il n'est pas nécessaire d'appliquer une règle de dominance sur chaque sommet puisqu'une règle de dominance est déjà appliquée par la fonction d'évaluation des plus courts chemins $eval\_CND\_ww$. Cette règle est détaillée dans le paragraphe 3.3.3.

Algorithme 5 eval_trip_ww

Input/Output parameters

λ : list of clients

Output parameters

A_λ : a set of trip non-dominated

Begin

for j **from** 1 **to** nb_start **do**
  $L \coloneqq (start_i, 0)$
  $add\_label\ (L, E_1)$
  $v \coloneqq \lambda(1)$ // depot_node
  $i := 2$
  $res \coloneqq true$
  while $(i \leq |\lambda|)$**AND**$(|E_{i-1}| > 0)$ **then**
    $u \coloneqq v$
    $v \coloneqq \lambda(i)$
    $C_{uv} \coloneqq\ eval\_CND\_ww(u, v, start_j, E_i^j,)$
    $Propagate\_trip\ (E_i^j, C_{uv}, E_j^j)$
    if $(i = |\lambda|)$ **then**
      $A_\lambda \coloneqq\ dominance(E_{|\lambda|})$
    end if
  end while
end for
return $\{A_\lambda\}$
End

3.6.3. La phase de recherche locale

La phase de recherche locale est exécutée après l'évaluation d'un élément β intermédiaire. A la fin l'évaluation, une population de solutions est générée. Comme présenté dans la partie précédente, ces solutions peuvent provenir de différents découpages de β.

La recherche locale doit s'appliquer sur une population de solutions. Chaque solution de la population est traitée successivement. L'objectif est de rechercher les solutions voisines dont les coûts dominent le coût de la solution courante. Comme elles dominent une solution courante du front, ces solutions améliorent la population. Comme illustré sur la figure 3.52, la recherche locale permet d'explorer un espace intéressant dans lequel au moins un des deux critères est amélioré et où l'autre n'est pas détérioré. Cette recherche locale est appliquée successivement à toutes les solutions du front.

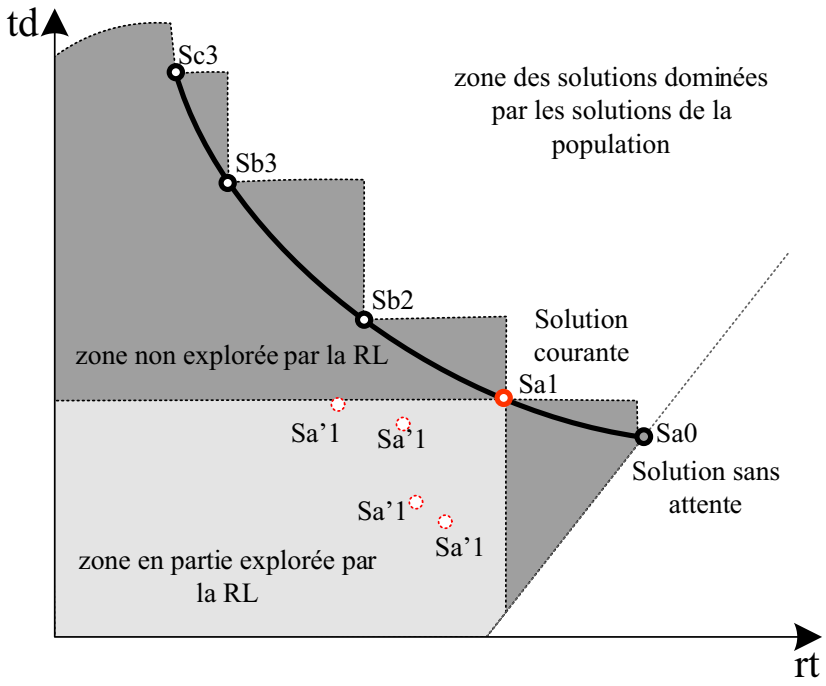


Figure 3.52 : Espace exploré par la recherche locale

Pour modifier les critères associés au coût d'une solution, il faut soit modifier l'attente soit modifier l'ordre de traitement des sommets. Comme la méthode d'évaluation de β peut être considérée comme optimale si le nombre de labels n'est limité dans la méthode split, alors il faut changer l'ordre de traitement des sommets pour améliorer la solution.

La recherche locale traite les tournées qui composent la solution courante successivement et seul des mouvements intratournées sont considérés. L'idée de la recherche locale consiste à réutiliser la recherche locale pour le problème sans attente en l'appliquant sur les parties de la tournée sans attente. A cause de la durée minimale imposée aux pauses et de la contrainte sur la durée totale de la tournée, chaque tournée n'est composée que d'un nombre fini de pauses. Entre deux pauses consécutives, comme illustré sur la figure 3.53, les fonctions d'évaluation de tournée et de recherche locale développée pour la partie TDVRP-nw peuvent être réutilisées.

Par exemple dans la figure 3.53, les pauses de la tournée se situent sur le client 1 et sur le client 4. La recherche local est appliquée sur la sous-tournée $\lambda_{fictif} = [1234]$. Dans cette sous-tournée, 1 et 4 sont considérés comme des dépôts. Une date unique de départ est possible et elle correspond à la date de départ du sommet 1 dans la tournée initiale. Si la recherche locale sur λ_{fictif} sans attente génère une meilleure solution, c'est-à-dire une solution avec un temps de conduite et une durée totale inférieure, alors cette nouvelle sous-tournée peut être intégrée dans la tournée courante comme illustré dans la figure 3.54. L'attente sur le sommet 4 est alors augmentée afin de ne pas modifier la date de départ du véhicule et la fin de la tournée reste identique. L'augmentation du temps de pause correspond à la diminution du temps de trajet, dans l'exemple cela correspond à 6 minutes. Comme le temps de conduite de la sous-tournée a diminué alors la nouvelle tournée à rt inférieur au rt précédant, la valeur totale de la tournée reste inchangée puisque la fin de la tournée reste inchangé. La solution générée domine donc la solution courante.

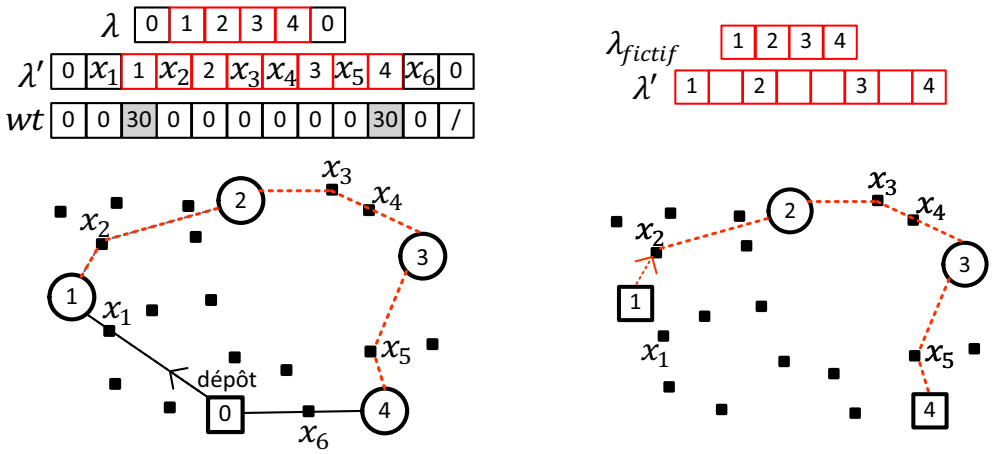


Figure 3.53: Recherche locale appliquée à une solution avec attente

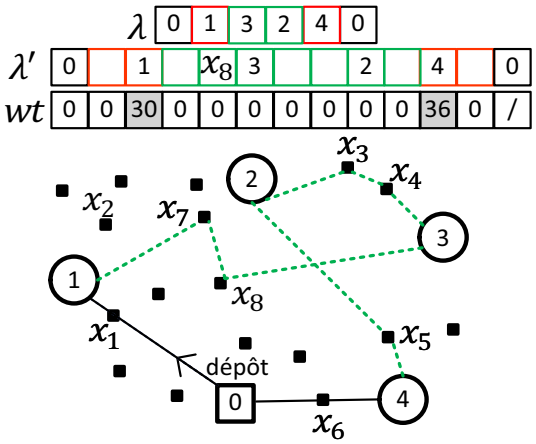


Figure 3.54: Reconstruction de la tournée si une amélioration est obtenue

3.6.4. Création de solutions initiales

Une population initiale de solutions est obtenue en évaluant l'ensemble des β obtenues à la fin de la première phase avec la fonction $split\_ww()$ et en conservant uniquement les solutions non-dominées. Comme illustré dans la figure 3.55, à chaque tour géant β est obtenu un ensemble de solutions non-dominées représentées dans la figure sur un front. Parmi l'ensemble de ces solutions, les solutions non-dominées forment un nouveau front de Pareto. Ce front est utilisé comme population initiale.

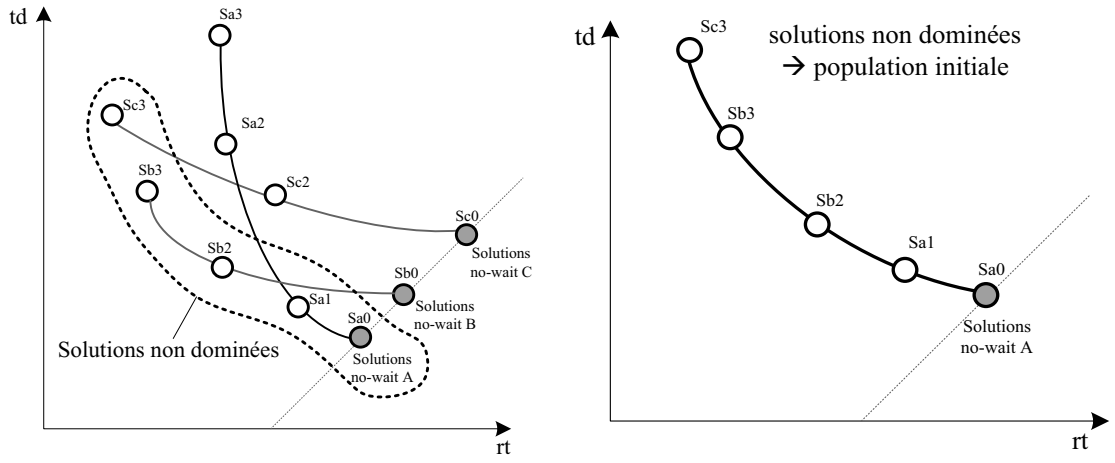


Figure 3.55 : Evaluation des β associés aux solutions sans attente et construction de la population initiale

3.7. Les résultats

3.7.1. Résultats sur les instances classiques du VRP

Le problème du TDVRP est une généralisation du problème du VRP dans lequel la vitesse du véhicule sur les arcs est constante et les plus courts chemins entre les clients sont connus. La première phase de la méthode concerne les solutions sans attente. Elle a été utilisée pour traiter 14 instances de la littérature du VRP proposées par (Christofides and Eilon, 1969) et (Christofides, 1979)

Les résultats obtenus ont été comparés avec 4 méthodes de la littérature :

- celle proposée par (Mester and Bräysy, 2007) ;
- celle proposée par (Nagata and Bräysy, 2009) ;
- celle proposée par (Vidal et al., 2014) ;
- celle proposée par (Lacomme et al., 2015).

Pour pouvoir comparer les temps de calcul entre ces méthodes le nombre de Mflops associé aux différentes processeurs a été obtenu en utilisant le LINPACK publié dans (Dongarra et al., 1979).

Les résultats obtenus par les différentes méthodes sont détaillés dans le tableau 3.2. Ce tableau se compose de deux colonnes qui décrivent l'instance avec : n le nombre de clients et BKS la meilleure solution connue. Elles sont suivies de 5 groupes associés à chacune des méthodes. Ces groupes se composent de 3 colonnes qui sont : GAP l'écart en pourcentage entre la BKS et la valeur de la solution moyenne obtenue, T le temps d'exécution total non normalisé et T' le temps d'exécution total normalisé.

Tableau 3.1:
Performance relative des ordinateurs utilisés dans la littérature

	Mester and Braysy (2007)	**Nagata and Braysy(2009)**	**(Jozefowiez et al.,2009)**	**Vidal et al. (2014)**	**Lacomme et al. (2015)**	**Chassaing et al. (2015)**
Processor	PIV 2.8GHz	Opteron 2.4GHz	RS 6000 1.1 GHz 8 processors	Opteron 2.4GHz	Intel Xeon 2.40 GHz	Intel® Core™ i7-3770 CPU @ 3.40GHz
OS	/	Linux	/	/	Linux	Windows 7
Lang.	/	C++	C++	C++	C	C++
MFlops	1850	3350	750	3350	3800	2500
Speed factor	0.7	1.3	0.3	1.3	1.5	1

Dans la littérature, le VRP est très bien traité et les méthodes dédiées au VRP sont à moins de 0,25% de la solution optimale en moins de 15 min. La méthode proposée par (Lacomme et al., 2015) n'est pas dédiée au VRP et les résultats sont en moyenne à 2,81% de la solution optimale avec un temps de calcul normalisé d'environ 18.9 min. La méthode proposée ici n'est pas non plus dédiée au VRP et donne des solutions à 2,33% de l'optimal avec un temps d'exécution de l'ordre de 25 min. Elle ne rivalise donc pas avec les méthodes dédiées au VRP, par contre elle est comparable avec la méthode non dédiée de (Lacomme et al., 2015).

Tableau 3.2:
Performance sur les instances du VRP

			Mester and Braysy (2007)			Nagata and Braysy (2009)			Vidal et al. (2014)			Lacomme et al. (2015)			la méthode proposée		
	n	BKS	GAP	T	T'	GAP	T	T'	GAP	T	T'	GAP	T	T'	GAP	T	T'
1	50	524.61	0.00	0.0	0.0	0.00	0.1	0.1	0.00	1.9	2.5	0.00	0.5	0.8	0.00	5.3	5.3
2	75	835.26	0.00	0.1	0.1	0.04	0.4	0.5	0.00	4.6	6.0	1.76	2.6	3.9	1.32	9.1	9.1
3	100	826.14	0.00	0.0	0.0	0.00	0.3	0.4	0.00	9.3	12.1	0.41	5.3	8.0	1.24	18.8	18.8
4	150	1028.14	0.00	0.2	0.1	0.00	1.3	1.7	0.31	16.9	22.0	2.45	13.9	20.9	2.99	32.7	32.7
5	199	1291.45	0.00	36.0	25.2	0.04	5.0	6.5	0.93	24.9	32.4	4.76	26.5	39.8	4.88	48.4	48.4
11	120	1042.11	0.00	0.0	0.0	0.00	0.4	0.5	0.00	11.6	15.1	12.41	10.6	15.9	0.21	32.3	32.3
12	100	819.56	0.00	0.0	0.0	0.00	0.1	0.1	0.00	4.5	5.9	0.00	5.5	8.3	0.00	15.2	15.2
6	50	555.43	0.00	0.1	0.1	0.00	0.1	0.1	0.00	1.9	2.5	0.18	1.2	1.8	0.44	6.3	6.3
7	75	909.68	0.00	0.1	0.1	0.08	0.6	0.8	0.00	3.5	4.6	1.21	3.3	5.0	3.04	12.6	12.6
8	100	865.94	0.00	0.0	0.0	0.00	0.4	0.5	0.00	6.3	8.2	0.43	7.2	10.8	1.36	20.9	20.9
9	150	1162.55	0.00	0.4	0.3	0.00	2.3	3.0	0.13	15.6	20.3	2.09	18.9	28.4	4.49	43.6	43.6
10	199	1395.85	0.38	0.9	0.6	0.18	6.5	8.5	1.22	24.8	32.2	4.92	49.4	74.1	6.33	63.2	63.2
13	120	1541.14	0.00	0.2	0.1	0.12	1.8	2.3	0.43	12.2	15.9	7.71	22.8	34.2	0.64	37.8	37.8
14	100	866.37	0.00	0.0	0.0	0.00	0.2	0.3	0.00	4.8	6.2	0.96	9.1	13.7	5.72	16.1	16.1
	Average time			2.7			1.4			10.2			12.6			25.9	
	Average scale time				1.9			1.8			13.3			18.9			25.9
	Avg. Gap		0.03			0.03			0.22			2.81			2.33		

3.7.2. Construction des instances

Un ensemble d'instances correspondant au modèle proposé a ensuite été construit. L'objectif est de créer différentes instances dans lesquelles les plus courts chemins évoluent en fonction du temps, comme illustré dans la figure 3.56. Sur la figure, les sommets en rouge représentent des clients, les autres sommets sont des sommets intermédiaires. Sur la figure, les arcs en rouge représentent le plus court chemin en temps entre le sommet 2 et le sommet 39 pour deux dates de départs différentes. Les dates de départ sur chaque sommet sont représentées dans un cadre noir. La figure à gauche représente le plus court chemin avec une date de départ à 0 et à droite avec un départ à la date 660.

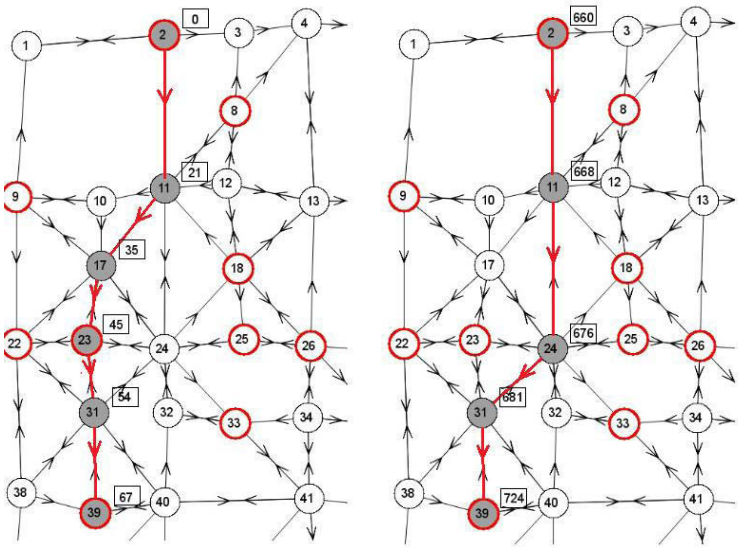


Figure 3.56 : Plus court chemin entre le sommet 2 et le sommet 39 avec un départ à la date 0 et à 660

Les instances ont été générées à partir d'instances publiées dans la littérature pour un problème de localisation et routage (LRP – *Location Routing Problem)*. Les instances ont été choisies car elles simulent un réseau routier et forment un graphe planaire asymétrique. Ainsi, la distance du sommet i au sommet j est différente de la distance du sommet j au sommet i. A partir des 11 instances nommées "DLPP", 33 instances du TDVRP ont été générées. Les instances du LRP sont disponibles sur http://www.isima.fr/~lacomme/lrp/lrp.html. Les instances générées sont composées de 110 à 341 sommets, avec un nombre de clients qui varie de 50 à 164 et un nombre d'arcs variant de 350 à 1124. Les caractéristiques de ces instances sont détaillées dans le tableau 3.3. Les instances et les solutions détaillées de chaque exécution sont téléchargeables depuis http://fc.isima.fr/~chassain/tdvrp/download.php.

Pour générer les instances de TDVRP, il faut affecter des vitesses pour chacune des périodes à chaque arc. Pour définir ces vitesses, plusieurs méthodes ont été utilisées. Chacune de ces méthodes correspond à une catégorie d'instances :

- catégorie 1, les vitesses sont choisies aléatoirement entre 30 et 90 km/h ;
- catégorie 2, les vitesses sont choisies aléatoirement entre 50 et 90 km/h ;
- catégorie 3, les vitesses sont choisies aléatoirement entre 70 et 90 km/h puis le graphe est découpé en différentes zones qui subissent des ralentissements. Cette catégorie est expliquée dans le paragraphe suivant.

Dans cette troisième catégorie, chaque instance est découpée en 4 zones différentes. Ces zones représentent des quartiers d'une ville. Les arcs d'une zone ou entre des sommets de deux zones identiques ont des comportements similaires dans le temps. L'objectif est de simuler les mouvements de population dans une ville correspondant aux heures de bureaux par exemple.

Tableau 3.3:
caractéristiques des instances utilisé pour générer des instances de TDVRP

nom de l'instance	nb. sommets	nb. arcs	nb. clients	nom pour la catégorie 1	nom pour la catégorie 2	nom pour la catégorie 3
DLP_110	110	350	49	DLP_110_1	DLP_110_2	DLP_110_3
DLP_126	126	403	59	DLP_126_1	DLP_126_2	DLP_126_3
DLP_130	130	401	68	DLP_130_1	DLP_130_2	DLP_130_3
DLP_137	137	433	55	DLP_137_1	DLP_137_2	DLP_137_3
DLP_170	170	534	72	DLP_170_1	DLP_170_2	DLP_170_3
DLP_210	210	659	100	DLP_210_1	DLP_210_2	DLP_210_3
DLP_221	221	720	90	DLP_221_1	DLP_221_2	DLP_221_3
DLP_224	224	710	96	DLP_224_1	DLP_224_2	DLP_224_3
DLP_260	260	863	113	DLP_260_1	DLP_260_2	DLP_260_3
DLP_285	286	907	130	DLP_285_1	DLP_285_2	DLP_285_3
DLP_341	341	1124	164	DLP_341_1	DLP_341_2	DLP_341_3

La figure 3.57 illustre ces différentes zones pour l'instance DLP_110_3. Chaque couleur représente un quartier et l'épaisseur des arcs représente le niveau de ralentissement qu'il subit. Plus l'arc est gras, plus la vitesse est faible. Sur l'image de gauche qui concerne la première période de l'instance, les zones en périphérie sont congestionnées. Elles représentent des zones résidentielles. A droite, une autre période est présentée dans laquelle la zone bleue est

congestionnée. Cette zone représente le centre-ville. Dans cette seconde représentation, des arcs reliant la zone violette à la zone rouge sont aussi fortement encombrés.

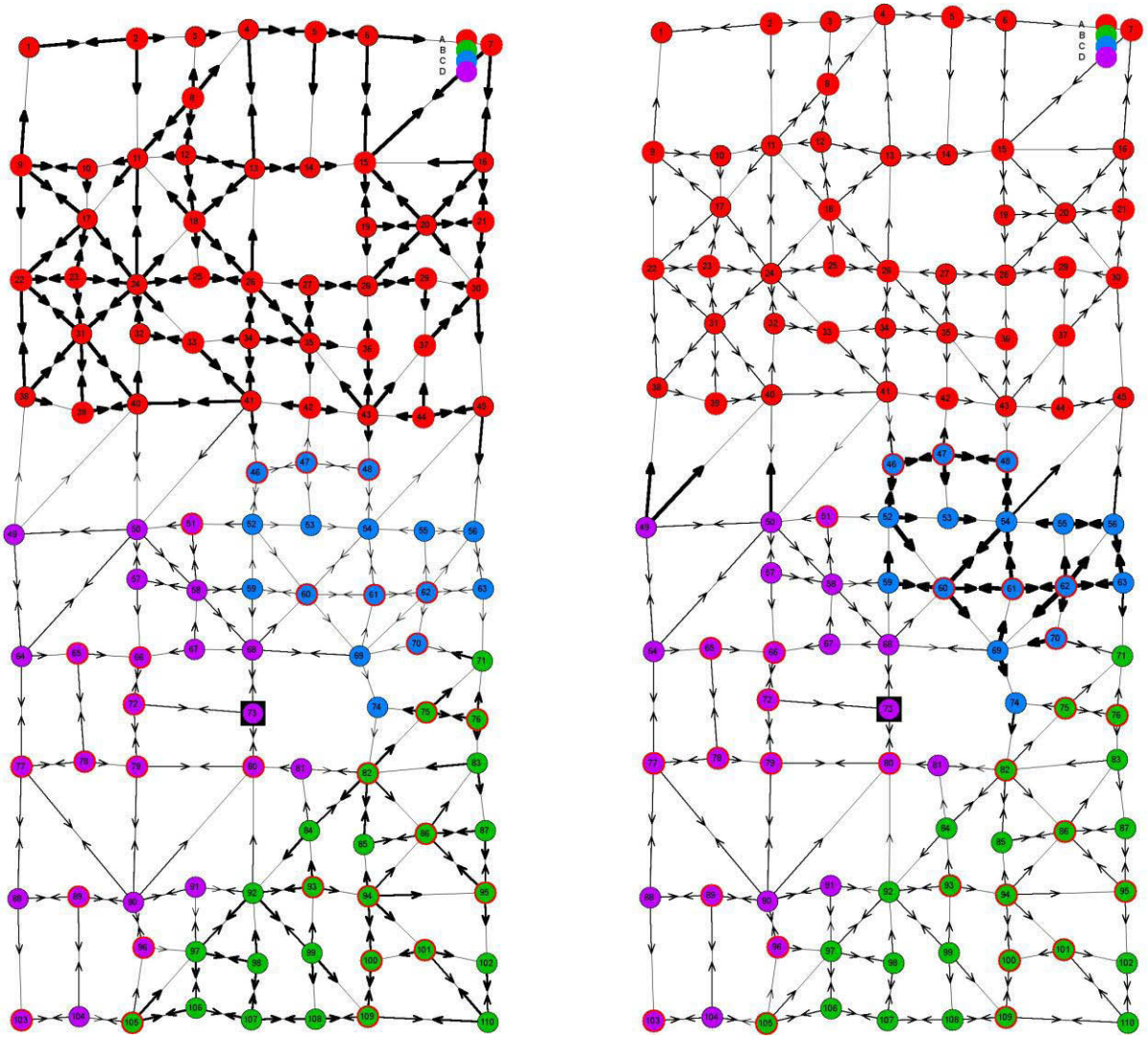


Figure 3.57 : Répartition en zone pour l'instance DLP_110_3 et évolution des temps de trajet au cours du temps

Les vitesses entre les arcs sont choisies aléatoirement entre 70 et 90 km/h puis une pondération est appliquée en fonction de l'encombrement associé au type d'arc. Par exemple la vitesse est divisée par 4 dans le centre-ville pour la période représentée dans la figure 3.57 (à gauche). Elle varie donc entre 17,5 km/h et 22,5 km/h. L'ensemble de ces instances est téléchargeable sur http://fc.isima.fr/~chassain/tdvrp/download.php.

3.7.3. Paramètres associés à la méthode de résolution.

Les paramètres de la méthode sont fixés de manière statique. Dans la première phase où les véhicules ne sont pas autorisés à attendre, 50 solutions initiales sont générées. Un ELS est appliqué sur chacune de ces solutions. Son critère d'arrêt est défini comme 5 itérations consécutives sans améliorer la meilleure solution record courante. Dans chaque itération 5

voisins sont générés et durant la phase de recherche locale une limite de 1000 permutations est fixée.

Dans la seconde phase, des Path Relinking sont réalisés entre la solution dont le (rt) est le plus faible et les autres solutions du front. Une fois ces évaluations terminées la population est mise à jour et le processus est répété 5 fois. Concernant le la procédure de calcul du plus court chemin dans le cas avec attente, le nombre de labels maximaux par sommet est limité à 50. Ce choix a été fait pour limiter le temps d'exécution de la méthode. Lorsque le nombre de labels sur un sommet dépasse cette limite, le label avec la durée totale la plus élevée est supprimé. La recherche locale est également limitée à 1000 permutations.

3.7.4. Les résultats obtenus

Le programme a été exécuté 5 fois pour chaque instance et à chaque exécution l'algorithme retourne un ensemble de solutions non-dominées qui peuvent être représentées par un front de Pareto comme illustré sur la figure 3.58. Une partie des caractéristiques des solutions qui le composent est notée dans le tableau 3.4. Ces solutions sont caractérisées par un temps de conduite rt et une durée totale td. La colonne lt représente le temps de chargement nécessaire pour servir la totalité des clients. Il est le même pour toutes les solutions de l'instance puisque les mêmes clients sont servis. Enfin wt représente la somme des temps d'attente des tournées. Il peut être calculé avec la formule : $wt = td - rt - lt$.

Tableau 3.4: ensemble de solution pour l'instance DLP_224_1 pour la 1ère l'exécution

	rd	td	lt	wt
s1	1398	3421	1440	583
s2	1399	3413	1440	574
s3	1401	3296	1440	455
s4	1402	3259	1440	417
…	…	…	…	…
s15	1417	2909	1440	52
s16	1418	2898	1440	40
s17	1419	2897	1440	38
s18	1420	2860	1440	0

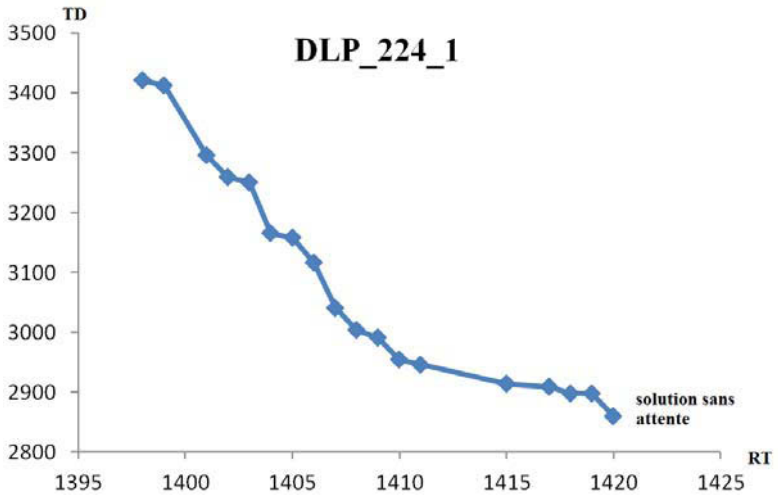


Figure 3.58 : Solutions non-dominées pour l'instance DLP_224_1 obtenue lors de la première exécution

Les résultats obtenus pour toutes les instances sont résumés dans les tableaux suivants (ils sont obtenus après 5 exécutions). L'hypervolume I_H a été choisi comme indicateur de qualité du front.

Les bornes $max(td)$ et $max(rt)$, nécessaires pour calculer I_H, sont obtenues en prenant les valeurs maximales de rt et td parmi toutes les solutions retournées par les 5 exécutions. En plus de cet indice, on a indiqué les solutions extrêmes du front, qui sont les solutions qui minimisent pour chaque instance le rt et le td.

Chaque tableau représente une catégorie d'instances. La catégorie 1 est dans le tableau 3.5, la catégorie 2 dans le tableau 3.6 et la catégorie 3 dans le tableau 3.7. Ces tableaux sont composés des mêmes colonnes.

De gauche à droite: "Instance" contient le nom de l'instance ; RT^M et TD^M représentent les bornes associées au calcul de l'indice hypervolume ; I_H^* est la mesure de l'hypervolume minimal obtenu parmi les 5 exécutions ; $\overline{I_H}$ est l'indice moyen ; les 4 colonnes suivantes concernent les solutions des fronts qui minimisent la durée totale (*i.e.* les solutions sans attentes obtenues) et sont détaillées dans le paragraphe suivant; les 4 colonnes suivantes concernent les solutions des fronts qui minimisent le temps de conduite rt ; LT contient le temps de chargement nécessaire pour servir les clients ; $\overline{n}$ est le nombre moyen de solutions contenues dans le premier front de Pareto ; $\overline{T1}$ est le temps d'exécution moyen de la méthode pour la première phase du GRASPxELS ; $\overline{T2}$ est le temps d'exécution moyen de la méthode pour la seconde phase du PR. Les temps de calcul sont obtenus sur la machine présentée dans le début du paragraphe 3.7.

Tableau 3.5:
résultats sur les instances TDVRP de catégorie 1

	bornes		hypervolume		solution sans attente				solution avec (rt) minimal							
Instance	$\mathbf{rt^M}$	**td**	$\mathbf{I_H^*}$	$\overline{\mathbf{I_H}}$	rt	td^*	$\overline{rt}$	$\overline{td^*}$	rt^*	td	$\overline{rt^*}$	$\overline{td}$	**lt**	$\overline{\mathbf{n}}$	$\overline{\mathbf{T1}}$	$\overline{\mathbf{T2}}$
DLP_110_1	579	1999	1.13	1.13	565	1300	571.2	1306.2	520	1931	524.0	1935.6	735	35.4	17.8	5.6
DLP_126_1	701	2061	1.42	1.43	679	1564	689.4	1574.4	650	2061	662.6	1999.0	885	18	26.3	7.2
DLP_130_1	798	2911	2.24	2.27	764	1784	784	1804.0	716	2487	740.8	2519.2	1020	29.6	25.6	8.0
DLP_137_1	848	2550	2.12	2.14	821	1646	831.4	1656.4	791	2550	816.4	2212.4	825	10.4	26.9	7.2
DLP_170_1	1039	3093	3.14	3.16	985	2065	1012.6	2092.6	959	2556	983.0	2747.6	1080	20.4	45.3	8.8
DLP_210_1	1365	3937	5.24	5.29	1287	2787	1327.6	2827.6	1243	3690	1281.6	3715.8	1500	33.8	22.7	11.6
DLP_221_1	1385	3487	4.74	4.77	1313	2663	1346.8	2696.8	1253	3456	1295.4	3428.8	1350	32.8	35.4	12.0
DLP_224_1	1520	3641	5.44	5.48	1420	2860	1480.4	2920.4	1398	3421	1431.6	3525.4	1440	33.4	41.7	18.4
DLP_260_1	1991	4400	8.46	8.54	1733	3428	1803.4	3498.4	1655	4178	1742.0	4258.6	1695	40.2	38.5	17.6
DLP_285_1	2299	5364	11.99	12.19	2091	4041	2229.4	4179.4	2030	5250	2172.0	5161.0	1950	40.6	39.6	17.6
DLP_341_1	3007	6420	19.09	19.19	2857	5317	2934.8	5394.8	2798	6217	2888.0	6171.6	2460	34.8	38.5	25.6
moyenne														29.9	32.5	12.7

Concernant les quatre colonnes représentant les solutions extrêmes du front, elles sont composées pour chaque instance de :

- rt et td^* qui correspondent aux deux critères de la solution qui minimise td dans le front pour les 5 exécutions ;
- $\overline{rt}$ et $\overline{td^*}$ qui correspondent aux valeurs moyennes (pour les 5 exécutions) des deux critères concernant la solution qui minimise td ;
- rt^* et td qui correspondent aux deux critères de la solution qui minimise rt dans le front pour les 5 exécutions ;
- $\overline{rt^*}$ et $\overline{td}$ qui correspondent aux valeurs moyennes (pour les 5 exécutions) des deux critères concernant la solution qui minimise rt.

Dans les tableaux, le nombre de solutions contenues dans le front varie en fonction de la catégorie d'instance. Pour les premières et deuxièmes catégories, le nombre moyen de solutions est 29.9 et 24.1 contrairement à la troisième catégorie qui en contient seulement 10.

Tableau 3.6:
résultats sur les instances TDVRP de catégorie 2

	bornes		hypervolume		solution sans attente				solution avec (rt) minimal							
Instance	$\mathbf{rt^M}$	**td**	$\mathbf{I_H^*}$	$\overline{\mathbf{I_H}}$	rt	td^*	$\overline{rt}$	$\overline{td}$	rt^*	td	$\overline{rt}$	$\overline{td}$	**lt**	$\overline{\mathbf{n}}$	$\overline{\mathbf{T1}}$	$\overline{\mathbf{T2}}$
DLP_110_2	479	1943	0.91	0.92	468	1203	472.2	1207.2	448	1943	453.4	1884.0	735	19	16.5	4.8
DLP_126_2	554	2009	1.10	1.10	547	1432	549.8	1434.8	530	1999	532.2	1959.2	885	17	20.9	5.6
DLP_130_2	630	2400	1.47	1.49	595	1615	613.8	1633.8	574	2400	597.8	2193.0	1020	13.8	22.5	5.6
DLP_137_2	692	2051	1.41	1.41	679	1504	685.8	1510.8	664	2035	671.0	1946.8	825	12	24.9	6.0
DLP_170_2	851	2864	2.39	2.41	822	1902	839.4	1919.4	802	2446	821.4	2458.2	1080	17	35.2	8.0
DLP_210_2	1208	3461	4.06	4.13	1087	2587	1159.6	2659.6	1063	3380	1134.8	3350.8	1500	24	23.9	8.4
DLP_221_2	1067	3559	3.72	3.76	1026	2376	1054.8	2404.8	996	3460	1028.4	3414.4	1350	25.2	37.8	8.4
DLP_224_2	1191	3515	4.13	4.15	1146	2586	1172.6	2612.6	1122	3410	1146.0	3375.6	1440	23.8	49.2	10.0
DLP_260_2	1400	4109	5.59	5.70	1282	2977	1371.6	3066.6	1237	4109	1336.8	4015.6	1695	30.4	52.6	11.2
DLP_285_2	1771	4787	8.41	8.43	1730	3680	1750.2	3700.2	1692	4787	1712.4	4575.0	1950	35.6	40	11.6
DLP_341_2	2366	6161	14.37	14.47	2260	4720	2325.0	4785.0	2204	6161	2273.2	5920.8	2460	47	41.3	17.6
moyenne														24.1	33.2	8.8

Tableau 3.7:
résultats sur les instances TDVRP de catégorie 3

	bornes		hypervolume		solution sans attente				solution avec (rt) minimal							
Instance	$\mathbf{rt^M}$	**td**	$\mathbf{I_H^*}$	$\overline{\mathbf{I_H}}$	rt	td^*	$\overline{rt}$	$\overline{td}$	rt^*	td	$\overline{rt}$	$\overline{td}$	**lt**	$\overline{\mathbf{n}}$	$\overline{\mathbf{T1}}$	$\overline{\mathbf{T2}}$
DLP_110_3	410	1767	0.72	0.72	406	1141	408.0	1143.0	399	1767	401.4	1657.2	735	7.6	13.7	4.0
DLP_126_3	476	1876	0.89	0.89	474	1359	475.2	1360.2	470	1615	470.8	1752.4	885	5.4	19	4.8
DLP_130_3	553	2640	1.43	1.44	534	1554	545.2	1565.2	526	2181	535.4	2231.2	1020	10.4	20.6	4.8
DLP_137_3	588	1882	1.10	1.10	575	1400	580.8	1405.8	571	1733	576.8	1778.8	825	5	20.7	4.8
DLP_170_3	717	2542	1.80	1.81	690	1770	704.8	1784.8	683	2315	698.0	2287.8	1080	7.6	38.5	6.0
DLP_210_3	1067	3444	3.55	3.60	950	2450	993.4	2493.4	939	3074	982.0	3164.2	1500	12.2	23.9	5.6
DLP_221_3	938	2913	2.70	2.71	888	2238	911.2	2261.2	881	2811	902.2	2815.6	1350	10	29.6	8.8
DLP_224_3	1041	2957	3.05	3.06	994	2434	1017.6	2457.6	987	2695	1009.8	2868.8	1440	8.6	45.5	7.6
DLP_260_3	1235	3777	4.53	4.60	1106	2801	1176.4	2871.4	1093	3777	1165.2	3513.2	1695	11.8	55.2	6.8
DLP_285_3	1496	4545	6.74	6.76	1455	3405	1472.2	3422.2	1441	4444	1458.6	4388.8	1950	14.2	36.5	8.4
DLP_341_3	2013	5619	11.24	11.25	1962	4422	1977.8	4437.8	1940	5607	1957.8	5435.4	2460	20.2	30.2	8.0
moyenne														10.3	30.31	6.3

Le tableau 3.8 contient la différence entre les temps de conduite pour des solutions extrêmes moyennes des fronts de Pareto obtenus sur les 5 exécutions de la méthode. Le tableau est composé de quatre parties. La première est le nom de l'instance à laquelle doit être ajoutée une catégorie (exemple: DLP_110 correspond aux instances DLP_110_1, DLP_110_2 et DLP_110_3). Ensuite, pour chacune des catégories, 3 colonnes sont associées : m le nombre de véhicules associés à l'instance, $dim.\overline{rt}$ la différence de temps de conduite des solutions

extrêmes moyennes ; % le pourcentage auquel correspond le $dim.\overline{rt}$ par rapport à la solution sans attente moyenne, $aug.\overline{td}$ l'attente moyenne introduire dans les solutions pour obtenir le gain maximal sur le temps de conduite entre les solutions extrêmes moyennes comme illustré dans la figure 3.59

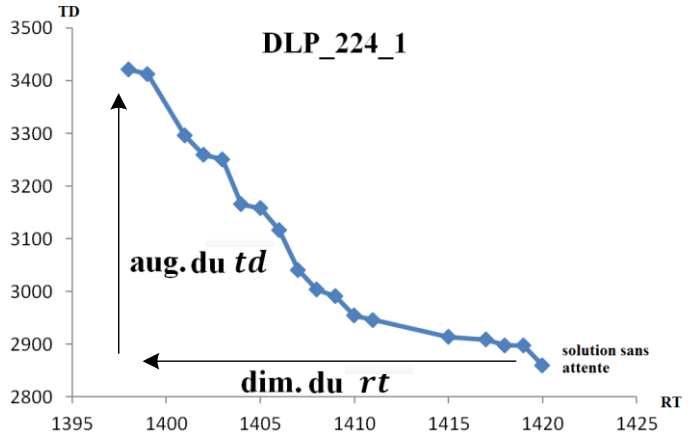


Figure 3.59 : Illustration des différences entre les solutions extrêmes du front

Premièrement, dans le tableau la diminution du temps de conduite réalisé est assez faible avec un gain maximal de 8% pour l'instance DLP_110_1 ce qui correspond à une diminution du temps de conduite de 47 min. De plus, pour obtenir cette diminution, l'augmentation engendrée sur le temps durée totale de la tournée est très élevée. Par exemple pour l'instance DLP_110_1 elle est de 676 min soit 170 min de pause moyenne par véhicule.

Tableau 3.8:
les différences entre les solutions extrêmes du front

instances	**catégorie 1**				**catégorie 2**				**catégorie 3**			
	m	**dim. du $\overline{rt}$**	**%**	**aug. du $\overline{td}$**	**m**	**dim. du $\overline{rt}$**	**%**	**aug. du $\overline{td}$**	**m**	**dim. du $\overline{rt}$**	**%**	**aug. du $\overline{td}$**
DLP_110	4	47.2	8.3	676.6	4	18.8	4.0	695.6	4	6.6	1.6	520.8
DLP_126	4	26.8	3.9	451.4	4	17.6	3.2	542.0	4	4.4	0.9	396.6
DLP_130	6	43.2	5.5	758.4	6	16.0	2.6	575.2	6	9.8	1.8	675.8
DLP_137	5	15.0	1.8	571.0	5	14.8	2.2	450.8	4	4.0	0.7	377.0
DLP_170	6	29.6	2.9	684.6	6	18.0	2.1	556.8	6	6.8	1.0	509.8
DLP_210	8	46.0	3.5	934.2	8	24.8	2.1	716.0	8	11.4	1.1	682.2
DLP_221	7	51.4	3.8	783.4	7	26.4	2.5	1036.0	7	9.0	1.0	563.4
DLP_224	7	48.8	3.3	653.8	7	26.6	2.3	789.6	7	7.8	0.8	419.0
DLP_260	9	61.4	3.4	821.6	9	34.8	2.5	983.8	8	11.2	1.0	653.0
DLP_285	12	57.4	2.6	1039.0	12	37.8	2.2	912.6	10	13.6	0.9	980.2
DLP_341	14	46.8	1.6	823.6	14	51.8	2.2	1187.6	13	20.0	1.0	1 017.6
moyenne		43.1	3.7	745.2		26.1	2.5	767.8		9.5	1.1	617.8

Les gains sur le temps de conduite obtenus pour les différentes catégories sont très différents. Les gains les plus importants sont réalisés pour les instances de la première catégorie avec une diminution moyenne du temps de trajet de 3,7%, puis la deuxième catégorie avec un gain de

2.5% et enfin la 3ème catégorie avec seulement 1.1%. L'introduction d'attente dans cette troisème catégorie n'apporte qu'un gain minime sur le temps de conduite des chauffeurs. Par exemple pour l'instance DLP_110_3, en moyenne le gain est inférieur à 7 min. Ce gain moins important pour la troisième catégorie provient surement du fait que les fortes fluctuations entre les arcs se situent seulement entre les différentes zones, contrairement aux autres catégories ou la fluctuation des vitesses entre les arcs est indépendante.

Pour l'entreprise qui gère la flotte de véhicules, ces solutions ont un intérêt car toutes les tournées respectent les contraintes imposées. Les pauses qui sont ajoutées peuvent être en réalité des pauses imposées au véhicule par la législation. Une solution avec attente, même si elle n'économise que quelques minutes de temps de conduite, peut respecter la législation sur le temps de conduite, contrairement à la solution sans attente. Si des pauses sont introduites dans la solution sans attente pour respecter la législation, alors le temps de conduite peut être augmenté et la solution valide peut ne plus l'être. Les conséquences pour l'entreprise peuvent être alors importantes. Par exemple, si le chauffeur atteint son quota de temps de conduite durant la tournée, l'entreprise pourrait être amené à envoyer un autre chauffeur pour récupérer le véhicule ou encore, si cela n'est pas possible, pourrait être contrainte à payer une indemnité pour que le chauffeur passe la nuit dans le camion et surveille les marchandises en attendant de reprendre la route le lendemain.

3.8. Conclusion

Pour résoudre le problème de tournées de véhicules avec temps de trajet dépendant du temps (TDVRP) dans lequel les véhicules sont autorisés à attendre, une approche en deux étapes a été proposée. La première étape consiste à résoudre le TDVRP sans autoriser le véhicule à attendre (TDVRP-nw) avec une métaheuristique de type GRASP combinée à une métaheuristique ELS. Les 10 meilleures solutions obtenues à la fin de cette phase sont alors utilisées pour l'initialisation de la seconde étape qui traite le TDVRP en autorisant les véhicules à attendre sur les sommets. Pour cela une nouvelle approche est utilisée nommée Path Relinking. Ces deux phases utilisent un codage indirect pour parcourir les solutions de l'espace. L'espace de codage est identique pour les deux étapes et il est constitué de l'ensemble des tours géants. Pour passer de cet espace de codage à l'espace des solutions, des méthodes basées sur la procédure "split" sont utilisées pour les deux étapes.

Deux critères concernant les solutions du TDVRP sont étudiés : le premier concerne la durée totale de la tournée (dt) et le second le temps de trajet (rt), c'est-à-dire le temps de conduite du chauffeur. L'algorithme proposé est bi-critère et propose un ensemble de solutions non-dominées.

Pour tester les performances de cette approche, des instances ont été générées à partir d'instances publiées pour le LRP. Les résultats permettent d'obtenir en moyenne 20 solutions non-dominées dont une solution sans attente. Les gains sur le temps de conduite, bien qu'assez faibles, peuvent quand même intéresser des entreprises qui gèrent des compagnies de transport. Ces solutions peuvent permettre d'éviter les problèmes de chauffeurs dans l'incapacité de terminer leur tournée à cause du respect de la législation du temps de conduite. Ces situations sont très dommageables pour la société de transport car elles entraînent un surcoût important.

L'intérêt principal du TDVRP est d'inclure dans la résolution la variabilité des durées de trajet. A ce titre, il ouvre de nombreuses perspectives. La première était envisagée initialement dans cette thèse mais n'a pu être réalisée, faute de temps. Elle consistait à étendre le time dependent au DARP, définissant ainsi le TDDARP. Dans le TDDARP des problèmes

nouveaux apparaissent notamment lors de l'évaluation qui doit prendre en compte la variabilité de la durée de trajet entre les clients. Comme pour le TDVRP, ceci engendre une augmentation importante de la complexité de l'évaluation et par conséquent des temps de calculs. Le besoin de recourir à des techniques de réduction de temps de calcul (en particulier les évaluations partielles) devient alors impératif pour conserver des temps de réponse raisonnables.

Deuxièmement, le modèle proposé pour le TDVRP inclut les temps de pauses dans les tournées pour pouvoir réduire le temps de conduite effectif. Ces temps de pause correspondent déjà à une réalité dans le domaine du transport car ils sont imposés par la législation sur le temps de travail des chauffeurs. Mais la législation est beaucoup plus vaste sur le sujet. Elle couvre la totalité de l'activité des chauffeurs (organisation des temps de pause, mécanismes de compensation et de dérogation partielle) et impose d'autres types de contraintes sur les pauses qui ne sont pas prises en compte ici. L'étape suivante serait d'ajouter ces contraintes de manière à coupler la problématique d'optimisation des tournées à celle de gestion des emplois du temps des chauffeurs. On obtiendrait ainsi des tournées respectant la législation.

Enfin le modèle pourrait être amélioré pour prendre en compte des découpages par périodes spécifiques à chaque arc du réseau. Une telle approche permettrait de prendre en compte de manière plus fine l'évolution quotidienne de la charge de trafic sur chaque arc. Il serait alors intéressant de tester les conséquences de ce niveau de détails sur les résultats obtenus et sur les temps de calcul.

Concernant le time dependent VRP, une limitation importante est que les fluctuations sur les durées de trajet ne peuvent prendre en compte que les phénomènes prévisibles et périodiques. Pour modéliser en partie les phénomènes non prévisibles et leur impact sur les durées de trajet, il est nécessaire d'introduire une approche stochastique. Dans le chapitre suivant, la notion de durée de trajet stochastique est introduite pour le DARP. Dans ce problème, les durées de trajet sont alors modélisées par des variables aléatoires dont les caractéristiques sont connues. L'objectif est alors de générer des solutions robustes, c'est-à-dire qui peuvent supporter certaines fluctuations des temps de transport.

Références

Baldacci, R., Christofides, N., Mingozzi, A., 2007. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Math. Program. 115, 351–385. doi:10.1007/s10107-007-0178-5

Beasley, J., 1983. Route first—Cluster second methods for vehicle routing. Omega 11, 403–408. doi:10.1016/0305-0483(83)90033-6

Cheng, R., Gen, M., Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. Comput. Ind. Eng. 30, 983–997. doi:10.1016/0360-8352(96)00047-2

Christofides, N., 1979. Combinatorial optimization. John Wiley & Sons Canada, Limited.

Christofides, N., Eilon, S., 1969. An Algorithm for the Vehicle-Dispatching Problem. OR 20, 309–318. doi:10.2307/3008733

Dongarra, J.J., Bunch, J.R., Moler, C.B., Stewart, G.W., 1979. LINPACK Users' Guide. SIAM.

Feo, T.A., Resende, M.G.C., 1995. Greedy Randomized Adaptive Search Procedures. J. Glob. Optim. 6, 109–133. doi:10.1007/BF01096763

Glover, F., Laguna, M., Taillard, E., De Werra, D., 1993. Tabu search. Baltzer Basel.

Ho, S.C., Gendreau, M., 2006. Path relinking for the vehicle routing problem. J. Heuristics 12, 55–72. doi:10.1007/s10732-006-4192-1

Ichoua, S., Gendreau, M., Potvin, J.-Y., 2003. Vehicle dispatching with time-dependent travel times. Eur. J. Oper. Res. 144, 379–396. doi:10.1016/S0377-2217(02)00147-9

Lacomme, P., Prins, C., Prodhon, C., Ren, L., 2015. A Multi-Start Split based Path Relinking (MSSPR) approach for the vehicle routing problem with route balancing. Eng. Appl. Artif. Intell. 38, 237–251. doi:10.1016/j.engappai.2014.10.024

Malandraki, C., Daskin, M.S., 1992. Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms. Transp. Sci. 26, 185–200. doi:10.1287/trsc.26.3.185

Mester, D., Bräysy, O., 2007. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Comput. Oper. Res. 34, 2964–2975. doi:10.1016/j.cor.2005.11.006

Nagata, Y., Bräysy, O., 2009. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. Networks 54, 205–215.

Pareto, V., 1896. Politique, Cours d'Economie, Rouge. Lausanne Switz. I 896.

Prins, C., 2009. A GRASP × Evolutionary Local Search Hybrid for the Vehicle Routing Problem, in: Pereira, F.B., Tavares, J. (Eds.), Bio-Inspired Algorithms for the Vehicle Routing Problem, Studies in Computational Intelligence. Springer Berlin Heidelberg, pp. 35–53.

Sedeño-Noda, A., Raith, A., 2015. A Dijkstra-like method computing all extreme supported non-dominated solutions of the biobjective shortest path problem. Comput. Oper. Res. 57, 83–94.

Toth, P., Vigo, D., 2001. The Vehicle Routing Problem, in: Toth, P., Vigo, D. (Eds.), . Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 1–26.

Vidal, T., 2013. Approches générales de résolution pour les problèmes multi-attributs de tournées de véhicules et confection d'horaires. Université de Montréal & Université de Technologie de Troyes.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2014. Implicit depot assignments and rotations in vehicle routing heuristics. Eur. J. Oper. Res. 237, 15–28. doi:10.1016/j.ejor.2013.12.044

Wen, L., Eglese, R., 2015. Minimum cost VRP with time-dependent speed data and congestion charge. Comput. Oper. Res. 56, 41–50. doi:10.1016/j.cor.2014.10.007

Zhang, C., Lin, Z., Zhang, Q., Lin, Z., 2005. Permutation Distance: Properties and Algorithms, in: Proceedings of the 6th Metaheuristics International Conference of MIC2005. Vienna, Austria. pp. 211–216.

Zitzler, E., 1999. Evolutionary algorithms for multiobjective optimization: Methods and applications. Citeseer.

CHAPITRE 4

Problème du transport à la demande avec temps de trajet stochastique

Objectifs du chapitre :

Ce chapitre concerne les problèmes de transport à la demande dans lesquels les temps de trajet sont modélisés par des réalisations de variables aléatoires. Ces temps de trajet varient en suivant une loi normale autour d'une valeur moyenne qui correspond au temps de trajet déterministe.
En utilisant l'espérance des lois, les précédentes méthodes de résolution sont réutilisables. Mais à cause des fluctuations des temps de transport, les solutions obtenues peuvent ne pas être valides lorsqu'elles sont réalisées par les chauffeurs. Dans le modèle, trois hypothèses ont été étudiées concernant les politiques de gestion des chauffeurs en cas de retard ou d'avance lors de l'arrivée sur un sommet. Ces comportements influencent le résultat.
Après avoir défini le problème, ce chapitre comporte trois parties. Une méthode d'évaluation d'une tournée est présentée dans la première partie et elle permet, pour un ordre de passage sur les clients donné, d'obtenir une tournée robuste. La robustesse est la capacité d'une solution à être réalisable malgré les variations sur le temps de transport. En utilisant cette méthode d'évaluation, on étudie la robustesse des meilleures solutions connues dans la littérature sur les problèmes déterministes. Dans une deuxième partie, une approche ELS (*Evolutionary Local Search*) est développée pour résoudre le problème et obtenir des solutions robustes. Enfin une approche multicritère est proposée pour obtenir un ensemble de solutions incomparables concernant deux critères qui sont le nombre de kilomètres (coût pour le problème déterministe) et la robustesse.

4.1. Les problèmes stochastiques

4.1.1. Les problèmes stochastiques dans la littérature

Dans la littérature, il n'existe pas de problème de transport à la demande (DARP) dans lequel les temps de trajet sont modélisés par des variables aléatoires dont les lois associées sont connues. Par contre il existe des problèmes de tournées de véhicules dans lesquels des évènements stochastiques viennent perturber les temps de trajet. Ces problèmes sont surtout des problèmes dynamiques. Par exemple dans l'article de (Schilde et al., 2014), les auteurs s'intéressent au problème du transport à la demande (DARP) dynamique incluant des évènements aléatoires dans les instances représentant des accidents sur le réseau. Ces accidents modifient le temps de trajet sur les arcs proches de l'évènement.

Les problèmes stochastiques dans la littérature peuvent aussi concerner d'autres caractéristiques. Par exemple (Fleury et al., 2005) traitent d'un problème de tournées sur arcs dans lequel les demandes sont représentées par des variables aléatoires dont les lois sont connues. Plus récemment, (Gauvin et al., 2014) considèrent un problème de tournées de

véhicules avec une méthode de "branch-cut-and-price" dans laquelle les quantités collectées sur les sommets suivent également une loi aléatoire connue.

Le problème traité dans ce chapitre est un problème de transport à la demande (DARP) dans lequel les temps de trajet entre les sommets sont modélisés par des variables aléatoires. Elles permettent de prendre en compte les variations qui existent dans un réseau routier, comme : les arrêts dus aux feux de circulation, la traversée de piétons, ou encore les difficultés d'insertion aux niveaux des intersections. Individuellement ces évènements ont un impact mineur sur le temps de trajet mais leur accumulation peut provoquer des variations des temps de transport qui sont loin d'être négligeables. Par exemple, sans prendre en compte les phénomènes de circulation, pour se rendre au travail certains matins, il faut 20 minutes si tous les feux de circulation sont au vert et au contraire il faut 30 minutes s'ils sont rouges. Dans le modèle, la variation est estimée à $\pm 10\%$ du temps de trajet moyen.

Toutes les requêtes sont connues au début de l'optimisation et une solution doit toutes les satisfaire. L'objectif est de prendre en compte ces variations afin de proposer une solution de faible coût et qui a une forte probabilité de satisfaire pleinement les clients quels que soient les temps de trajet réels dans le réseau routier.

4.1.2. L'interprétation des fenêtres de temps dans le problème

Dans la version statique du problème, la partie optimisation de la tournée est réalisée avant le début des tournées. Cette optimisation permet d'obtenir une tournée, c'est-à-dire un ordre de passage sur tous les clients ainsi que des dates de passage du véhicule au niveau de chaque client.

Comme l'optimisation intervient avant la réalisation des tournées, il existe deux interprétations possibles de la contrainte de fenêtres de temps dans le DARP. Soit ce sont les disponibilités réelles des clients, soit ce sont seulement des plages horaires dans lesquelles l'entreprise de transport peut choisir une date. Dans ce dernier cas, cette date est ensuite communiquée au client et une nouvelle fenêtre de temps plus réduite autour de cette date est alors définie, comme illustré dans la figure 4.21.

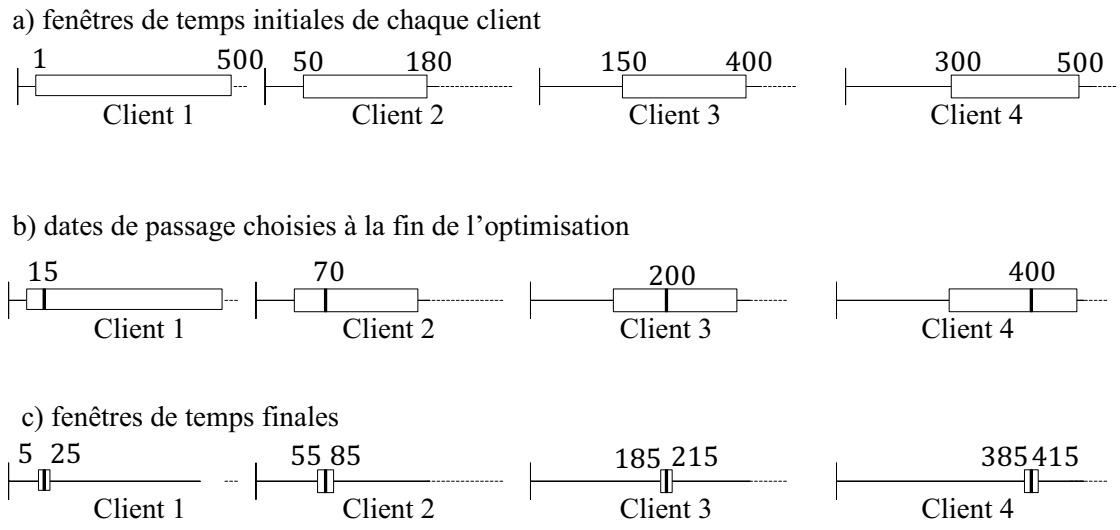


Figure 4.1: Données initiales sur le vecteur λ

Pour illustrer ces deux interprétations, deux exemples réels qui peuvent être modélisés par un problème de voyageur de commerce avec des fenêtres de temps sur les clients (TSPTW) sont développés.

Exemple 1 : le problème modélise une livraison de magazines dans différents bureaux de tabac. Les fenêtres de temps correspondent aux horaires d'ouvertures des différents magasins. Une fois la tournée optimisée, les dates de passage ne sont pas communiquées aux clients donc dans ce premier exemple, le véhicule peut prendre de l'avance ou du retard sur les dates en respectant toutefois les horaires d'ouverture du magasin.

Exemple 2 : le problème modélise la livraison de produits dans des supermarchés, avec des produits volumineux qui nécessitent par exemple de décharger le camion au niveau d'un quai avec l'aide d'une équipe spécialisée. Dans cette situation, on peut supposer que le client fournit à l'entreprise de livraison une fenêtre de temps initiale. L'entreprise optimise alors sa tournée puis communique les dates de livraison précises aux clients. Le supermarché (le client) réserve alors l'équipe pour décharger le camion à la date sélectionnée. Dans cette situation, les retards et les avances sur les dates de livraisons peuvent avoir des conséquences importantes sur la solution. Si l'heure d'arrivée est différente de l'horaire prévu, les équipes de déchargement peuvent ne pas être disponibles et la cargaison peut donc être perdue, par exemple dans le cas de produits congelés.

Dans la thèse, seule la version dans laquelle les dates de début de service ne sont pas communiquées aux clients a été envisagée (exemple 1). Une tournée est valide lors d'une réalisation si toutes les dates de début de service (qui peuvent être différentes des dates pré-calculées à cause des fluctuations) respectent les contraintes associées au problème du DARP. Il est donc possible que les dates d'une réalisation soient éloignées des dates prévues sans pour autant avoir des conséquences sur la tournée.

4.1.3. Le comportement du chauffeur et l'influence sur la solution

Une fois l'optimisation réalisée, une solution est obtenue. Cette solution comporte un ensemble de tournées qui se composent d'une liste de clients à visiter et des dates de passages prévisionnelles. Ces dates peuvent s'avérer exactes ou non en fonction des conditions de circulations qui ne sont pas prévisibles.

Une tournée est transmise à un chauffeur sous la forme d'une feuille de route qui définit :

- une suite ordonnée de clients à traiter (comme c'est le cas pour le VRP);
- les dates d'arrivée prévues pour chaque client (comme c'est le cas pour le VRPTW)
- une politique de gestion à appliquer pour prendre en compte l'heure d'arrivée réelle chez le client. Une telle politique définit par exemple, qu'en cas d'avance par rapport à une date prévue, le chauffeur doit attendre l'heure initialement planifiée comme c'est le cas pour les horaires de bus.

Dans le modèle, le chauffeur n'a pas d'influence sur la réalisation du temps de trajet entre les différents sommets. Par exemple, il ne peut pas accélérer pour compenser un retard. Par contre son comportement sur un sommet en retardant ou en avançant la date de début de service peut avoir des conséquences pour la suite de la tournée. L'entreprise doit donc donner au chauffeur des consignes à respecter en fonction de l'avance ou du retard. Dans le modèle, ces consignes sont données au chauffeur au début de la tournée dans la feuille de route en même temps que la liste des sommets à visiter. Elles ne peuvent pas être modifiées durant la tournée.

Trois consignes différentes sont étudiées dans ce chapitre, chacune représentant une hypothèse. Pour simplifier les exemples, les dates de début de service B_i correspondent aux dates de départ du sommet D_i.

a) Hypothèse 1 : respecter les dates de début de service fixées

Cette hypothèse permet d'appréhender le fonctionnement d'un grand nombre de systèmes de transport. L'entreprise communique au chauffeur les dates de début de service B_i sur les sommets. Cette date de début de service B_i peut être différente de la date d'arrivée prévue sur un sommet notée A_i. Le chauffeur doit respecter les dates B_i si possible. S'il arrive après cette date sur un sommet, il repart immédiatement. La consigne donnée au chauffeur est : "en cas d'arrivée avant l'heure prévue de début de service, vous devez attendre la date prévue, sinon repartez le plus tôt possible".

Ceci correspond par exemple à ce qu'on retrouve couramment lors des livraisons ou des approvisionnements des supermarchés par des centrales d'achat. Si le camion de livraison arrive avant que les quais de chargement/déchargement ne soient libres (avant l'ouverture prévue) le camion doit attendre que les personnels du supermarché arrivent pour ouvrir les quais de livraison.

Un exemple est illustré dans le tableau 4.1, la première colonne est la liste des sommets à visiter, puis chaque colonne représente un des acteurs :

- les clients avec leurs fenêtres de temps,
- l'entreprise avec les dates de trajet théorique t_{s_{i-1},s_i} ainsi que les heures d'arrivées A_i et de début de service théorique B_i ;
- le véhicule avec les informations qui lui sont données par l'entreprise et qui dépendent de l'hypothèse, comme l'attente W_i sur les sommets et les date de début B_i ;
- le véhicule avec les informations qui lui sont données par l'entreprise et qui dépendent de l'hypothèse, comme l'attente W_i sur les sommets et les date de début B_i.

La dernière colonne représente une réalisation possible de la tournée, avec la durée réelle du temps de trajet, la date d'arrivée A_i puis l'attente et la date de départ du véhicule qui dépendent de l'hypothèse considérée.

Dans l'exemple, le véhicule part à 8h du dépôt, le trajet jusqu'au client A+ prend 1h au lieu des 50 minutes prévues. Il arrive à 9h sur le sommet $A+$. Il est en retard, mais il arrive avant la date prévue 9h15. Il attend donc 9h15 avant de repartir. Sur le sommet suivant, le véhicule arrive en avance, à 10h au lieu de 10h05 prévu. Le véhicule attend quand même la date de début de service prévue (10h15) pour repartir. Il arrive au dépôt à l'heure prévue 11h05.

Tableau 4.1:
Exemple du comportement d'un chauffeur sous l'hypothèse 1

	Client (dispo.)		**Entreprise (valeurs théoriques)**			**Véhicule (informations)**		**Exemple de scénario**			
s_i	e_i	l_i	t_{s_{i-1},s_i}	A_i	B_i	W_i	B_i	t_{s_{i-1},s_i}	A_i	W_i	B_i
Départ	8h	12h	50 min	/	8h00	/	8h00	/	/	/	8h00
A+	9h	10h	50 min	8h50	9h15	/	9h15	1h	9h00	15 min	9h15
A-	10h	11h	50 min	10h05	10h15	/	10h15	45 min	10h00	15 min	10h15
Fin	8h	12h	50 min	11h05	/	/	/	50 min	11h05	/	/

Cette hypothèse $H1$ correspond au cas où le rendez-vous est fixé à l'avance avec le client dans la fenêtre de temps. Si le chauffeur arrive en avance, il attend que le client arrive avant de repartir. Il ne peut donc pas profiter d'un temps de trajet avantageux pour repartir en avance et compenser un futur temps de trajet moins avantageux.

b) Hypothèse 2 : respecter les temps d'attente sur les sommets

La seconde hypothèse consiste à communiquer aux chauffeurs les durées d'attente sur les différents sommets qui composent la tournée. Le chauffeur doit alors respecter ce temps d'attente, qu'il soit arrivé en avance ou en retard sur le sommet. Il ne connait pas les horaires d'ouverture des clients et il ne connait pas non plus les heures d'arrivée prévues par le système de planification des tournées. La feuille de route d'un chauffeur ne contient qu'une liste ordonnée de clients avec une durée d'attente. La consigne donnée au chauffeur est donc : "quel que soit l'heure d'arrivée chez le client, vous devez attendre la durée prévue avant de commencer votre service".

Ce type de fonctionnement est un fonctionnement identique à ce qu'on peut trouver sur une rame de métro. Les utilisateurs des lignes automatisées de métro de certaines grandes villes ont déjà constaté que les rames arrivent parfois en retard par rapport aux informations des tableaux d'affichages et parfois en avance. En tout état de cause, en cas d'avance ou de retard, le panneau d'affichage est mis à jour et la rame de métro repart quelques minutes après son arrivée. On est ici dans un cas complètement différent du cas 1, puisque ce sont les horaires qui s'adaptent (en quelques sortes) aux fluctuations.

Tableau 4.2:
Exemple du comportement d'un chauffeur sous l'hypothèse 2

	Client (dispo.)		**Entreprise (valeurs théoriques)**			**Véhicule (informations)**		**Exemple de scénario**			
s_i	e_i	l_i	t_{s_{i-1},s_i}	A_i	B_i	W_i	B_i	t_{s_{i-1},s_i}	A_i	W_i	B_i
Départ	8h	12h	50 min	/	8h00	/	8h00	/	/	/	8h
A+	9h	10h	50 min	8h50	9h15	25 min	/	1h	9h	**25 min**	9h25
A-	10h	11h	50 min	10h05	10h15	10 min	/	35 min	10h00	**10 min**	10h10
Fin	8h	12h	50 min	11h05	/	0 min	/	50 min	11h00	/	/

Dans l'exemple du tableau 4.2, le véhicule arrive donc à 9h sur le sommet $A+$. Il respecte sa pause de 25 minutes, même s'il est en retard puisqu'il aurait dû arriver à 8h50 d'après les valeurs théoriques. Il repart donc à 9h25 qui correspond dans le tableau à la case B_i dans le scénario. Sur le sommet suivant, le véhicule arrive 10h00 et il est en avance par rapport à la l'heure d'arrivée initialement prévue de 10h05. Sur le sommet A+, le chauffeur attend 10 minutes avant de repartir à la date B_i = 10h10. Il repart donc aussi en avance par rapport à l'heure de départ planifiée initialement à 10h15.

Si l'avance ou le retard est trop important sur un sommet, alors la date B_i peut violer les contraintes du problème, notamment les fenêtres de temps minimales et maximales. La solution est alors non réalisable par le véhicule.

c) Hypothèse 3 : respecter les temps d'attente en cas d'avance sinon respecter les dates de début de service

Dans la troisième hypothèse, l'entreprise communique au chauffeur les durées d'attente et les dates de début de services B_i sur chaque sommet. Si le chauffeur est en avance sur la date d'arrivée prévue A_i alors il doit attendre W_i. Sinon, il y a deux possibilités : soit il arrive en

retard mais quand même avant la date B_i et il repart à la date B_i, soit il arrive après la date B_i et il repart alors le plus tôt possible. La consigne donnée au chauffeur est donc : "en cas d'arrivée avant l'heure prévue, attendez soit la date prévue, soit que votre attente soit égale à l'attente prévue, sinon repartez le plus tôt possible".

La troisième consigne consiste à communiquer aux chauffeurs à la fois, les durées d'attente W_i et les dates de début de service B_i qui composent la tournée. Le chauffeur doit alors respecter ce temps d'attente s'il arrive en avance et respecter la date de début de service B_i s'il arrive en retard sur le sommet.

Ce type de fonctionnement se retrouve dans les entreprises de transport où le transbordement des marchandises nécessite l'intervention d'équipes de manutention appartenant au client, de sorte que le déchargement d'un camion n'est possible qu'à l'horaire initialement prévu avec le client.

Dans l'exemple du tableau 4.3 le véhicule part à 8h, le trajet dure 1h au lieu des 50 minutes prévues et le véhicule arrive à 9h sur le sommet $A+$. Comme il est en retard, la durée de sa pause est réduite à 15min (la durée théorique est de 25min) et le véhicule repart à 9h15 qui correspond à la date B_i dans le scénario. Sur le sommet suivant, le véhicule arrive en avance à 10h au lieu de 10h05 qui est la date prévue et il attend donc 10 minutes (la durée imposée par l'entreprise) avant de repartir à 10h10. Il arrive au dépôt à 11h.

Tableau 4.3:
Exemple du comportement d'un chauffeur sous l'hypothèse 3

	Client (dispo.)		**Entreprise (valeurs théoriques)**			**Véhicule (informations)**		**Exemple de scénario**			
s_i	e_i	l_i	t_{s_{i-1},s_i}	A_i	B_i	W_i	B_i	t_{s_{i-1},s_i}	A_i	W_i	B_i
Départ	8h	12h	50 min	/	8h00	/	8h00	/	/	/	8h
A+	9h	10h	50 min	8h50	9h15	25 min	9h15	1h	9h00	**15 min**	9h15
A-	10h	11h	50 min	10h05	10h15	10 min	10h15	45 min	10h00	**10 min**	10h10
Fin	8h	12h	50 min	11h05	/	0 min	/	50 min	11h00	/	/

Cette hypothèse $H3$ correspond à un cas intermédiaire entre les deux hypothèses précédentes H_1 et H_2. Elle a l'avantage de profiter de l'avance obtenue sur les temps de trajet mais aussi de réduire les retards en diminuant son temps d'attente si c'est possible.

d) Autres hypothèses : les cas hybrides

D'autres hypothèses peuvent être envisagées, mais seules les trois hypothèses précédentes sont traitées dans la suite de ce chapitre. Une politique de gestion qui serait très efficace consisterait à réévaluer la solution après chaque arrivée du véhicule sur un sommet, une fois que la date d'arrivée est connue. Ainsi, l'algorithme pourrait prendre en compte les réalisations des variables aléatoires sur les temps de trajet précédents puis modifier au besoin les dates B_i sur les sommets suivants. Dans une telle situation le chauffeur n'a pas de politique de gestion imposée sur chaque sommet. C'est l'algorithme qui lui indique s'il doit prolonger l'attente (et de combien) ou s'il doit repartir immédiatement. Cette hypothèse se rapproche fortement d'une version dynamique du DARP.

4.2. Définition du problème

Le problème du transport à la demande avec temps de transports stochastiques (SDARP - *Stochastic DARP*) est en grande partie identique à la définition introduite dans le chapitre 2 pour le DARP, proposée par (Cordeau and Laporte, 2003). Pour éviter les confusions, dans la suite de ce chapitre, le problème de transport à la demande avec des temps de trajet déterministes est noté DDARP (*Deterministic DARP*) au lieu de DARP. Dans le SDARP, les temps de transports varient autour d'une durée moyenne connue.

Le problème est constitué de n clients et de m véhicules. Chaque client est associé à deux sommets : un sommet origine et un sommet destination. Enfin, chaque client a une demande qui représente le nombre de personnes à transporter.

Un ensemble de tournées représente une solution si cet ensemble respecte un certain nombre de contraintes. Parmi ces contraintes, certaines ne dépendent pas de la variabilité des temps de trajet comme :

- le nombre de véhicules utilisés doit être inférieur à m
- la capacité du véhicule ;
- l'ordre de passage du véhicule sur les sommets associés aux clients (*pickup* et *delivery*) ;

D'autres, en revanche, dépendent des dates de passage sur les sommets de la tournée qui dépendent à leur tour du temps de trajet et des consignes suivies par le chauffeur. Il s'agit des contraintes sur :

- le temps de trajet des clients ;
- les fenêtres de temps pour le début du service sur un sommet ;
- la durée maximale de la tournée.

A une solution du SDARP est donc affecté un coût qui représente la distance parcourue par la flotte et un critère de robustesse qui correspond à la probabilité que la solution soit réalisable, c'est-à-dire à la probabilité que les contraintes qui dépendent du temps de trajet entre les sommets soient vérifiées.

4.2.1. Notations et modélisation des temps de trajet

Le temps de transport entre deux sommets i et j, précédemment noté t_{ij} n'est plus déterministe. Il est maintenant associé à une variable aléatoire. Pour faire la différence entre les variables stochastiques et déterministes, les variables suivantes sont définies :

- t_{ij} : le temps déterministe de trajet entre les sommets i et j ;
- T_{ij} : la variable aléatoire associée au temps de trajet entre les sommets i et j ;
- $T_{ij}(\omega)$: la valeur d'une réalisation de la variable aléatoire T_{ij}.

Pour les autres variables, les notations introduites dans le chapitre 2 pour le problème du DDARP sont conservées.

Afin de modéliser la variation autour du temps de trajet moyen t_{ij}, la variable aléatoire T_{ij} suit une loi normale centrée sur t_{ij}. La loi normale a été utilisée car, dans le réseau routier, les variations sur le temps de transport entre deux positions sont dues à un grand nombre de perturbations aléatoires. En faisant l'hypothèse que ces perturbations sont indépendantes, le théorème central limite (TCM) peut être appliqué. Il établit qu'une somme de variables aléatoires indépendantes converge en loi vers une loi normale.

Dans le modèle, on applique une restriction sur les lois normales utilisées car elles doivent définir des durées de transport et doivent être positives. C'est pourquoi, même si une loi

normale est définie sur $\mathbb{R}$ et à valeur dans $\mathbb{R}$ ($\mathbb{R} \rightarrow \mathbb{R}$), seules les réalisations à valeur positive sont considérées $\mathbb{R} \rightarrow \mathbb{R}^+$. En pratique, de telles valeurs négatives ont une très faible probabilité d'occurrence, mais leur survenue ne peut pas être ignorée dans des programmes informatiques qui font appel, lors de réplications, à des millions de simulations de variables aléatoires.

Les caractéristiques de ces lois normales dépendent du trajet considéré. Dans le modèle, pour deux sommets quelconques i et j, l'espérance est égale à t_{ij} et la variance σ_{ij} à la valeur $t_{ij}/10$. La valeur σ_{ij} représente une variation de plus ou moins 10% du temps de trajet et ceci pour plus de 68% des réalisations. La loi suivie par T_{ij} est notée $\mathcal{N}(t_{ij}, \sigma_{ij})$ et la fonction de densité de probabilité associée à cette loi est représentée dans la figure 4.2 et vaut :

$$f_{T_{ij}}(x) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} \times e^{-\frac{1}{2}\left(\frac{x-t_{ij}}{\sigma_{ij}}\right)^2}$$

La densité de probabilité permet de calculer la probabilité sous la forme d'une intégrale. Par exemple, pour une loi normale $\mathcal{N}(t_{ij}, \sigma_{ij})$, la probabilité que la valeur d'une réalisation aléatoire de la loi soit comprise entre t_{ij} - σ_{ij} et t_{ij} + σ_{ij} est égale à $\int_{t_{ij}-\sigma_{ij}}^{t_{ij}+\sigma_{ij}} f_{T_{ij}}(x)dx$ soit 68,2%. Cette probabilité est notée : $\mathrm{P}(t_{ij} - \sigma_{ij} \leq T_{ij} \leq t_{ij} + \sigma_{ij}) = 68{,}2\%$

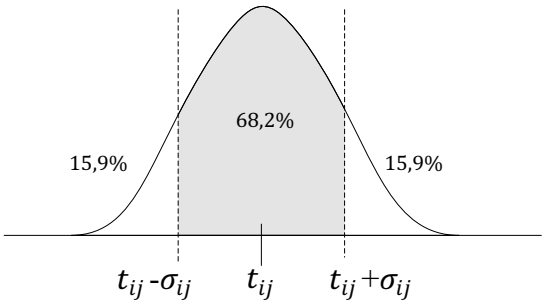


Figure 4.2 : Densité d'une loi normale d'espérance t_{ij} et de variance σ_{ij}

Lors de la réalisation de la variable aléatoire $T_{s_{i-1}s_i}$ trois cas peuvent se présenter. Le premier cas correspond à un temps de trajet égal à $t_{s_is_j}$, ce qui modélise le fait que le véhicule arrive à l'horaire prévu. Il est intéressant de noter que, comme le temps de trajet suit une loi normale de densité de probabilité continue sur $\mathbb{R}$, la probabilité que ce cas précis se produise est nulle.

$$\mathrm{P}\left(t_{s_is_j} \leq T_{s_is_j} \leq t_{s_is_j}\right) = \int_{t_{s_is_j}}^{t_{s_is_j}} f_{T_{s_is_j}}(x)dx = 0\%$$

Les deux autres cas ont la même probabilité : dans 50% des réalisations le temps de trajet du véhicule est inférieur au temps prévu et dans les autres 50% de réalisations il est supérieur.

$$\mathrm{P}\left(T_{s_is_j} \leq t_{s_is_j}\right) = \mathrm{P}\left(t_{s_is_j} \leq T_{s_is_j}\right) = \int_{-\infty}^{t_{s_is_j}} f_{T_{s_is_j}}(x)dx = \int_{t_{s_is_j}}^{+\infty} f_{T_{s_is_j}}(x)dx = 50\%$$

4.2.2. Rappel sur les variables pour les tournées du DDARP

Dans le DDARP, un objet nommé λ a été introduit, constitué d'une suite ordonnée de n_k sommets. L'évaluation de λ consiste à calculer un certain nombre de dates associées au passage du véhicule afin de vérifier si une tournée qui suit l'ordre de sommets imposé par λ est possible. Les notations concernant ces dates sont les suivantes :

- s_i: définit le sommet à la i^{eme} position dans la tournée, s_1=s_{nk} représentent le dépôt ;
- B_i: définit la date à laquelle le véhicule commence son service sur le i^{eme} sommet de la tournée ;
- W_i: définit la durée d'attente du véhicule sur le i^{eme} sommet de la tournée ;
- D_i: définit la date de départ du véhicule du i^{eme} sommet de la tournée ;
- A_i: définit la date d'arrivée du véhicule sur le i^{eme} sommet de la tournée.

Dans le transport à la demande déterministe (DDARP), le temps de trajet entre deux sommets est connu. Il est donc possible de déterminer les valeurs des variables B_i, A_i etc... sur les sommets en utilisant par exemple la méthode de (Firat and Woeginger, 2011). Comme cela a été souligné au chapitre 2, certaines de ces variables sont liées les unes aux autres. Si la date de début de service sur un sommet d'une tournée est fixée, alors la date de départ de ce sommet est connue. La différence entre les deux correspond à la durée du service. L_i correspond au temps de trajet pour transporter du client i de son sommet d'origine S_i à son sommet de destination S_j. Cette durée est définie comme la différence entre D_i, date de départ du véhicule sur S_i et B_j, date de début de service sur le sommet S_j qui correspond au moment où le client est déposé.

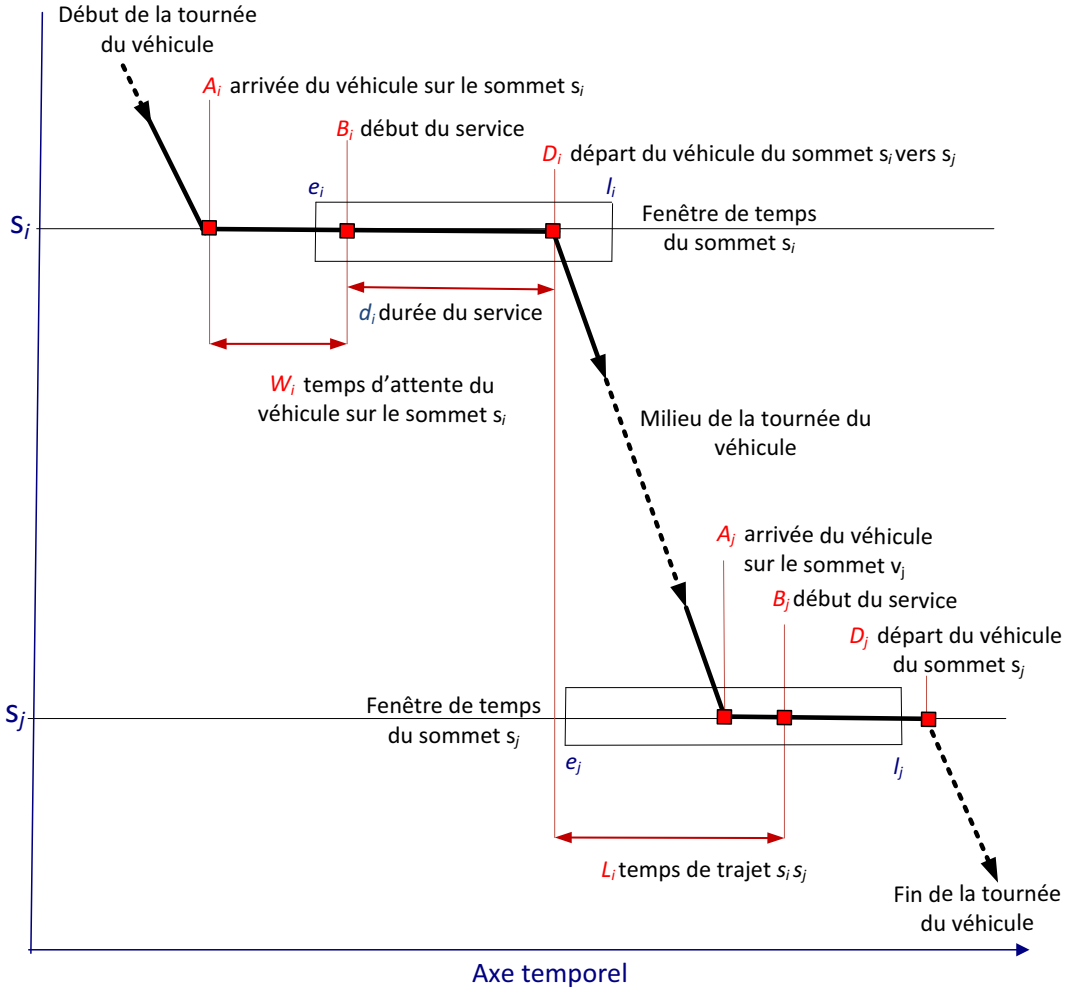


Figure 4.3 : Variables dans une tournée du DDARP

4.2.3. Les variables aléatoires du SDARP et leurs réalisations

Dans un problème de transport avec des temps de trajet stochastiques, comme pour le cas déterministe, si la date de début du service sur un sommet est connue alors la date de départ est aussi connue avec :

$$D_i = \mathrm{B_i} + \mathrm{d}_{s_i}$$

Mais, contrairement au cas déterministe, la date d'arrivée lors de la réalisation d'une tournée n'est pas connue car elle correspond à la date de départ sur le sommet précédant plus le temps de trajet, avec le temps de trajet comme une variable aléatoire.

$$A_i \neq \mathrm{D_i} + \mathrm{t}_{s_i s_{i+1}}$$

Ce cas déterministe est illustré par la figure 4.4.

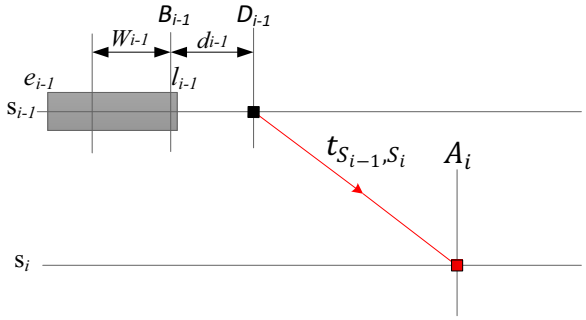


Figure 4.4 : Variables dans une tournée du DDARP

Pour le SDARP, les notations : $\mathrm{A_i^\omega}$, $\mathrm{B_i^\omega}$, $\mathrm{W_i^\omega}$ et $\mathrm{D_i^\omega}$ sont utilisées pour représenter les réalisations des variables associées à l'évènement ω et $\mathrm{A_i^{VA}}$, $\mathrm{B_i^{VA}}$, $\mathrm{W_i^{VA}}$ et $\mathrm{D_i^{VA}}$ sont les variables aléatoires associées. La figure 4.5 illustre qu'à chaque réalisation de la variable aléatoire $\mathrm{T}_{s_{i-1}s_i}$ la date d'arrivée peut changer. Deux réalisations sont alors notées A_i^ω et $A_i^{\omega\prime}$. La courbe en pointillés représente la densité de la loi suivie par A_i^{VA}, qui est égale à la fonction de densité de la loi suivie par $\mathrm{T}_{s_{i-1}s_i}$ centrée sur la valeur A_i.

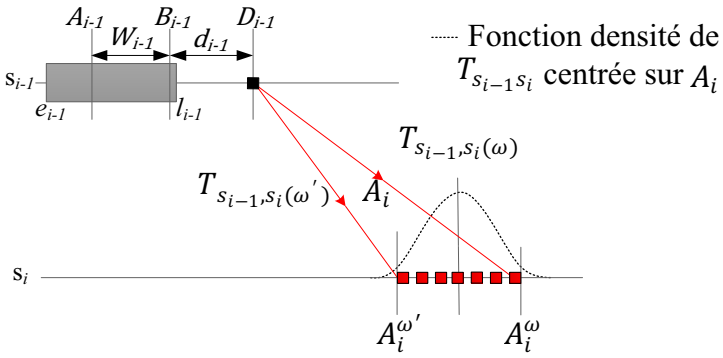


Figure 4.5 : Conséquence sur la tournée de la réalisation de $T_{s_{i-1}s_1}(\omega)$

La date d'arrivée est égale à :

$$\mathrm{A_i^\omega} = \mathrm{D_i} + \mathrm{T}_{s_i s_{i+1}}(\omega)$$

Il n'est donc pas possible de connaître la date d'arrivé $\mathrm{A_i^\omega}$ sur un sommet. Par contre il est possible d'augmenter ou de diminuer l'attente sur le sommet afin de positionner $\mathrm{B_i^{VA}}$ de

manière à augmenter la probabilité que B_i^{ω} soit proche de la valeur désirée. Puisque la date de début de service est égale à $B_i^{VA} = A_i^{VA} + W_i$.

Par exemple, pour que la contrainte $B_i^{\omega} \leq Cst$ illustrée sur la figure 4.6 ait une probabilité maximale d'être vraie, il faut que la valeur de $\mathbb{E}[B_i^{VA}]$ soit la plus petite possible. Cela signifie que pour avoir le plus de chance d'arriver sur un sommet avant la date limite Cst, il faut concevoir un planning où la date d'arrivée a été fixée au plus tôt. Pour illustrer ces propos, c'est le comportement classique adopté par un voyageur désirant prendre son vol et qui part de chez lui très tôt afin d'être certain de rejoindre l'aéroport avant l'embarquement et ceci "quelles que soient" les conditions de circulation rencontrées.

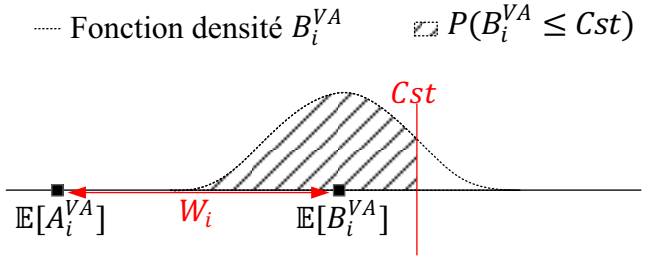


Figure 4.6 : Variables dans une tournée du SDARP

Au contraire, plus la valeur de $\mathbb{E}[B_i^{VA}]$ est grande et plus la probabilité est faible, comme le montre la figure 4.7.

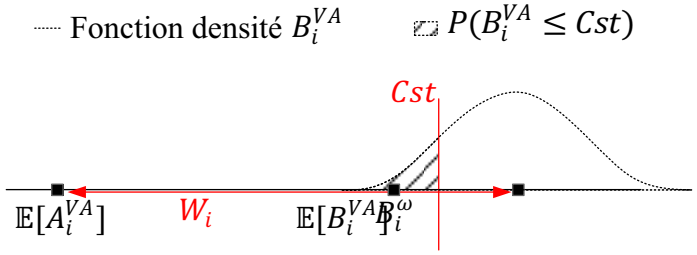


Figure 4.7 : Une contrainte non valide pour le DDARP dans le SDARP

On peut noter qu'une contrainte non valide pour le DDARP avec, par exemple, une date d'arrivée $A_i > Cst$, possède une probabilité non nulle d'être valide dans le SDARP, à cause des variations sur le temps de trajet. Dans la figure 4.7, la réalisation B_i^{ω} représente une date d'arrivée valide même si la valeur de $\mathbb{E}[B_i^{VA}]$ est supérieure à Cst. Donc, quelles que soient les dates d'arrivées fixées sur les sommets de λ, la tournée possède une probabilité non nulle d'être réalisable dans le SDARP (cette probabilité pouvant être extrêmement faible).

Dans la suite les variables A_i, B_i sont associées pour le SDARP aux valeurs de $\mathbb{E}[A_i^{VA}]$ et $\mathbb{E}[B_i^{VA}]$. Les temps d'attente, qui peuvent être considérés comme des variables durant la partie optimisation, sont aussi des variables aléatoires qui dépendent des réalisations sur les temps de trajet de la tournée. Mais, à la différence des autres, ils dépendent aussi du comportement suivi par le chauffeur, comme ceci est mis en évidence dans la partie suivante.

4.2.4. Modélisation des différentes hypothèses sur le comportement du chauffeur

Comme expliqué dans la partie 4.1.3, trois hypothèses différentes sont envisagées et elles représentent des comportements différents des chauffeurs au moment de leur arrivée sur les clients. Si aucune variation dans le temps de trajet n'a lieu, ces trois hypothèses permettent au chauffeur de réaliser une tournée valide.

La figure 4.8 représente le trajet du client entre deux sommets consécutifs s_{i-1} et s_i d'une tournée dans le cas déterministe. Dans cette figure, le temps de chargement est omis par souci de simplification.

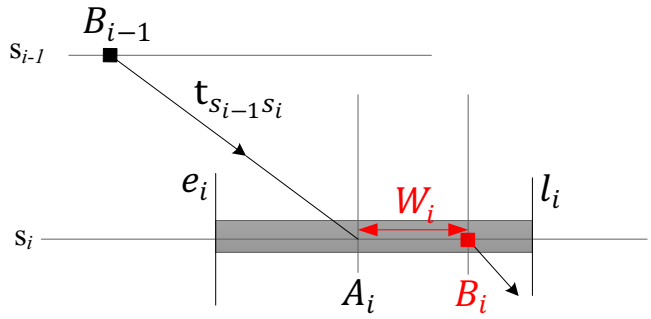


Figure 4.8 : Illustration du cas déterministe

a) Hypothèse 1 : respecter les temps de début de service

Cette hypothèse permet d'appréhender le fonctionnement d'un grand nombre de systèmes de transport. Dans ce mode de fonctionnement, l'entreprise communique au chauffeur une feuille de route contenant les dates de début de service B_i sur les différents sommets qui composent la tournée. Le chauffeur ajuste le temps d'attente sur le sommet en fonction de sa date d'arrivée. S'il arrive avant la date prévue, le temps d''attente W_i est augmenté, comme illustré sur la partie gauche de la figure 4.9. S'il arrive après la date A_i, l'attente W_i est diminuée, comme illustré sur la figure 4.9 à droite.

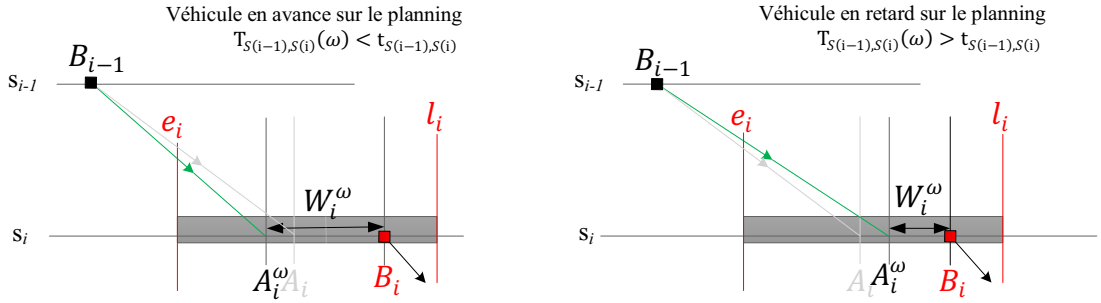


Figure 4.9 : Illustration dans le cas stochastique de l'hypothèse 1

Le pseudocode associé à cette hypothèse est détaillé dans la figure 4.10.

```
1  // Bloc hypothèse H1
2  |  if (A_i^ω ≤ B_i) then
3  |  |     B_i^ω := B_i
4  |  else
5  |  |     B_i^ω := A_i^ω
6  |  endif
```

Figure 4.10 : Lignes utilisées pour la simulation de l'hypothèse 1

Les variations sur les temps de trajet sont compensées à l'aide de l'attente W_i uniquement si $A_i^\omega \leq B_i$. Si ce n'est pas le cas, c'est-à-dire si $A_i^\omega > B_i$, alors soit $A_i^\omega \leq l_i$ et le début du service doit démarrer immédiatement après l'arrivée du véhicule, soit $A_i' > e_i$ et l'exécution de la

tournée est irréalisable, le client du sommet s_i n'ayant pas pu être servi dans la fenêtre de temps prévue (ou demandée).

Comme le véhicule attend la date B_i dans le cas où son arrivée a lieu plus tôt que la date planifiée, il est évident qu'aucune violation de contrainte ne peut avoir lieu, notamment par rapport aux fenêtres de temps minimales des clients. Ceci a des répercussions sur l'évaluation des tournées sous cette hypothèse. De plus, en suivant les consignes, le véhicule ne peut pas améliorer son temps de service pour les clients, ni son temps total de trajet, car les valeurs de B_i^ω sont supérieures ou égales aux B_i.

b) Hypothèse 2 : respecter les temps d'attente sur les sommets

Sous cette hypothèse, on considère que la feuille de route donnée aux chauffeurs comprend les durées d'attente W_i sur les différents sommets qui composent la tournée. Le chauffeur doit alors respecter ce temps d'attente, qu'il soit arrivé en avance ou en retard sur le sommet. Un exemple est illustré dans la figure 4.11.

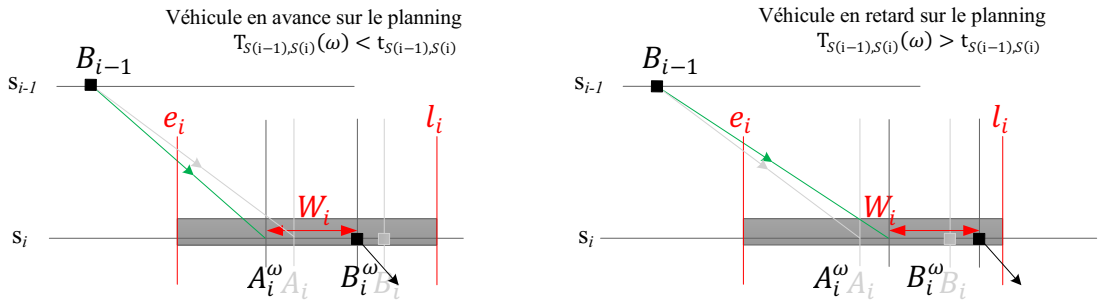


Figure 4.11 : Illustration dans le cas stochastique de l'hypothèse 2

Le pseudocode associé à cette hypothèse est détaillé dans la figure 4.12. L'avance (ou le retard) qu'il a obtenu durant le trajet est donc directement répercutée sur la date de départ. Si le temps de trajet entre les deux sommets est inférieur à l'espérance, c'est-à-dire inférieur au temps de trajet déterministe, alors $A_i^\omega < A_i$ et donc $B_i^\omega < B_i$. Inversement, si $A_i^\omega > A_i$ alors $B_i^\omega > B_i$.

```
1  // Bloc hypothèse H2
2  | B_i^ω := A^ω_i + W_i
```

Figure 4.12 : Lignes utilisées pour la simulation de l'hypothèse 2

Cette avance peut être un avantage pour compenser les retards futurs potentiels qu'il risque de rencontrer. Ceci peut aussi permettre d'améliorer le temps de transport des clients.

c) Hypothèse 3 : respecter les temps d'attente en cas d'avance sinon respecter les dates de début de service

Sous cette hypothèse, la feuille de route d'un chauffeur comprend à la fois les durées d'attente W_i et les dates de début de service B_i liées aux sommets qui composent la tournée. Le chauffeur doit alors respecter ce temps d'attente, s'il arrive en avance par rapport à la date prévue. Sinon, deux autres cas sont possibles : soit il arrive avant la date B_i prévue et alors il commence le service à la date B_i, soit il arrive après et alors il commence immédiatement à son arrivée. Le chauffeur se comporte donc comme dans l'hypothèse 2 si $A_i^\omega < A_i$ et comme l'hypothèse 1 si $A_i^\omega > A_i$. La figure 4.13 illustre ces deux comportements.

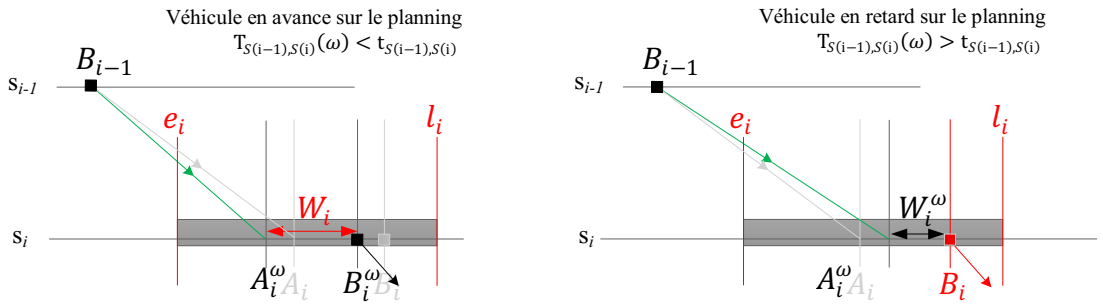


Figure 4.13 : Illustration dans le cas stochastique de l'hypothèse 3

Le pseudocode associé à cette hypothèse est détaillé dans la figure 4.14.

```
1 // Bloc hypothèse H3
2     |     if (A_i^ω < A_i) then
3     |     |     B_i^ω := A^ω_i + W_i
4     |     else if (A_i^ω ≤ B_i) then
5     |     |     |     B_i^ω := B_i
6     |     |     else if (A_i^ω > B_i) then
7     |     |     |     B_i^ω := A_i^ω
8     |     |     endif
9     |     endif
```

Figure 4.14 : Lignes utilisées pour la simulation de l'hypothèse 3

4.2.5. Conséquence des variations sur le temps de trajet pour le SDARP

Le retard ou l'avance pris par le véhicule peut entrainer des violations de contraintes sur le sommet courant mais aussi sur les sommets qui suivent dans la tournée. Pour le retard, il est évident que si le véhicule arrive après la fenêtre de temps du client alors le client n'est plus disponible. Mais prendre de l'avance peut aussi avoir des conséquences pour le SDARP, notamment à causes des contraintes de durée maximale de transport entre un nœud de *pickup* (le client est chargé dans le véhicule) et le nœud de destination (*delivery* node) où le client quitte le véhicule.

La figure 4.15 représente une tournée partielle dans laquelle le client A est traité en premier par le véhicule, suivi ensuite du chargement du client B. La troisième étape de la tournée consiste à déposer le client A sur son nœud de destination. Dans cette tournée, on peut supposer que des dates de passage ont été calculées sur chaque sommet et qu'elles correspondent aux dates de la figure 4.15, à savoir : le véhicule prend le client A à la date 25 puis le client B à la date 50 et enfin dépose le client A à la date 75.

La durée du trajet du client A entre son nœud de chargement et son nœud de destination est donc de 50. Si on suppose que le *maximal riding time* du client est de 55, la solution du problème de tournée respecte toutes les contraintes.

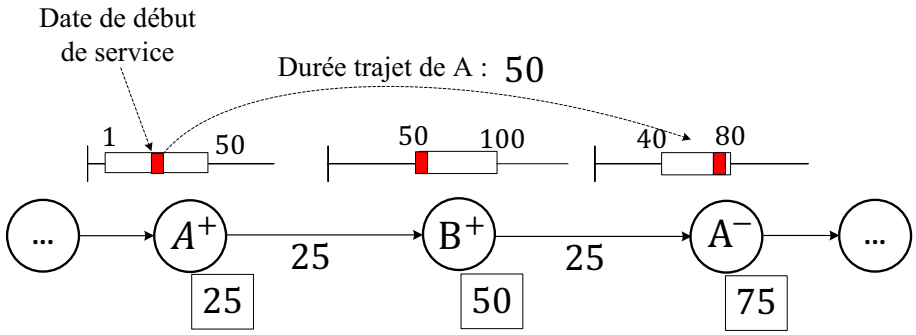


Figure 4.15: Violation des fenêtres de temps minimales

On peut envisager une politique de gestion des véhicules dans laquelle on autorise un véhicule à repartir plus tôt que la date prévue si les conditions de circulation ont permis d'arriver plus tôt sur un sommet. La figure 4.16 représente une réalisation de la tournée précédente dans laquelle le véhicule arrive pour récupérer le client A avec 10 minutes d'avance. Si le véhicule décide de le ramasser en avance, il repart alors à la date 15 et arrive pour récupérer le client B à la date 40. Malheureusement le client B n'est disponible qu'à partir de la date 50. Le véhicule ne peut pas repartir immédiatement, il attend donc 50 et arrive sur le sommet A^- pour déposer le client A à la date 75. La date d'arrivée de 75 sur le sommet de destination du client A est identique à la date initialement prévue. La différence entre les deux tournées est le temps de trajet du client 1 qui a augmenté de 10 minutes pour passer de 50 à 60 min. Il ne respecte donc plus la durée maximale prévue de 55 min.

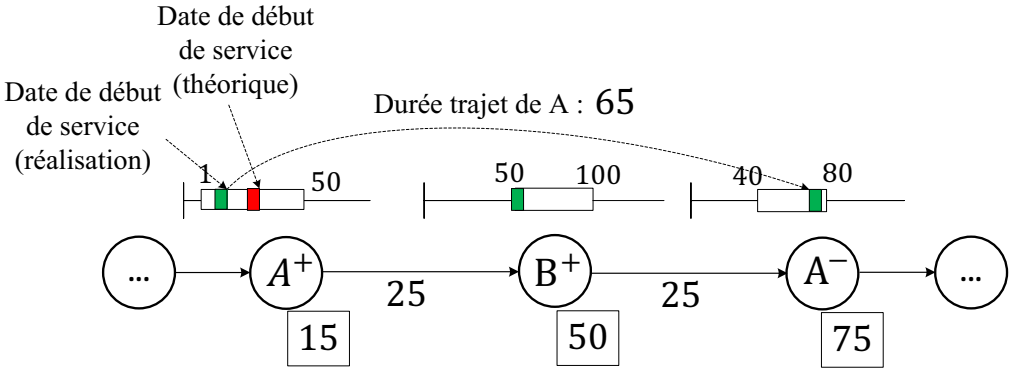


Figure 4.16: Violation des fenêtres de temps minimales

De multiples situations peuvent être envisagées. Si le véhicule ne profite pas de l'avance obtenue au début de la tournée, les retards qui vont s'accumuler dans la suite vont entrainer la violation de fenêtre de temps maximale des clients. D'autres fois, en prenant de l'avance, les temps de trajet des clients sont pénalisés. C'est pourquoi la suite du chapitre consiste à trouver, pour chaque consigne suivie par le chauffeur (chaque hypothèse), des solutions qui ont un coût intéressant mais qui en plus ont des probabilités de réalisabilité élevées.

4.3. Démarche de résolution générale du SDARP

4.3.1. Fonction-objectif et critères des solutions

A cause des temps de trajet stochastiques, à une solution S sont associées deux critères. Premièrement, le coût $c(S)$ de la distance parcourue par la flotte de véhicules. Ce coût est indépendant des variations sur le temps de trajet et il est aussi indépendant de l'hypothèse considérée. Il s'exprime comme :

$$\mathrm{c(S)} = \sum_{k=1}^{m} \left(\sum_{i=1}^{n_k - 1} c_{s_i^k s_{i+1}^k} \right)$$

avec n_k le nombre de sommets dans la $k^{ème}$ tournée. Le second critère est une notion de robustesse. Elle qui représente la probabilité $F_{Hx}(S)$ que la solution soit réalisable. Cette robustesse dépend de l'hypothèse considérée. Les tournées étant indépendantes dans le modèle, cette probabilité est égale au produit de la probabilité que chaque tournée soit réalisable. La probabilité $\mathrm{F_{Hx}}(t_i)$ qu'une solution soit réalisable, avec t_i une tournée de S, est comme suit :

$$\mathrm{F_{Hx}(S)} = \prod_{i=1}^{m} \mathrm{F_{Hx}}(t_i)$$

Comme la robustesse d'une solution $F_{Hx}(S)$ est difficile à calculer, un critère ρ^* associé à cette probabilité a été mis en place.

Le premier critère (le coût) doit être minimisé, alors que la robustesse est à maximiser. La robustesse étant une probabilité, sa valeur est comprise entre 0 et 1. Comme illustré dans la figure 4.17, à chaque hypothèse sur le comportement du véhicule correspond un espace de solutions Ω_{Hx}. Pour parcourir cet espace de solutions, un espace de codage intermédiaire Ω_β est utilisé. Cet espace est le même que celui utilisé pour le DDARP et il a été présenté dans le chapitre 2.

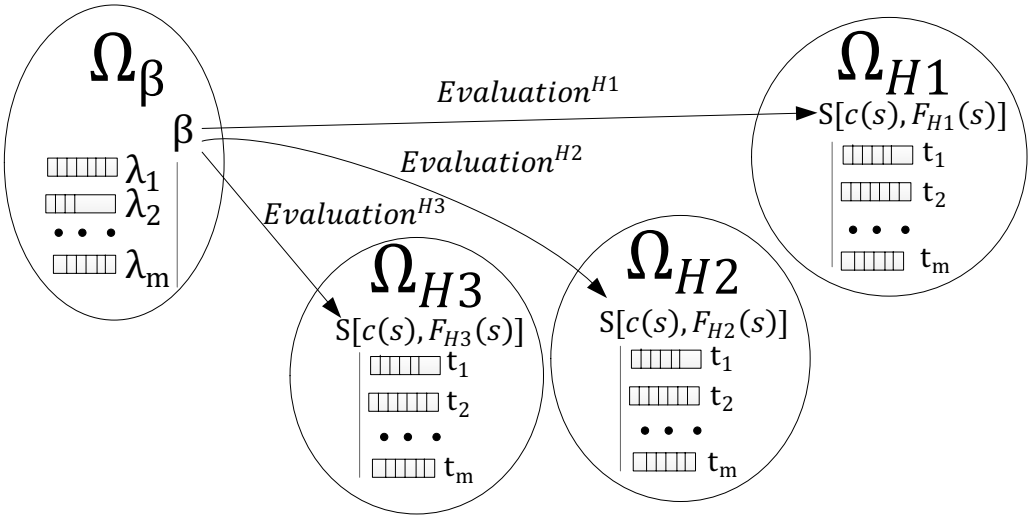


Figure 4.17 : Espace de codage et solutions

Un élément appartenant à Ω_β est nommé β. Il se compose d'un ensemble de m listes λ_i. Ces λ_i sont des listes ordonnées de sommets devant être visités par un véhicule.

Pour chaque hypothèse Hx, un élément de codage est associé à une solution S de Ω_{Hx} à l'aide d'une fonction dite d'évaluation. Le coût de ces solutions est identique. Par contre les dates de passage sur les sommets ainsi que la robustesse des solutions changent en fonction de l'hypothèse retenue.

4.3.2. Démarche de résolution pour le SDARP

La résolution du SDARP peut être divisée en deux parties. La première traite de l'évaluation d'un élément β du SDARP et la deuxième présente les différentes méthodes de résolution utilisées.

Concernant la première partie, trois grandes étapes sont développées:

- l'évaluation d'une solution du SDARP, qui nécessite de pouvoir tester la réalisabilité des différentes tournées et donc de définir ce qui est considéré comme "une contrainte valide" dans le cas d'un problème stochastique ;
- le calcul d'un critère de robustesse ρ^* associé à la solution. Ce critère dépend des critères ρ_M^i obtenus lors de l'évaluation des vecteurs λ_i ;
- le calcul de la robustesse d'une solution correspond à l'estimation de la probabilité de réalisabilité de S. Il peut être effectué avec deux méthodes : par calcul analytique ou par simulation.

Ces étapes sont illustrées dans la figure 4.18.

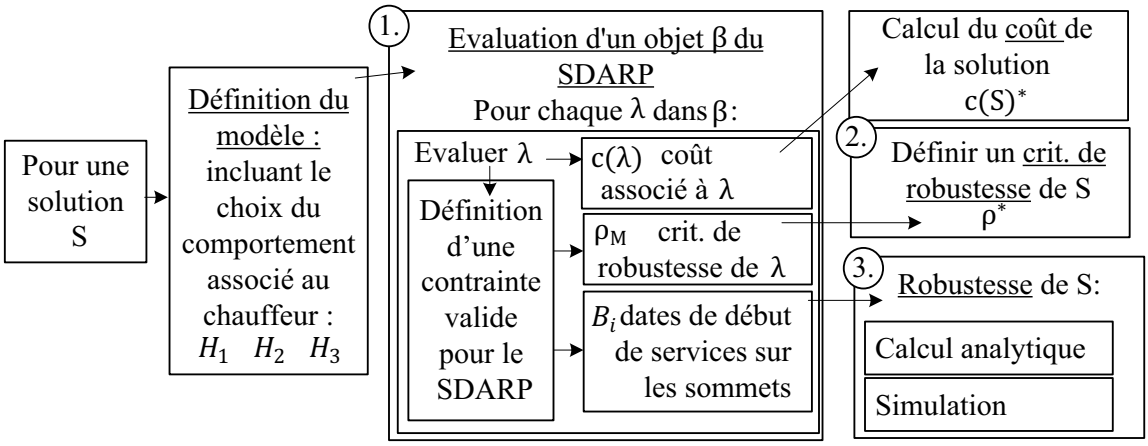


Figure 4.18 : L'évaluation d'une solution et le calcul de la robustesse

Dans la deuxième partie, deux méthodes de résolution ont été développées :

- une première méthode, dans laquelle la fonction objectif traite ces critères de manière hiérarchisée avec en premier la robustesse puis en second le coût. Cette méthode est basée sur le schéma de recherche locale évolutionnaire (ELS) comme proposé dans le chapitre 2 pour le DARP.
- une seconde méthode de résolution, cette fois multicritère, est proposée pour rechercher un front de Pareto sur ces deux critères comme illustré dans la figure 4.19

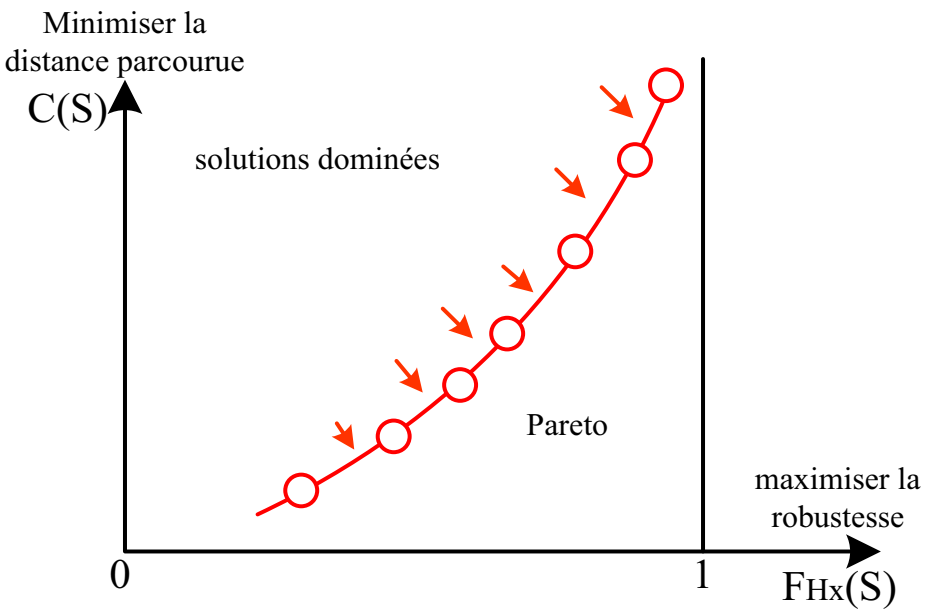


Figure 4.19 : Pareto des solutions multicritères recherchées

Ces parties seront suivies des résultats numériques obtenus avec les différentes méthodes sur des instances classiques de la littérature du DARP adaptées au SDARP.

4.4. Evaluation d'un objet β pour le SDARP

4.4.1. Rappel sur le DDARP

Pour le DDARP, une solution est obtenue en évaluant un objet β qui se compose d'un ensemble ordonné de vecteurs λ_i. L'évaluation consiste donc à vérifier plusieurs contraintes sur β ainsi que sur les vecteurs λ_i qui le composent. Ces contraintes sont les suivantes :

- la totalité des clients est traitée ;
- le nombre d'éléments λ_i dans β est inférieur ou égal au nombre de véhicules disponibles dans la flotte ;
- tous les λ_i sont valides.

Pour vérifier la troisième contrainte, la fonction d'évaluation "evaluation_S" de la figure 4.20 fait appel à une sous-fonction pour tester la validité de chaque λ_i. Cette sous-fonction est appelée "evaluation_T". Pour que λ soit valide, il doit respecter un certain nombre de contraintes:

- la charge transportée par le véhicule est inférieure ou égale à la capacité du véhicule tout au long de la tournée;
- les sommets d'origine et de destination d'un client doivent être dans la même tournée et le sommet d'origine doit se situer avant le sommet de destination ;
- λ est réalisable.

Un vecteur λ est réalisable s'il est possible de calculer des dates de passage sur les sommets de λ tels que la tournée correspondante respecte l'ensemble des contraintes sur les dates de service du problème, qui sont :

- la date de début du service sur chaque sommet est située dans la fenêtre de temps du sommet ;

- le temps de trajet du véhicule k est inférieur à la durée maximale T_k ;
- le temps de transport des clients est inférieur à la valeur maximale L.

Pour vérifier ces contraintes, la fonction fait appel à une sous-fonction nommée *is_feasible_tour()*. Si β est valide, alors une solution est obtenue. Elle se compose d'un ensemble de tournées et d'un coût qui représente la distance totale parcourue par la flotte. Cette fonction est illustrée dans la figure 4.20.

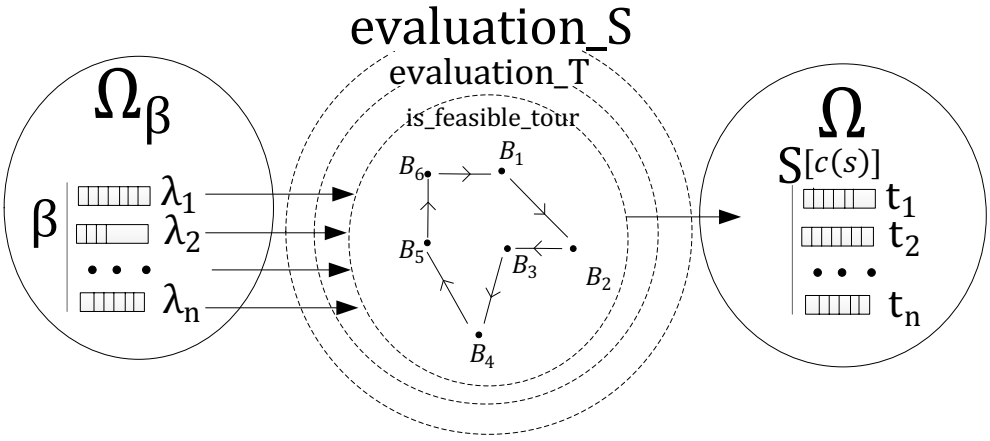


Figure 4.20 : Evaluation d'un objet β

La fonction $is\_feasible\_tour()$ affecte des dates de début de service B_i à l'ensemble des sommets d'une tournée. Il a été montré dans le chapitre 2 qu'il existe plusieurs fonctions d'évaluation. Chacune de ces fonctions retourne des dates B_i différentes.

Même si elles n'influencent pas le coût associé à la tournée, ces dates modifient des caractéristiques qui sont importantes pour la qualité du service proposé aux clients. Parmi les caractéristiques influencées par B_i, les trois principales sont la durée totale de la tounée, la somme des temps de trajet des clients et la somme des temps d'attente.

Pour illustrer ces influences, un exemple a été mis en place. Les données de cet exemple sont illustrées sur la figure 4.21. Le vecteur λ est composé de 4 sommets : le dépôt initial, le sommet d'origine de A, le sommet de destination de A, puis le dépôt final. Pour chaque sommet, des fenêtres de temps sont définies.

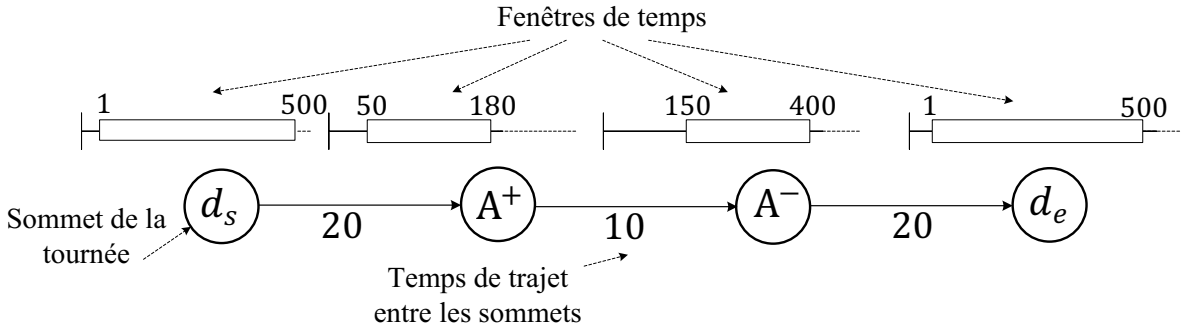


Figure 4.21: Données initiales sur le vecteur λ

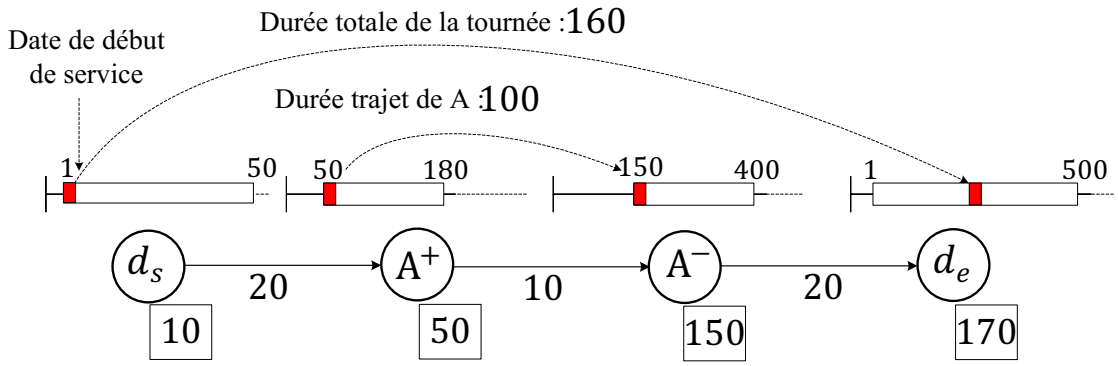


Figure 4.22: Evaluation avec les dates au plus tôt

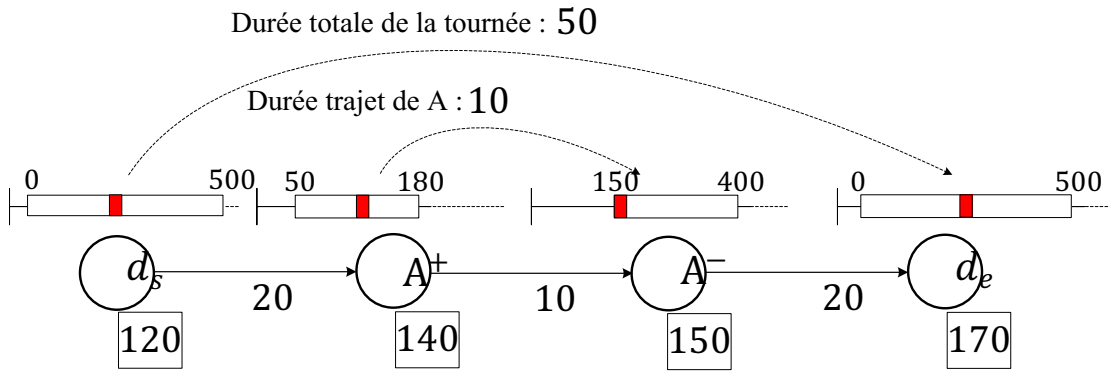


Figure 4.23: Dates obtenues avec l'évaluation de (Cordeau and Laporte, 2003)

Les caractéristiques des tournées de la figure 4.22 et de la figure 4.23 sont représentées. Dans la première, la durée totale de la tournée est de 160 unités de temps contre 50 dans la deuxième. Le temps de trajet du client A est de 100 unités dans la première contre seulement 10 pour la deuxième.

4.4.2. Conséquences sur le SDARP

Pour le SDARP, on utilise le même espace de codage que celui utilisé pour le DDARP. La fonction d'évaluation suit le même schéma que pour le DDARP, c'est-à-dire que quel que soit l'hypothèse Hx, la fonction $evaluation\_S()$ vérifie les contraintes sur les clients et sur le nombre de véhicules de la solution puis elle fait appel à une fonction $evaluation\_T()$ pour vérifier les contraintes sur chacune des tournées. Cette fonction $evaluation\_T()$ vérifie la contrainte sur la charge et l'ordre de traitement des clients puis utilise une sous-fonction $maximize\_\rho max\_Hx()$ pour vérifier la réalisabilité des tournées.

Les différences entre le DDARP et le SDARP apparaissent dans cette sous-fonction. Pour le DDARP, la réalisabilité d'une tournée est simple à définir car soit il existe des dates telles que toutes les contraintes sont vérifiées soit il n'en existe pas. Dans le cadre stochastique, il faut définir la notion de réalisabilité d'un λ_i. Contrairement au DARP, et sauf certaines situations, quelles que soit les dates de début de service B_i affectées aux sommets du vecteur λ_i, la tournée résultant a une probabilité non nulle d'être réalisable (cette probabilité pouvant être très proche de 0).

L'évaluation cherche alors, parmi toutes les dates de passage possibles, la tournée qui maximise sous hypothèse Hx un critère ρ introduit dans la suite de cette section. Pour cela, elle fait appel à la fonction $maximize\_\rho max\_Hx()$ qui elle-même fait appel à une sous-

fonction $is\_feasible\_tour\_Hx\_\rho()$, comme illustré sur la figure 4.24. Les fonctions sont présentées en détail dans la session 4.4.3. Mais avant, il est important de mettre en évidence l'influence des dates de passage et de l'hypothèse de gestion des véhicules sur la robustesse de la tournée.

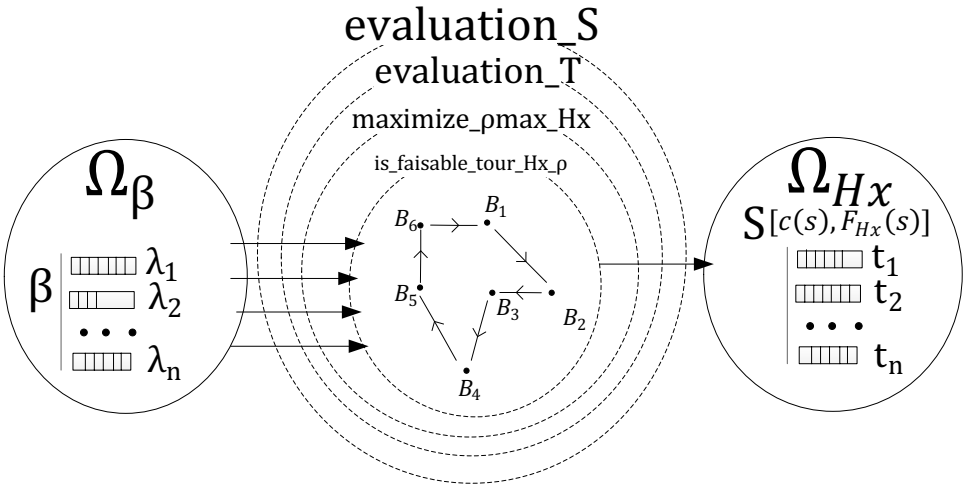


Figure 4.24 : Evaluation d'un objet β

a) Influence des valeurs affectées aux dates de passage

Soient deux tournées présentées dans la figure 4.25 : la tournée t_1, en rouge sur la figure, est obtenue avec la méthode de (Cordeau and Laporte, 2003) pour le DDARP. La tournée t_2, en vert, correspond aux dates précédentes mais avec un retard de 20 minutes sur chaque sommet par rapport au dates initialement planifiées. Ces deux tournées respectent toutes les contraintes imposées par le problème et de plus le temps de trajet du client *A* ainsi que la durée totale sont les mêmes dans les deux cas.

Concernant la réalisabilité de la tournée, des différences notables sur la position des dates de début dans la fenêtre de temps vont avoir un impact sur la probabilité que la tournée soit faisable.

Sur le sommet A^+, la date de début de service de t_2 (celle qui part du dépôt à 140) est plus proche de la fin de la fenêtre de temps que dans pour la tournée t_1(celle qui part du dépôt à 120). En cas de retard sur le temps de trajet entre le dépôt et le client A^+, la tournée t_2 peut violer la contrainte de fenêtre de temps maximale associée au client *A* alors que la tournée t_1 reste valable avec le même retard. Par contre, sur le sommet A^-, la tournée t_1 est moins robuste que la tournée t_2. En cas d'avance, la date de début de service de tournée t_1 est égale à la fenêtre de temps minimale sur le sommet A^-. Il est donc évident que la réalisabilité des deux tournées n'est pas la même selon les réalisations des variables aléatoires sur les temps de trajet et donc la robustesse de ces deux tournées est différente.

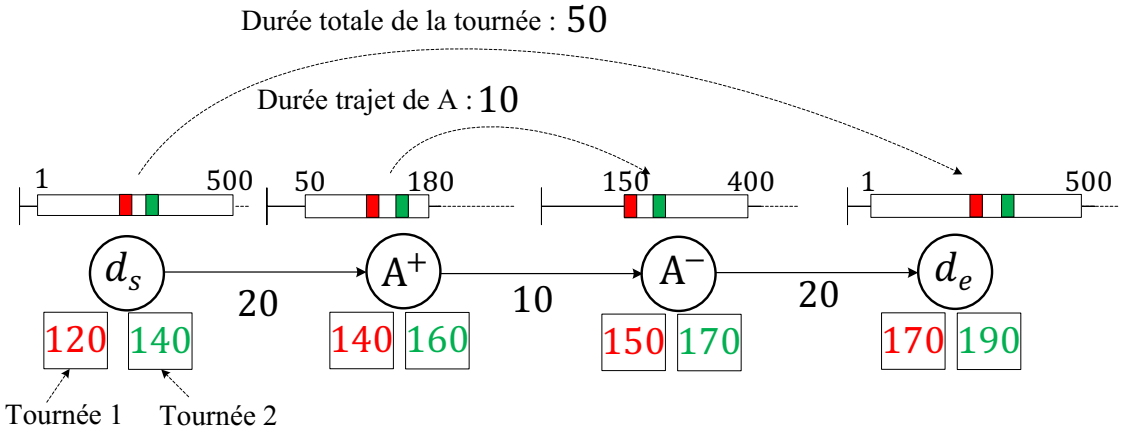


Figure 4.25: Influence des valeurs B_i sur la réalisabilité de la tournée

b) Influence de la politique suivie par le chauffeur

Supposons que la tournée soit construite à l'aide des dates au plus tôt obtenues pour le DDARP, comme illustré dans la figure 4.26. Dans cette tournée, les dates de début sur les sommets A^+ et A^- coïncident avec leurs fenêtres de temps minimales. Donc, pour être réalisable, le véhicule ne doit pas commencer son service en avance sur ces sommets.

On a deux hypothèses différentes :

- Sous l'hypothèse $H1$, le véhicule doit respecter les dates au plus tôt imposées. Donc s'il arrive avant, le chauffeur attend B_i avant de charger le client. Il respecte les fenêtres de temps minimales pour toutes les réalisations des temps de trajet. Exemple : si le temps de trajet entre d_s et A^+a pris 15 min au lieu de 20 min, le chauffeur attend 35 min au lieu de 30 et commence son service à 50.
- Sous l'hypothèse $H2$, le véhicule doit respecter les temps d'attente sur les sommets avant de commencer son service. Donc, si le temps de trajet entre d_s et A^+a pris 15 min au lieu de 20 min, le chauffeur attend 30 min comme il lui a été demandé et commence son service à la date 45. O, si le client n'est pas disponible avant la date 50, la tournée n'est pas réalisable.

Il paraît évident avec cet exemple que choisir l'hypothèse 2 pour une tournée évaluée avec les dates au plus tôt n'est pas un choix judicieux. Cet exemple illustre le fait que les politiques suivies par les chauffeurs ont une importance certaine sur la réalisabilité d'une tournée.

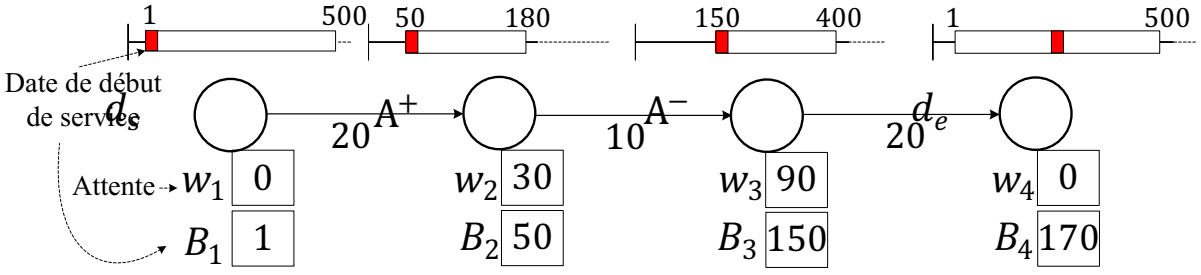


Figure 4.26: Influence de la politique suivie par le chauffeur sur la réalisabilité de la tourné

Dans la suite, l'évaluation d'un vecteur λ est détaillée, avant de chercher à estimer la valeur de la robustesse associée à une tournée. Cette valeur de robustesse est estimée en utilisant une fonction analytique ou en effectuant de la simulation. Un critère associé à la robustesse d'une tournée a été aussi mis au point. Pour finir, la robustesse de la solution est abordée.

c) Les solutions qui peuvent être qualifiées de non pertinentes

Comme expliqué pour le SDARP, toutes les affectations de B_i (ou presque) ont une probabilité non nulle d'être réalisables à cause des variations sur les temps de trajet. Par exemple la tournée illustrée sur la figure 4.27 n'est pas une solution du DDARP puisque les dates B_i representées en rouge sur la figure ne respectent pas les fenêtres de temps. Par contre, la probabilité que cette tournée soit réalisable n'est pas nulle pour le SDARP. Si lors de la réalisation Ω, le temps de trajet $T_{d_s A^+}(\Omega) = 50$ et $T_{A^+A^-}(\Omega) = 100$ alors la solution correspondra à la solution au plus tôt illustrée sur la figure 4.22 et la tournée associée à cette réalisation sera valide.

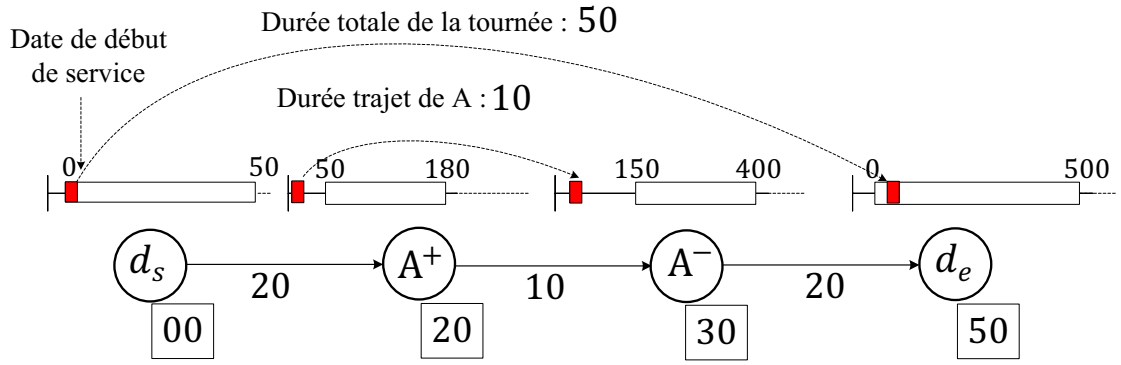


Figure 4.27: Evaluation avec les dates au plus tôt

Pour supprimer ces solutions qui peuvent être qualifiées de non pertinentes, une première limitation est imposée sur la réalisabilité d'une tournée :

Une tournée du SDARP doit être valide pour le DDARP. Par conséquent, une solution du SDARP doit être une solution valide du DDARP

4.4.3. La validation des contraintes pour le problème stochastique

a) Modélisation

La fonction d'évaluation utilisée dans le chapitre 2 a été adaptée pour le SDARP. Elle utilise un test de réalisabilité des tournées basé sur celui proposé par (Firat and Woeginger, 2011). Les notations suivantes sont nécessaires :

- γ_i : temps déterministe de trajet pour le véhicule dans la tournée entre le dépôt et le sommet s_i avec $i \in \{1, \dots, nk\}$ tel que $\gamma_i = \sum_{k=1}^{i-1} t_{s_k, s_{k+1}}$;
- X_i : variable de temps d'attente totale depuis le début de la tournée jusqu'au début du service sur le sommet s_i avec $i \in \{1, \dots, nk\}$, $\mathrm{X}_i = \sum_{k=1}^{i} \mathrm{W_k}$;
- Γ_i : variable aléatoire associée au temps déterministe de trajet pour le véhicule entre le sommet s_1 et le sommet s_i avec $i \in \{1, \dots, nk\}$;
- $\Gamma_i(\omega)$: valeur d'une réalisation de la variable aléatoire Γ_i, avec $i \in \{1, \dots, nk\}$

La variable aléatoire Γ_i est associée à la somme des temps de trajet entre le dépôt et les autres sommets de la tournée $\Gamma_\mathrm{i} = \sum_{k=1}^{i-1} T_{s_k, s_{k+1}}$. Comme chaque temps de trajet entre deux sommets s_i et s_j est représenté par une loi normale $\mathcal{N}(t_{s_i s_j}, \sigma_{s_i s_j})$ et que les temps de trajet sont indépendants dans le modèle, la propriété d'additivité des lois normales peut être utilisée.

La propriété d'additivité des lois normales est comme suit : soit $\mathcal{N}(x_1, \sigma_1)$ et $\mathcal{N}(x_2, \sigma_2)$ deux variables aléatoires indépendantes de lois normale, la somme de ces deux lois est elle-même une variable aléatoire qui suit une loi normale d'espérance x_1+x_2 et de variance $\sqrt{{\sigma_1}^2 + {\sigma_2}^2}$.

En appliquant plusieurs fois la propriété d'additivité des lois normales pour tous les arcs de la tournée entre s_1 et s_i, on en déduit que la variable aléatoire Γ_i suit une loi normale d'espérance $\gamma_i = \sum_{k=1}^{i-1} t_{s_k,s_{k+1}}$ et de variance $\sigma_i = \sqrt{\sum_{k=1}^{i-1} \left(\frac{t_{s_i,s_i+1}}{10}\right)^2}$.

b) Rappel sur les contraintes d'une tournée du DDARP

Dans le chapitre 2, des algorithmes permettant de tester la réalisabilité des tournées ont été présentés pour le DDARP. Une tournée est considérée comme réalisable s'il existe une valeur B_i sur chaque sommet telle que toutes les contraintes de la tournée soient satisfaites. Ces algorithmes permettent en plus d'obtenir les valeurs A_i, W_i, B_i, D_i qui définissent la tournée. Rappelons que, d'après la définition de (Cordeau and Laporte, 2003), les contraintes sur les dates de début de services sont :

- $B_i \geq e_i$: contrainte de fenêtre de temps minimale sur le $i^{ème}$ sommet de la tournée ;
- $B_i \leq l_i$: contrainte de fenêtre de temps maximale sur le $i^{ème}$ sommet de la tournée ;
- $B_{n_k} - D_1 \leq T_k$: contrainte sur la durée maximale de la tournée ;
- $B_j - D_i \leq L$: contrainte sur le temps de transport maximal pour le client, où s_i et s_j sont les sommets de *pickup* et de *delivery* d'un client.

Ceci peut se résumer à un ensemble de 4 contraintes :

$$B_i \geq e_i \quad (1)$$

$$B_i \leq l_i \quad (2)$$

$$B_j - D_i \leq L \quad (3)$$

$$B_{n_k} - D_1 \leq T_k \quad (4)$$

A l'aide d'un changement de variables qui utilise les égalités $B_i = X_i + \gamma_i$ et $D_i = B_i + d_{s_i}$, l'algorithme proposé par (Firat and Woeginger, 2011) détermine les valeurs des variables X_i qui respectent les contraintes. Les contraintes exprimées en utilisant les variables X_i deviennent :

$$(e_{s_i} - \gamma_i) \leq X_i \quad (1)$$

$$(l_{s_i} - \gamma_i) \geq X_i \quad (2)$$

$$(T_k - \gamma_{n_k}) \geq X_{n_k} - X_1 \quad (3)$$

$$(d_{s_i} + L - (\gamma_j - \gamma_i)) \geq X_j - X_i \quad (4)$$

Les variables A_i, B_i, W_i et D_i sont ensuite calculées en appliquant le changement de variables inverse sur les X_i.

4.4.4. Les contraintes "valides" dans le SDARP

Dans le SDARP, il est difficile de définir si une contrainte est satisfaite par des dates de passage sur les sommets car les temps de trajet sont des réalisations de variable aléatoire. Les valeurs associées à ces variables dépendent de la réalisation de la variable aléatoire Γ_i. Les inégalités doivent être réécrites en utilisant $\Gamma_i(\omega)$, ce qui correspond à :

$$(e_{s_i} - \Gamma_i(\omega)) \leq X_i \quad (1)$$

$$\left(l_{s_i} - \Gamma_i(\omega)\right) \geq X_i \quad (2)$$

$$\left(T_k - \Gamma_{n_k}(\omega)\right) \geq X_{n_k} - X_1 \quad (3)$$

$$(d_{s_i} + L - (\Gamma_j(\omega) - \Gamma_i(\omega))) \geq X_j - X_i \quad (4)$$

Comme ces inégalités dépendent de $\Gamma_i(\omega)$, la validité des contraintes pour une valeur de X_i dépend des réalisations des variables aléatoires pour la réalisation ω. Les contraintes ont donc une certaine probabilité d'être réalisables. Cette probabilité dépend de la variable aléatoire Γ_i, de la valeur affectée aux variables X_i, mais aussi de l'hypothèse sur le comportement du véhicule, notée H_k avec $k \in \{1; 2; 3\}$.

$$P_{Hk}(C_1) = P_{Hk}\left(\left(e_{s_i} - \Gamma_i\right) \leq X_i\right) \quad (1)$$

$$P_{Hk}(C_2) = P_{Hk}\left(\left(l_{s_i} - \Gamma_i\right) \geq X_i\right) \quad (2)$$

$$P_{Hk}(C_3) = P_{Hk}\left(\left(T_k - \Gamma_{n_k}\right) \geq X_{n_k} - X_1\right) \quad (3)$$

$$P_{Hk}(C_4) = P_{Hk}\left(\left(d_{s_i} + L - \left(\Gamma_j - \Gamma_i(\omega)\right)\right) \geq X_j - X_i\right) \quad (4)$$

Dans le modèle du SDARP, une nouvelle définition est introduite pour caractériser ces contraintes qui possèdent une certaine probabilité d'être valide:

Définition (contrainte probabilisée d'ordre ρ): on appelle une contrainte probabilisée d'ordre p, une contrainte qui est vraie avec une probabilité ρ sous l'hypothèse H2. Elle est notée C^{ρ} et elle est telle que $P_{H_2}(C) > \rho$.

Donc, contrairement au DDARP, le résultat du test de réalisabilité ne dépend pas uniquement de la liste ordonnée des sommets. Il varie aussi en fonction du paramètre ρ choisi, qui représente le critère associé à la contrainte probabilisée.

L'hypothèse H_2 est utilisée car elle correspond à la situation où le véhicule doit respecter l'attente avant de repartir. La variable X_i est la somme des attentes. Elle n'est donc pas influencée par la date d'arrivée, ce qui n'est pas le cas dans les autres hypothèses. La figure 4.28 représente la densité de la loi Γ_i centrée sur la valeur de B_i qui correspond à la loi suivie par B_i^{VA} sous H_2. La probabilité $P_{H_2}(\Gamma_i + X_i \leq Cst)$ est alors représentée par l'aire hachée sous la courbe. Elle correspond à la valeur $\int_{-\infty}^{Cst-X_i} f_{\Gamma_i}(x)dx$.

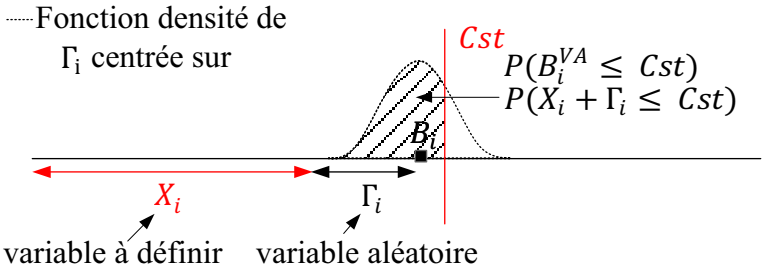


Figure 4.28 : Probabilité qu'une contrainte soit satisfaite

La variable aléatoire Γ_i représente la somme des temps de trajet. Comme ni le chauffeur ni l'entreprise n'a d'influence sur ces temps de trajet, il n'est pas possible de modifier la probabilité de respecter la contrainte. Donc, pour qu'une contrainte C_x soit valide, il faut définir une valeur à X_i telle que $P_{H_2}(C_x) > \rho$. Pour les contraintes de la forme $\Gamma_i - X_i \leq Cst$, plus la valeur de X_i est grande, plus Bi se rapproche de Cst et plus l'aire de la fonction de densité $B_i^{VA} < Cst$ diminue et donc plus la probabilité diminue. Au contraire, plus la valeur de X_i est petite et plus la probabilité augmente, *i.e.* l'aire de la courbe de $B_i^{VA} < Cst$ augmente. Inversement pour les contraintes de la forme $\Gamma_i - X_i \geq \text{Cst}$.

Les contraintes deviennent alors :

$$P_{H_2}(C_1) \geq \rho \rightarrow P_{H_2}(\Gamma_i \geq e_{s_i} - X_i) \geq \rho \quad (1)$$

$$P_{H_2}(C_2) \geq \rho \rightarrow P_{H_2}(\Gamma_i \leq l_{s_i} - X_i) \geq \rho \quad (2)$$

$$P_{H_2}(C_3) \geq \rho \rightarrow P_{H_2}(\Gamma_{n_k} \leq T_k - (X_{n_k} - X_1)) \geq \rho \quad (3)$$

$$P_{H_2}(C_4) \geq \rho \rightarrow P_{H_2}(\Gamma_j - \Gamma_i \leq L + d_{s_i} - (X_j - X_i)) \geq \rho \quad (4)$$

Ensuite, comme la variable aléatoire qui modélise le temps de trajet suit une loi normale : $\Gamma_i \backsim N(\gamma_i, \sigma_i)$, elle peut être approximée par une loi normale centrée / réduite $N(0,1)$ avec la transformation suivante :

$$P_{H_2}(\Gamma_i \geq e_{s_i} - X_i) \geq \rho \Leftrightarrow P_{H_2}\left(\frac{\Gamma_i - \gamma_i}{\sigma_i} \geq \frac{e_{s_i} - X_i - \gamma_i}{\sigma_i}\right) \geq \rho \Leftrightarrow P_{H_2}\left(N(0,1) \geq \frac{e_{s_i} - X_i - \gamma_i}{\sigma_i}\right) \geq \rho$$

La transformation précédente peut être appliquée pour les quatre contraintes, en remarquant que pour la contrainte C_4, Γ_j et Γ_i sont des lois normales et que la soustraction de deux lois normales est aussi une loi normale, avec $\Gamma_j - \Gamma_i \backsim N\left(\gamma_j - \gamma_i, \sqrt{{\sigma_j}^2 + {\sigma_i}^2}\right)$.

Les quatre contraintes se réécrivent ainsi :

$$P_{H_2}(C_1) \geq \rho \rightarrow P_{H_2}\left(N(0,1) \geq \frac{e_{s_i} - X_i - \gamma_i}{\sigma_i}\right) \geq \rho \quad (1)$$

$$P_{H_2}(C_2) \geq \rho \rightarrow P_{H_2}\left(N(0,1) \leq \frac{l_{s_i} - X_i - \gamma_i}{\sigma_i}\right) \geq \rho \quad (2)$$

$$P_{H_2}(C_3) \geq \rho \rightarrow P_{H_2}\left(N(0,1) \leq \frac{T_k - (X_{n_k} - X_1) - \gamma_{nk}}{\sigma_{nk}}\right) \geq \rho \quad (3)$$

$$P_{H_2}(C_4) \geq \rho \rightarrow P_{H_2}\left(N(0,1) \leq \frac{L + d_{s_i} - (X_j - X_i) - (\gamma_j - \gamma_i)}{\sqrt{{\sigma_j}^2 + {\sigma_i}^2}}\right) \geq \rho \quad (4)$$

Pour la loi normale centrée réduite $N(0,1)$, il est possible d'obtenir la valeur de c telle que $\varphi(c) = \rho$, en utilisant la fonction de répartition de cette loi :

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} . dt$$

L'avantage de se ramener à la loi normale centrée réduite $N(0,1)$ est qu'une seule et même loi est utilisée quelle que soit la contrainte et quel que soit le sommet de la tournée. Il n'est donc plus utile de connaître les autres fonctions de répartition des lois Γ_i. Ainsi les valeurs de $\varphi(x)$ sont suffisantes et peuvent être prédéfinies dans l'algorithme pour améliorer les performances de la méthode de résolution.

En utilisant c, les inégalités de la forme $\mathrm{P}(\mathrm{N}(0,1) \leq \mathrm{y}) \geq \rho$ deviennent équivalentes à des inégalités de la forme $y \geq c$ et celles de la forme $\mathrm{P}(\mathrm{N}(0,1) \geq \mathrm{y}) \geq \rho$ sont équivalentes à des inégalités de la forme $y \leq -c$.

Cette équivalence peut être appliquée sur les quatre contraintes. Ainsi, pour la première contrainte :

$$\mathrm{P}_{H_2}\left(N(0,1) \geq \frac{e_{s_i} - X_i - \gamma_i}{\sigma_i}\right) \geq \rho \iff \frac{e_{s_i} - X_i - \gamma_i}{\sigma_i} \leq -c \iff e_{s_i} - X_i - \gamma_i \leq -c\sigma_i$$

$$\iff -X_i \leq -c\sigma_i - e_{s_i} + \gamma_i \iff X_i \geq c\sigma_i + e_{\lambda(i)} - \gamma_i$$

Les équivalences donnent pour les quatre contraintes :

$$\mathrm{P}_{H_2}(C_1) \geq \rho \Leftrightarrow e_{s_i} - \gamma_i + c\sigma_i \leq X_i \quad (1')$$

$$\mathrm{P}_{H_2}(C_2) \geq \rho \Leftrightarrow l_{s_i} - \gamma_i - c\sigma_i \geq X_i \quad (2')$$

$$\mathrm{P}_{H_2}(C_3) \geq \rho \Leftrightarrow \left(T_k - \gamma_{n_k}\right) - c\sigma_{n_k} \geq X_{n_k} - X_1 \quad (3')$$

$$\mathrm{P}_{H_2}(C_4) \geq \rho \Leftrightarrow L + d_{s_i} - \left(\gamma_j - \gamma_i\right) - c\sqrt{\sigma_j^2 + \sigma_i^2} \geq X_j - X_i \quad (4')$$

Notons que les contraintes (1), (2), (3) et (4) du SDARP sont équivalentes aux contraintes associées au DDARP dans le cas où $\rho = 0.5$ car la valeur de c telle que $\varphi(c) = \rho = 0.5$ est alors 0.

4.4.5. Test de réalisabilité d'une tournée : $is\_feasible\_tour\_Hx\_\rho$

La fonction $is_feasible_tour_Hx_\rho()$détermine si, pour une probabilité ρ donnée, une liste de sommets constitue une tournée valide. Pour les trois hypothèses considérées, deux tests de réalisabilités sont nécessaires, un pour l'hypothèse H_1, et un pour les hypothèses H_2 et H_3.

La différence entre ces deux tests provient du fait que sous l'hypothèse 1 les contraintes C de la forme $B_i^{VA} \geq Cst$ sont toujours valides si la valeur B_i est valide. La probabilité associée à cette contrainte ne dépend pas de la réalisation de la variable aléatoire Γ_i. Ce n'est pas le cas pour les deux autres hypothèses. C'est pourquoi les contraintes de la forme $B_i^{VA} \geq Cst$ pour le test de réalisabilité sous H_1 ne change pas par rapport au DDARP.

Le principe de cette fonction d'évaluation a été défini à l'origine par (Firat and Woeginger, 2011) pour le DDARP et il teste la réalisabilité d'un vecteur λ. Son fonctionnement a été décrit dans le chapitre 2, les contraintes sont premièrement modélisées sous la forme d'un graphe qui possède des propriétés spécifiques et qui est ensuite évalué pour affecter des valeurs aux variables X_i, dans le cas où le vecteur λ est réalisable. Puis, en utilisant ces variables X_i, les valeurs des autres variables de la tournée A_i, W_i, B_i et D_i sont calculées .

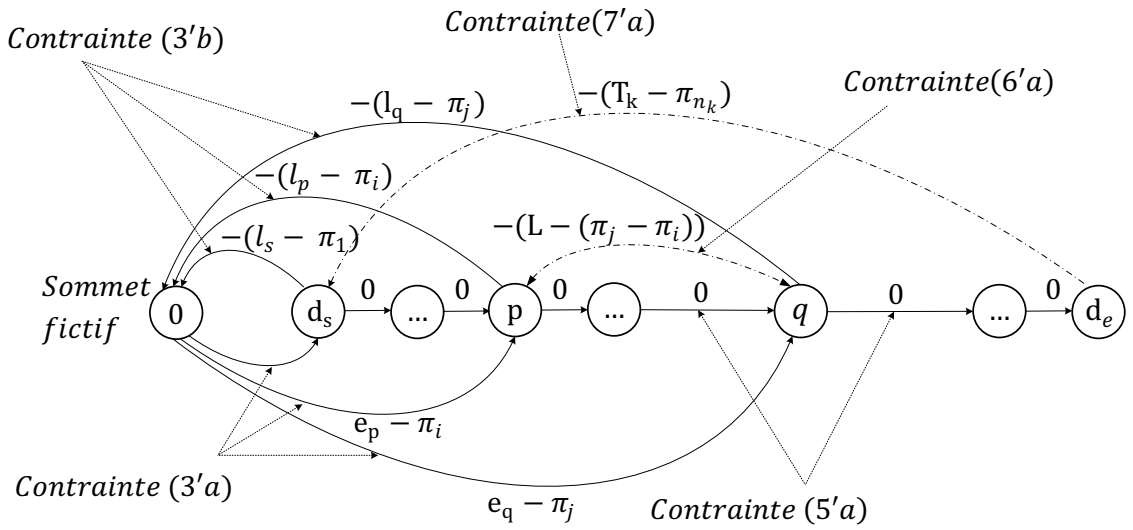


Figure 4.29: Graphe G_2 pour l'algorithme de (Firat and Woeginger, 2011)

La modification de l'algorithme est entièrement effectuée au niveau de la construction du graphe. Les valeurs des arcs qui composent le graphe sont calculées pour le DDARP en fonction des contraintes de précédences (1) (2) (3) (4). Pour le SDARP, les valeurs des arcs sont calculées en fonction des contraintes (1')/(1) (2') (3') (4') pour un paramètre ρ spécifique. Ce paramètre ρ est passé en paramètre à la fonction d'évaluation.

Tableau 4.4:
Contraintes utilisées dans la phase de construction du graphe dans is_feasible_tour_Hx_ ρ :

	contraintes sous H1	contraintes sous H2 et H3
$(3'a)$	$(e_{s_i} - \gamma_i) \leq X_i$	$e_{s_i} - \gamma_i + c\sigma_i \leq X_i$
$(3'b)$	$l_{s_i} - \gamma_i - c\sigma_i \geq X_i$	$l_{s_i} - \gamma_i - c\sigma_i \geq X_i$
(7'a)	$(T_k - \gamma_{n_k}) - c\sigma_{n_k} \geq X_{n_k} - X_1$	$(T_k - \gamma_{n_k}) - c\sigma_{n_k} \geq X_{n_k} - X_1$
$(6'a)$	$L + d_{s_i} - (\gamma_j - \gamma_i)$-$c\sqrt{\sigma_j^2 + \sigma_i^2} \geq X_j - X_i$	$L + d_{s_i} - (\gamma_j - \gamma_i)$-$c\sqrt{\sigma_j^2 + \sigma_i^2} \geq X_j - X_i$

Le paramètre c est calculé en utilisant la fonction de répartition de la loi normale centrée réduite :

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} . dt$$

Le but est de trouver la valeur de c telle que $\Phi(c) = \rho$. Pour ne pas avoir à calculer la valeur de la réciproque $\Phi^{-}1\ (\rho)$ à chaque appel de la fonction de réalisabilité, il est intéressant de stocker dans un tableau les valeurs de cette fonction pour différentes valeurs de ρ. La fonction $\varphi(x)$ est connue et ces valeurs $\Phi^{-}1\ (\rho)$ sont disponibles avec une grande précision.

La procédure est appelée dans la suite de ce chapitre " $is\_feasible\_tour\_Hx\_\rho$ ", représenté dans l'algorithme 1. Elle prend en paramètre un vecteur λ et la probabilité ρ utilisée pour le calcul des contraintes probabilisée d'ordre ρ . t se compose de la liste des sommets S_i ainsi que des différentes variables qui constituent la tournée : A_i, W_i, B_i, D_i. La probabilité ρ doit appartenir à [0,5;1]. Pour la valeur $\rho = 0.5$ la fonction obtenue est celle utilisée dans le problème du DDARP. La partie concernant la construction du graphe se situe à la ligne 5.

Algorithme 1 $is\_feasible\_tour\_Hx\_\rho$

Input parameters

ρ : minimal order imposed for a constraint

Global parameters

c_{ij} : distance between node i and j

Variable parameters

$stack$: Stack of integer
G_2 : graph with $n_T + 1$ nodes
$E[]$: vector for sets representation
$X[]$: value associated with each sets

Output parameters

$violation$: Boolean, $false$ if the trip is valid, $true$ else
t : a trip which contains all the trip variables like : n_t, s[], B[], A[], D[]

Begin

1 $\Pi[1] \coloneqq 0$ **// STEP 1 : compute new variables**
2 **for** $i \coloneqq 2$ **to** t**do**
3 | $\Pi[i] \coloneqq \Pi[i-1] + c_{t.s[i-1],t.s[i]}$
4 **end for**
5 $G_2 \coloneqq$ **build_Firat_graph_Stochastic(**$T[], \Pi[], \rho$**)** **// STEP 2 : build the graph**
6 **for** $i \coloneqq 1$ **to** t **do** **// STEP 3 : initialize the sets and variables**
7 $E[i] \coloneqq i; X[i] \coloneqq 0$;
8 **end for**
9 $violation := false$ **// STEP 4 : calculate the lowest X_i on nodes**
10 $stack := null$
11 **for** $i \coloneqq n_t$ **to** 1 **do**
12 | $X[i] \coloneqq \max_{j \in \delta_i}(X[\text{Find}(E,j)] + \text{Arc}(G_2, i, j))$
13 | **while** (stack_is_not_empty)&& (stop = $false$) **do**
14 | | $\{j\} \coloneqq$**all** pop(stack)
15 | | **if** $(X[i] \geq X[j])$ **then**
16 | | | Union(E,i,j)
17 | | **else**
18 | | | push$(stack, j)$
19 | | | stop:= $true$
20 | | **end if**
21 | **end while**
22 | **call** push$(stack,i)$
23 **end for**
24 **for** $i \coloneqq 1$ **to** n_t **do** **// STEP 5 : check violation and calculate B_i**
25 | $j \coloneqq \text{Find}(E, i)$
26 | **if** $(X[j] + Arc(G_2, i, 0) \geq 0)$ **then**
27 | | $violation = true$
28 | **end if**
29 | $t.B[i] \coloneqq X[i] + \Pi[i]$
30 **end for**
31 return $\{violation, t\}$

End

L'algorithme retourne un booléen qui vaut $true$ si des valeurs satisfaisant les contraintes probabilisées d'ordre p peuvent être affectées aux A_i et B_i, $false$ sinon. Il retourne aussi un objet t qui correspond à une tournée et qui contient toutes les variables, en particulier : A_i, W_i, B_i, D_i.

Un exemple d'appel à la fonction est le suivant :

$$(bool, t) \coloneqq is\_feasible\_Hx\_\rho\ (\lambda, \rho)$$

4.4.6. Schéma général d'évaluation du vecteur λ : $maximize\_\rho max\_Hx()$

L'objectif de cette procédure est de déterminer des valeurs pour B_i, A_i, D_i et W_i sur λ afin que l'ordre ρ_M de validation des contraintes soit le plus élevé possible :

$$\rho_M = \max_{\rho}\{\rho | is\_feasible\_Hx\_\rho\ (\lambda, \rho) = vrai\}$$

En d'autre termes, si λ est évalué avec une valeur ρ > ρ_M, alors il sera irréalisable. Il semble a priori difficile de définir méthode permettant de maximiser la valeur de ρ_M avec un seul appel à évaluation. Pour chaque contrainte C, il est par contre possible de déterminer la valeur $\rho = \rho_M^c$ qui permet de satisfaisant la contrainte C. Ces valeurs ne peuvent pas être utilisées pour calculer ρ^* car imposer une valeur ρ sur un sommet peut modifier la valeur ρ_M^c sur les sommets suivants.

Dans l'exemple de la figure 4.30, les valeurs de $\rho_M^{c1}, \rho_M^{c2}, \rho_M^{c3}$ sont données dans la partie de gauche et elles valent 100%. Pour chaque contrainte, il est possible de choisir une valeur X_i telle qu'une contrainte soit satisfaite avec une probabilité de 100%. Pourtant, lorsque la valeur ρ_M associée à toutes les contraintes est calculée, elle vaut seulement 55% comme illustré à droite sur la figure 4.30. Cette différence est due au fait que fixer une date de début sur un sommet aura des conséquences sur les sommets suivants. Dans l'exemple, choisir une valeur B_i plus élevée augmente la date d'arrivée sur le sommet j et diminue donc la probabilité de validé de la contrainte $C3$.

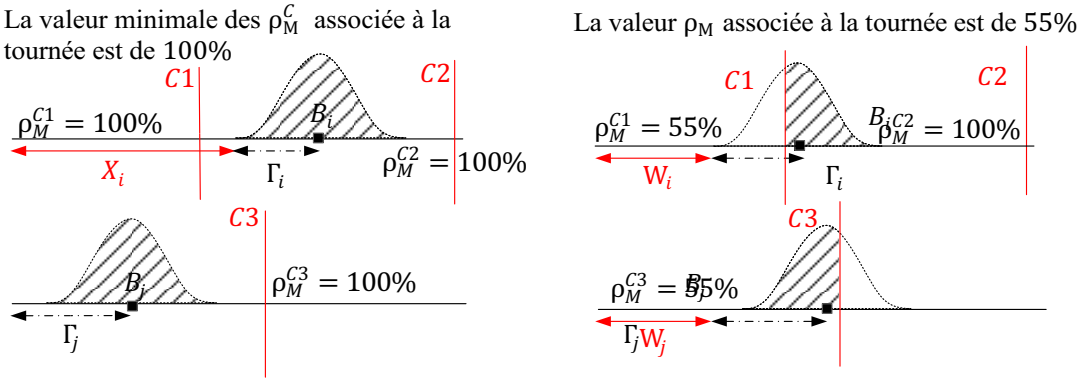


Figure 4.30 : Les difficultés pour déterminer la valeur du paramètre ρ

Pour trouver ρ_M, une recherche par dichotomie est mise en place. Cette recherche consiste à effectuer le test de réalisabilité des différentes tournées avec différents paramètres ρ afin de trouver la valeur maximale. L'algorithme 2 commence par évaluer la tournée en utilisant la fonction $is\_feasible\_Hx\_\rho$ avec ρ = mp où mp est la valeur maximale. Comme la valeur de $\Phi(x) = 1$ n'est pas finie, mp ne peut pas être fixée à 100%. Dans la pratique, elle est fixée à 99.95%. Si une tournée est valide pour mp, alors la fonction retourne $\rho_M = mp$ et s'arrête. Sinon la tournée est testée de nouveau avec $\rho = 0.5$ qui correspond au cas déterministe. Si la tournée n'est pas valide pour le cas déterministe alors elle ne pourra pas constituer une tournée valide pour le SDARP. L'algorithme retourne $F_t = false$, ligne 16. Si par contre, $\rho = 0.5$ est valide, il existe une valeur $\rho_M \in [0.5; mp]$ et une recherche dichotomique est utilisée pour la trouver.

Algorithme 2 $Maximize\_\rho\max\_Hx$

Input parameters:

λ	:	a node list

Global parameters:

$m\rho$	:	float ∈ [0.5,1[
delta	:	minimal gap to stop dichotomy

Output parameters:

F_t	:	a boolean of feasibility
ρ_M	:	float ∈ [0.5,1[
c	:	cost of the trip
t	:	a trip which contains all the trip variables like : n_t, s[], B[], A[], D[]

Begin

c = compute_cost (λ)
$test$ = Firat_feasibility_$SDARP$ $(\lambda, m\rho)$
if ($test$= false) **then**
  $(test, t) \coloneqq is\_feasible\_Hx\_\rho\ (\lambda, 0.5)$
  if ($test$= true) **then**
    $\rho_{max} \coloneqq \rho_M$;
    $\rho_{min} \coloneqq 0.5$
    while $\left((\rho_{max} - \rho_{min}) > delta\right)$ **do**
      $\rho := (\rho_{max} - \rho_{min})/2$
      $(test, t') \coloneqq is\_feasible\_Hx\_\rho\ (\lambda, \rho)$
      if ($test$= true) **then** $\rho_{min} \coloneqq \rho$; $t = t'$
      else $\rho_{max} \coloneqq \rho$
    end do
    $test = true;\ \rho_M = \rho_{min}$; return $\{test, \rho_M, \text{c}, \text{t}\}$
  else
    $test = false;\ \rho_M = 0.0$; return $\{test, \rho_M, \text{c}, \emptyset\}$
  end if
end if
$t = true;\ \rho_M = m\rho$; return $\{t, \rho, \text{c}, \text{t}\}$

End

En utilisant cette méthode, on associe à chaque tournée un coût $c(\lambda)$ et un paramètre $\rho_M \in [0.5,1[$. Le coût $c(\lambda)$ correspond à la somme des distances parcourues par la flotte de véhicules pour le DDARP, c'est-à-dire au critère à minimiser. ρ_M est le critère de réalisabilité associé à la tournée.

4.4.7. Définition du critère de robustesse d'une solution

La robustesse d'une solution est sa capacité à être réalisable malgré les variations sur le temps de trajet entre les sommets. Cela correspond à la probabilité que la tournée soit réalisable. Il est donc possible d'estimer la robustesse d'une solution à l'aide d'un calcul analytique, mais ce calcul analytique, comme expliqué dans le paragraphe 4.5.2 est trop coûteux en temps de calcul pour être intégré dans la méthode de résolution.

Une autre méthode consiste à trouver un critère suffisamment corrélé avec la robustesse de la solution mais beaucoup moins coûteux à calculer. L'objectif est qu'il donne une bonne estimation de la robustesse. On introduit un paramètre ρ^* associé à une solution. Il est définit comme suit :

$$\rho^* = \prod_{i=1}^{nk} \rho_M^i \ .$$

Ce critère ρ^* suppose que la probabilité qu'une tournée λ soit réalisable dépend fortement de la valeur de ρ_M, notée ρ_M^i, associée à une tournée λ_i. Par conséquent la robustesse est aussi liée à l'ensemble des ρ_M^i.

Le produit est utilisé car les ρ_M^i sont définis sur $[0.5; 1[$ et ainsi ρ^* est définie sur $]0; 1[$. Ce sont des bornes élémentaires simples pouvant être utilisées dans des fonctions objectif hiérarchisées. De plus le produit permet de prendre en compte toutes les tournées.

L'hypothèse faite dans ce modèle est que générer des solutions robustes peut être fait en maximisant ρ^*, même si ce paramètre n'est pas strictement équivalent à la probabilité que la solution soit réalisable. L'objectif de la partie suivante est de calculer la robustesse de la solution, c'est-à-dire la probabilité qu'une solution soit réalisable pour les différentes hypothèses envisagées.

4.5. Estimation de la probabilité de réalisabilité d'une solution

Comme expliqué précédemment, à cause des variations sur les temps de trajet, une solution a une certaine probabilité d'être réalisable pour chaque hypothèse considérée H_1, H_2, H_3. Cette probabilité représente la robustesse de la solution. Elle dépend de la probabilité de chaque tournée qui la compose. Notons que cette probabilité de réalisabilité ne correspond pas au critère ρ^* associé à une solution. Ce dernier est uniquement un critère de réalisabilité. Dans cette section, on recherche par simulation un estimateur de la probabilité de réalisabilité associée à une solution.

Comme les tournées peuvent être considérées indépendantes dans le modèle, la probabilité de réalisabilité de la solution est donc égale au produit des probabilités de réalisabilité des tournées qui la composent :

$$F_{Ht}(S) = \prod_{i=1}^{k} P_{Ht}(\lambda_i)$$

Dans cette équation λ_i est la $i^{ème}$ tournée de la solution S et $Ht \in \{H_1, H_2, H_3\}$. La probabilité qu'une tournée soit réalisable peut être calculée analytiquement en utilisant les lois et leur fonction de densité associée. Elle peut aussi être estimée à l'aide de simulations.

4.5.1. Calcul de la probabilité associée à une contrainte pour les différentes hypothèses

Dans la partie précédente, les contraintes probabilisées d'ordre ρ ont étés définies. Une contrainte C est dite probabilisée d'ordre ρ si la probabilité $P_{H_2}(C) \geq \rho$. Cette probabilité $P_{H_2}(C)$ est utilisée car elle peut être facilement calculée durant l'évaluation $is\_feasible\_Hx\_\rho()$. Comme indiqué dans le paragraphe suivant, elle représente une borne inférieure de la probabilité que la contrainte soit valide pour les autres hypothèses, *i.e.* $P_{H_2}(C) \leq P_{H_1}(C)$ et $P_{H_2}(C) \leq P_{H_3}(C)$. Donc si $\rho < P_{H_2}(C)$ alors $\rho < P_{H_2}(C)$, et $\rho < P_{H_3}(C)$.

Une fois les variables de la tournée déterminées, il est possible de calculer les différentes probabilités $P_{H_1}(C)$ et $P_{H_2}(C)$ et $P_{H_3}(C)$. Le calcul de chacune de ces probabilités est détaillé dans la suite, en commençant par $P_{H_2}(C)$. Elles sont nécessaires pour calculer analytiquement la robustesse associée à la solution.

a) Calcul de la probabilité d'une contrainte d'être valide sous l'Hypothèse H_2

Sous l'hypothèse 2, les temps d'attente W_i sont imposés au chauffeur pour chaque sommet de la tournée et quelle que soit la réalisation ω de Γ_i : $W_i^\omega = W_i$. La valeur X_i sous l'hypothèse H_2 est par conséquent une constante valant $X_{i-1} + W_i$. La variable aléatoire $B_i^{VA} = X_i + \Gamma_i$ suit une loi normale centrée sur $B_i = X_i + \gamma_i$ dont la variance est la même que Γ_i, *i.e.* σ_i. La probabilité associée à la variable aléatoire de la contrainte C, de la forme $P(B_i \leq Cst)$, peut être calculée en utilisant une loi normale centrée réduite. Pour le calcul de cette probabilité, seule X_i et la variance de la loi suivie par Γ_i sont nécessaires.

$$P(B_i^\omega \leq Cst) = P\left(N(0,1) \leq \frac{Cst - X_i - \gamma_i}{\sigma_i}\right) = \varphi\left(\frac{Cst - B_i}{\sigma_i}\right)$$

Sur la figure 4.31 la fonction de densité $f_{\Gamma_i}(x)$ centrée sur la valeur B_i est tracée. La probabilité que la contrainte soit valide est égale à 65%, ce qui représente l'aire de la courbe hachurée.

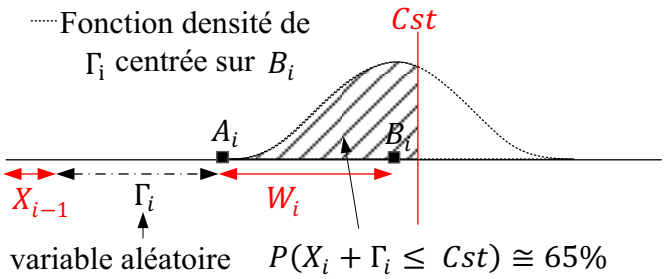


Figure 4.31 : Probabilité qu'une contrainte soit valide sous l'hypothèse 2

Le même calcul peut être réalisé pour les contraintes de la forme : $P(X_i + \Gamma_i \geq Cst)$

b) Calcul de la probabilité d'une contrainte d'être valide sous l'Hypothèse H_1

Sous l'hypothèse 1, le véhicule peut modifier l'attente sur le sommet afin de respecter des dates de début B_i. Contrairement à l'hypothèse précédente, $W_i^\omega \neq W_i$. De fait, X_i n'est pas constante, sa valeur dépend de W_i^ω, ce qui peut se noter $X_i^\omega = X_{i-1} + W_i^\omega$. La loi suivie par B_i^{VA} n'est pas une loi normale comme dans H_2. Par contre la probabilité qu'une contrainte C soit valide est calculable.

Pour les contraintes de la forme $B_i^{VA} \geq Cst$, la politique de gestion suivie dans l'hypothèse 1 implique que $P_{H_1}(B_i \geq Cst) = 1$ si la valeur de B_i est valide car le véhicule doit attendre B_i pour commencer son service, donc $B_i^\omega \geq B_i$.

Pour les contraintes de la forme $B_i^{VA} \leq Cst$, les consignes imposées par l'hypothèse 1 impliquent d'abord que $B_i^\omega = B_i$ si $A_i^\omega < B_i$ et que $B_i^\omega = A_i^\omega$ si $A_i^\omega > B_i$. Par conséquent $B_i^\omega \leq Cst$ est vraie uniquement si $A_i^\omega \leq Cst$. Donc $P(B_i^{VA} \leq Cst) = P(A_i^{VA} \leq Cst)$

Ensuite, si la loi suivie par A_i^{VA} est connue, alors X_{i-1} est une constante puisque les autres contraintes ne sont pas prises en compte. A_i^{VA} suit donc une loi normale centrée sur $A_i = X_{i-1} + \gamma_i$ dont la variance est la même que Γ_i, *i.e.* σ_i. La probabilité $P_{H_1}(A_i^{VA} \leq Cst)$ est connue en se rapportant à une loi centrée/réduite. Elle est donnée par :

$$P_{H_1}(B_i^{VA} \leq Cst) = P_{H_1}(A_i^{VA} \leq Cst) = P\left(N(0,1) \leq \frac{Cst - (X_{i-1_i} + \gamma_i)}{\sigma_i}\right) = \varphi\left(\frac{Cst - A_i}{\sigma_i}\right)$$

Pour le calcul de $P_{H_1}(C)$, les valeurs X_{i-1} et σ_i sont nécessaires. Deux exemples sont donnés dans la figure 4.32 pour la même valeur X_i mais avec des valeurs W_i différentes.

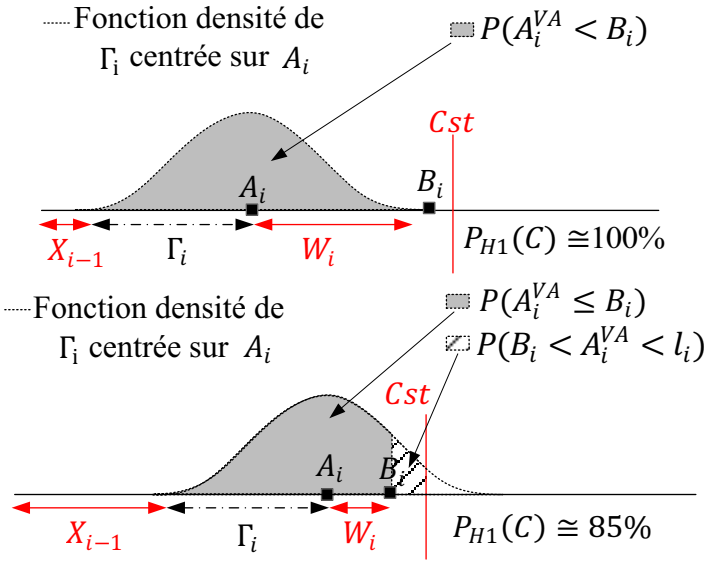


Figure 4.32 : Probabilité qu'une contrainte soit valide sous H_1 avec $W_i > 0$

Comme illustré dans la figure 4.33, si $W_i = 0$ alors $A_i = B_i$ et donc la même probabilité que sous l'hypothèse H_2 est obtenue, soit 65%.

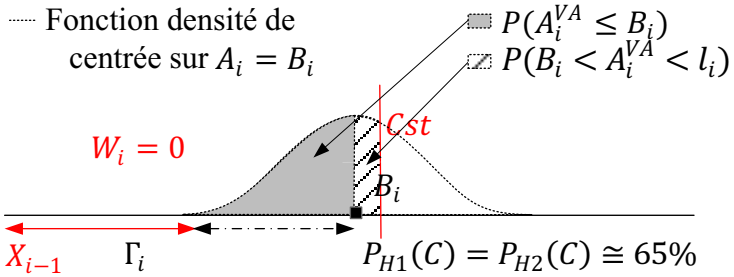


Figure 4.33 : Probabilité qu'une contrainte soit valide sous H_1 avec $W_i = 0$

Concernant le fait que $P_{H_2}(C)$ représente une borne inférieure de $P_{H_1}(C)$:

- Pour les contraintes de la forme $B_i \geq Cst$, comme $P_{H1}(B_i \geq Cst) = 1$ alors $P_{H_1}(B_i \geq Cst) \geq P_{H_2}(B_i \geq Cst)$.
- Pour les contraintes de la forme $B_i \leq Cst$ comme $W_i, \leq B_i \; \forall W_i$ alors $\varphi\left(\frac{Cst-\mathrm{A_i}}{\sigma_i}\right) \geq \varphi\left(\frac{Cst-\mathrm{B_i}}{\sigma_i}\right)$ donc $P_{H_1}(C) \geq P_{H_2}(C)$

c) Calcul de la probabilité d'une contrainte d'être valide sous l'Hypothèse H_3

Sous l'hypothèse 3, le chauffeur doit connaître à la fois W_i et B_i pour chaque sommet. Les consignes sont que, s'il arrive sur le sommet avant la date A_i, alors il doit respecter l'attente W_i. S'il arrive après A_i, il doit respecter la date de début du service B_i. La situation est alors assez similaire à l'hypothèse 1 : le temps d'attente sur i dépend de la réalisation ω, X_i n'est donc pas une constante, sa valeur est notée $X_i^{\omega} = X_{i-1} + W_i^{\omega}$ et dépend de W_i^{ω}. La loi suivie par B_i^{VA} n'est pas une loi normale. Par contre la probabilité qu'une contrainte C soit valide est là-aussi calculable.

Pour les contraintes de la forme $B_i \geq Cst$ on a $P_{H_3}(B_i \geq Cst) = P_{H_2}(B_i \geq Cst)$ car cette contrainte ne peut être violée que si $\Gamma_i(\omega) < \gamma_i$. Donc le véhicule a de l'avance et la consigne que doit appliquer le chauffeur est la même que sous H_2 : il doit attendre W_i avant de commencer son service et $P_{H3}(B_i^{VA} \geq Cst) = P_{H2}(B_i^{VA} \geq Cst)$.

Pour les contraintes de la forme $B_i^{VA} \leq Cst$, les consignes imposées par l'hypothèse 3 impliquent tout d'abord que si $A_i^{\omega} < A_i$ alors $B_i^{\omega}=A_i^{\omega} + W_i$, si $A_i < A_i^{\omega} < B_i$ alors $B_i^{\omega}=B_i$ et si $A_i^{\omega} > B_i$ alors $B_i^{\omega}=A_i^{\omega}$. Par conséquent et comme pour H_1, $B_i^{\omega} \leq Cst$ est vrai uniquement si $A_i^{\omega} < Cst$. Donc $P(B_i^{VA} \leq Cst) = P(A_i^{VA} \leq Cst)$

Ensuite, la loi suivie par A_i^{VA} est connue. C'est une loi normale centrée sur $A_i = X_i - W_i + \gamma_i$ de même variance que Γ_i, *i.e.* σ_i. Donc la probabilité $P_{H_3}(A_i^{VA} \leq Cst)$ est connue en se rapportant à une loi centrée réduite. Elle est donnée par :

$$P_{H_3}(B_i^{VA} \leq Cst) = P_{H_3}(A_i^{VA} \leq Cst) = P\left(N(0,1) \leq \frac{Cst - (X_{i-1} + \gamma_i)}{\sigma_i}\right) = \varphi\left(\frac{Cst - A_i}{\sigma_i}\right)$$

Ces probabilités sont illustrées sur la figure 4.34 et la figure 4.35 pour différentes valeurs de W_i.

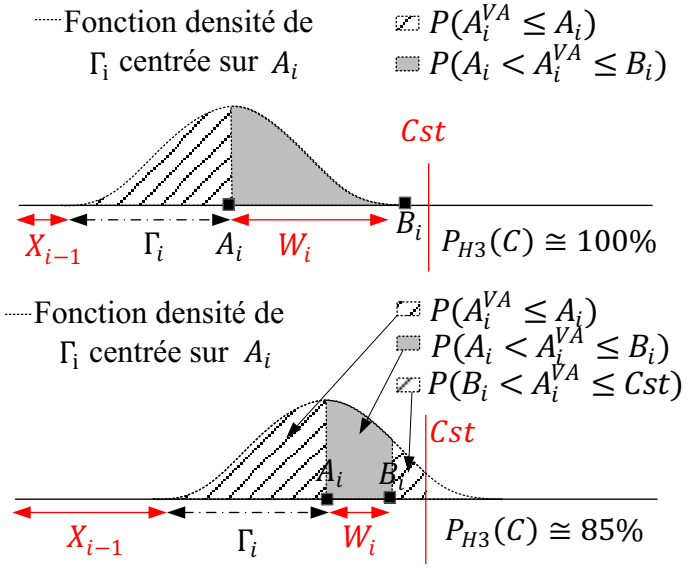


Figure 4.34 : Probabilité qu'une contrainte soit valide sous H_3 avec $W_i > 0$

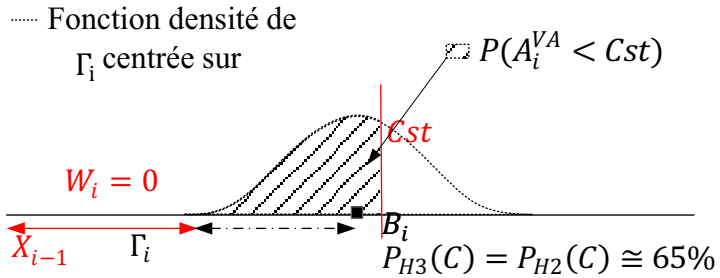


Figure 4.35 : Probabilité qu'une contrainte soit valide sous H_3 avec $W_i = 0$

Concernant le fait que $P_{H_2}(C)$ représente une borne inférieure pour $P_{H_3}(C)$:

- Pour les contraintes de la forme $B_i^{VA} \geq Cst$ comme $P_{H_3}(B_i \geq Cst) = P_{H_2}(B_i \geq Cst)$.

- Pour les contraintes de la forme $B_i^{VA} \leq Cst$ comme $A_i \leq B_i \; \forall W_i$ alors $\varphi\left(\frac{Cst-\mathrm{A_i}}{\sigma_\mathrm{i}}\right) \geq \varphi\left(\frac{Cst-\mathrm{B_i}}{\sigma_\mathrm{i}}\right)$ et donc $P_{H_3}(C) \geq P_{H_2}(C)$

4.5.2. Calcul analytique de la probabilité de réalisabilité de la tournée

Le calcul analytique est complexe dans la mesure où une tournée est composée d'un ensemble de sommets et que, sur chacun de ces sommets S_i, la variable B_i doit respecter un ensemble de contraintes qui a été défini précédemment.

Pour le calcul de la probabilité, une nouvelle variable aléatoire E_i avec $i \in \{1, \dots, n_k\}$ est introduite. Elle correspond au respect par B_i de l'ensemble des contraintes qui lui sont associées. Pour qu'une tournée soit réalisable, il faut que l'ensemble des E_i, $i \in \{1, \dots, n_k\}$ soit valide. La probabilité de réalisabilité de la solution est alors égale à la probabilité :

$$P_{Ht}(\lambda) = P(E_1 \cap E_2 \cap E_3 \cap E_4 \cap E_5 \cap E_6 \cap E_7 \cap E_8 \cap E_9 \cap \dots \cap E_{nk})$$

Deux cas concernant le calcul de la probabilité analytique d'une tournée sont étudiés dans cette partie :

- Soient les $\mathrm{E_i}$ sont considérés indépendantes : $\forall(\mathrm{i,j}) \in \{1; \dots; n_k\}^2 \; \mathrm{P}(\mathrm{E_i} \cap \mathrm{E_j}) = \mathrm{P}(\mathrm{E_i})\mathrm{P}(\mathrm{E_j})$, ce qui est une simplification du modèle ;
- Soient les E_i ne sont pas considérés indépendantes : $\forall(\mathrm{i,j}) \in \{1; \dots; n_k\}^2 \; \mathrm{P}(\mathrm{E_i} \cap \mathrm{Ej}) = \mathrm{P}(\mathrm{E_i})\mathrm{P}(\mathrm{E_j}|\mathrm{E_i}) \neq \mathrm{P}(\mathrm{E_i})\mathrm{P}(\mathrm{E_j})$.

Pour pouvoir calculer la probabilité associée à une contrainte E_i, il faut être en mesure de connaître la loi associée à la variable B_i^{ω}. Ces deux cas sont détaillés dans les paragraphes suivants.

a) Avec l'hypothèse simplificatrice que les variables E_i sont indépendantes

Cette hypothèse est proche de celle faite dans la fonction d'évaluation où chaque contrainte est traitée indépendamment des autres. Sous cette hypothèse, chaque sommet de la tournée peut être traité indépendamment des autres. Contrairement à l'évaluation, pour calculer la probabilité de E_i , les contraintes qui concernent les mêmes sommets sont traitées en même temps.

Sous ces hypothèses, la loi suivie par la variable aléatoire A_i^{VA} est alors connue. Elle est égale à la somme constante des temps d'attente imposés sur les sommets précédents X_{i-1} à laquelle est ajoutée la valeur aléatoire Γ_i associée à la somme des trajets entre de dépôt et le sommet i. A_i^{VA} suit donc une loi normale centrée sur $A_i = X_{i-1} + \gamma_i$ dont la variance est la même que Γ_i, *i.e.* σ_i.

Le calcul de la probabilité de E_i dépend du type de sommet :

- Pour i sommet d'origine d'un client :

$$P(E_i) = P\left(B_i^{VA} \geq e_i\right) \cap P\left(B_i^{VA} \leq l_i\right) = P\left(e_i \leq B_i^{VA} \leq l_i\right)$$

$$\mathrm{P(E_i)} = P\left(B_i^{VA} \leq l_i\right) - \left(P\left(B_i^{VA} \leq e_i\right)\right)$$

- Pour j sommet de destination d'un client associé à un sommet d'origine i :

$$P\left(E_j\right) = P\left(B_j^{VA} \geq e_j\right) \cap P\left(B_j^{VA} \leq l_j\right) \cap P\left(\Gamma_\mathrm{j} - \Gamma_\mathrm{i} \leq L - X_j - X_i\right)$$

Concernant la probabilité $\mathrm{P}\left(\Gamma_\mathrm{j} - \Gamma_\mathrm{i} \leq \mathrm{L} - \mathrm{X_j} - \mathrm{X_i}\right)$, la variable aléatoire $\Gamma_\mathrm{j} - \Gamma_\mathrm{i}$ est considérée comme indépendante des autres contraintes du sommet j car la loi suivie

par $\Gamma_j - \Gamma_i$ n'est pas la même que celle suivie par B_j^{VA}.

$$P(E_j) = P(B_j^{VA} \geq e_j) \cap P(B_j^{VA} \leq l_j) P(\Gamma_j - \Gamma_i \leq L - X_j - X_i)$$

$$P(E_j) = \left(P(B_j^{VA} \leq l_j) - P(B_j^{VA} \leq e_j)\right) P(\Gamma_j - \Gamma_i \leq L - X_j - X_i)$$

- Pour le dépôt final :

$$P(E_{nk}) = P(B_{nk}^{VA} \geq e_{nk}) \cap P(B_{nk}^{VA} \leq l_{nk}) \cap P(B_{nk}^{VA} \leq T_k + X_1)$$

$$P(E_{nk}) = P(e_{nk} \leq B_{nk}^{VA} \leq \min(l_{nk}; T_k + X_1))$$

$$P(E_{nk}) = \left(P(B_{nk}^{VA} \leq \min(l_{nk}; T_k + X_1)) - P(B_{nk}^{VA} \leq e_{nk})\right)$$

Comme expliqué dans la partie précédente, le calcul de la probabilité qu'une contrainte soit réalisable sous les différentes hypothèses H_1, H_2, H_3 est possible (session 4.5.1). Pour cela la fonction B_i^{VA} est étudiée pour les différentes valeurs que peut prendre A_i^{VA}.

Il est possible de connaître la fonction de densité associée à la variable aléatoire B_i^{VA} pour chaque sommet. Il est donc aussi possible de calculer la probabilité $P(E_i)$ en calculant l'intégrale de ces lois. Sous cette hypothèse d'indépendance, la probabilité $P(\lambda)$ de réalisabilité associée à λ est égale au produit des probabilités des E_i.

$$P(\lambda) = P(E_1)\ P(E_2)\ P(E_3)\ \dots\ P(E_{nk})$$

Sous cette hypothèse, $P(\lambda)$ est calculable analytiquement.

b) Sans hypothèse d'indépendance pour les variables E_i

Sans cette hypothèse d'indépendance, la probabilité de réalisabilité de la tournée obtenue par l'évaluation de λ est égale à la formule suivante :

$$P(\lambda) = P(E_1)\ P(E_2|E_1)\ P(E_3|E_1 \cap E_2)\ \dots\ P(E_{nk}|\ E_1 \cap\ E_2 \cap E_3\ \dots\ E_{nk-1})$$

Avec $P(A|B)$ la probabilité conditionnelle qui représente la probabilité de l'évènement A sachant que l'évènement B a déjà eu lieu. Sous cette condition, il n'est plus possible d'utiliser la loi suivie par Γ_i et la somme des temps d'attente X_{i-1} pour définir la loi suivie par A_i^{VA} car les contraintes sur les sommets précédents ainsi que les actions du chauffeur ont un impact sur A_i^{VA}.

Pour mettre en évidence la différence par rapport au cas précédent, un exemple est donné sur la figure 4.39. Dans la figure, les temps de chargement sont omis (donc $B_i = D_i$) et on note E_i^{VA} la variable aléatoire telle que les valeurs de B_i^{VA} respectent toutes les contraintes sur le sommet i. A gauche, les sommets sont indépendants et la loi suivie par A_i^{VA} est connue pour tous les sommets. Ceci permet de connaître la loi suivie par B_i^{VA} et donc E_i^{VA} qui correspond à B_i si la valeur respecte les contraintes. Dans la figure de droite, les temps de trajet entre les sommets ne sont plus indépendants et pour connaître la loi de A_i^{VA} il faut connaître la loi de $T_{i-1,i}$ (le temps de trajet entre $i-1$ et i est connu) et la loi suivie par E_{i-1}^{VA}(sur le sommet précèdent). Dans la figure de droite la valeur de départ est fixée sur le sommet $i-2$ et la loi suivie par A_{i-1}^{VA} est connue. Elle suit la même loi que $T_{i-1,i}$ centrée en $A_{i-1} = B_{i-2} + t_{i-2,i-1}$. Comme précédemment la loi de E_{i-1} peut-être calculée en fonction du comportement suivi par le chauffeur. Sur le sommet suivant, la loi suivie par E_i n'est plus une loi normale et la loi suivie par A_i^{VA} n'est plus la somme de deux loi normales : il faut donc la calculer pour pouvoir obtenir la loi suivie par E_i^{VA}.

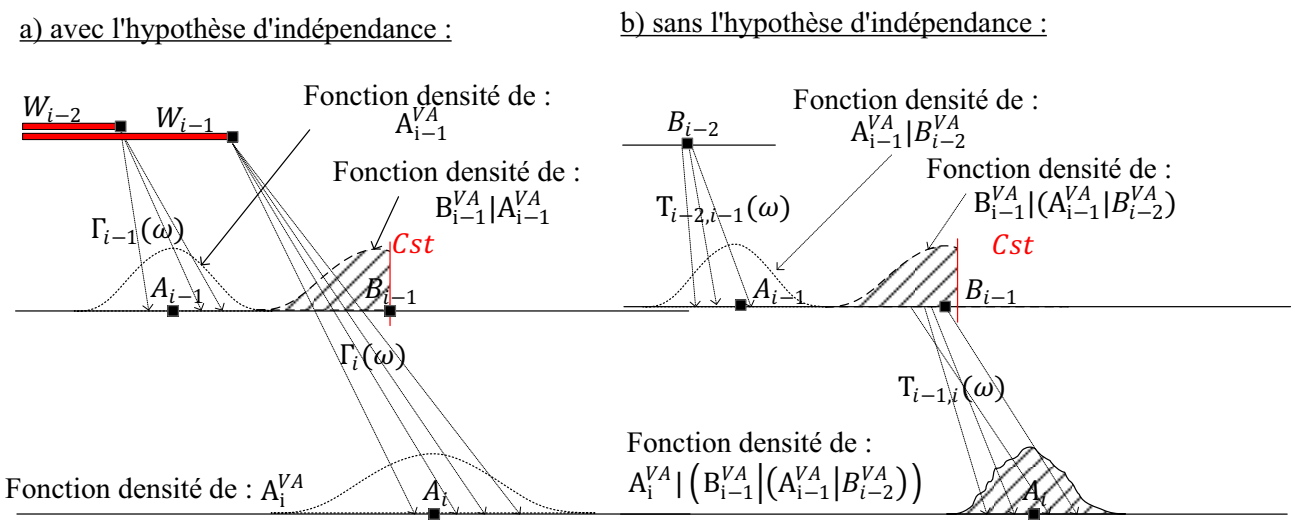


Figure 4.36 : Représentation de la loi de A_i^{VA} et B_i^{VA} sous H3

Dans le cas où A_i suit une loi normale, il est possible d'obtenir la valeur de la probabilité associée en utilisant la loi normale centrée réduite selon le principe présenté dans la section précédente. Dans le cas où la fonction de densité ne suit pas une loi normale, le calcul est possible, mais il est beaucoup plus coûteux. Il consiste en deux étapes : calculer la fonction de densité associée à A_i^{VA} puis, en fonction de l'hypothèse sur la politique suivie par le chauffeur, calculer la fonction de densité $f_i^E(x)$ associée à E_i^{VA}. Enfin, calculer l'intégrale entre les valeurs des contraintes pour obtenir la probabilité de E_i.

Trois étapes sont nécessaires :

1 – calculer la densité de probabilité associée à la loi suivie par E_{i-1}.

Pour cela, la date de démarrage de la tournée est connue et la loi suivie par $T_{i,i+1}$est connue pour tous les arcs de la tournée. Par récurrence, en utilisant la formule suivante, il est possible de calculer la densité de la variable $A_i|E_{i-1}$.

$$f_{i+1}^A(x) = \frac{\int_{-\infty}^{+\infty} e^{\frac{(x-m-t)^2}{2s^2}} f_i^E(t)dt}{\int_a^b \left(\int_{-\infty}^{+\infty} e^{\frac{(u-m-t)^2}{2s^2}} f_i^E(t)dt \right) du}$$

2 – comme le comportement du chauffeur est aussi connu pour les différentes valeurs de A_i^{VA}, il est possible d'en déduire la fonction de densité $f_{i+1}^B(x)$ associée à B_i^{VA}.

Si le comportement suivi par le chauffeur est connu, il est possible de construire une représentation de la fonction de densité suivie par la variable aléatoire B_i^{VA} en fonction des différentes valeurs de A_i^{VA}. Voici les illustrations dans le cas où la variable A_i^{VA} suit une loi normale de caractéristique connue. La figure 4.37 illustre la fonction de densité associée à la variable aléatoire B_i^{VA} pour l'hypothèse 1. La probabilité que B_i^{VA} soit égale à B_i correspond à la probabilité que A_i^{VA} soit inférieure à B_i, ce qui est représenté sur la figure par l'aire de la fonction de densité de A_i^{VA}. Ensuite la fonction de densité associée à B_i^{VA} est la même que celle de A_i^{VA}.

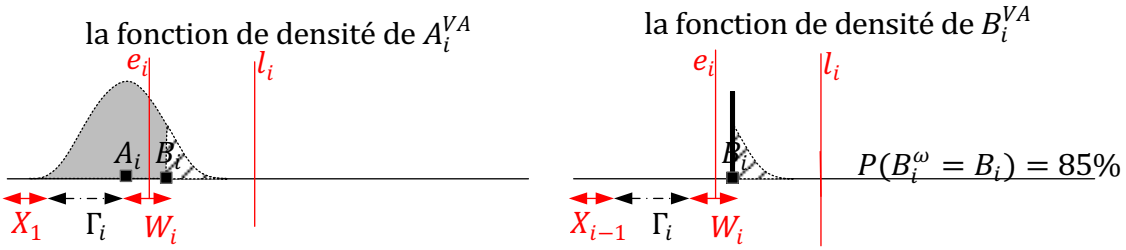


Figure 4.37 : Représentation de la loi de A_i^{VA} et B_i^{VA} sous H_1

La figure 4.38 illustre la fonction de densité associée à la variable aléatoire B_i^{VA} pour l'hypothèse 2. Sous cette hypothèse, la fonction de densité associée à B_i^{VA} est la même que celle de A_i^{VA} après une translation de valeur W_i.

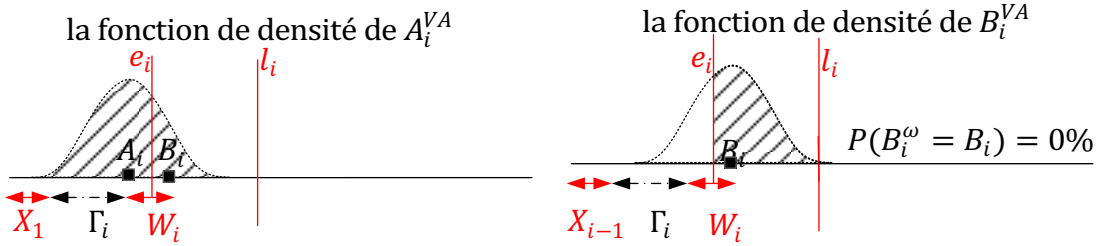


Figure 4.38 : Représentation de la loi de A_i^{VA} et B_i^{VA} sous H_2

La figure 4.39 illustre la fonction de densité associée à la variable aléatoire B_i^{VA} pour l'hypothèse 3. Sous cette hypothèse, la probabilité que B_i^{VA} soit égale à B_i n'est pas nulle et elle correspond à la probabilité que A_i^{VA} soit comprise entre A_i et B_i, ce qui est représenté par l'aire grisée sur la figure. Ensuite, la fonction de densité de la variable aléatoire B_i^{VA} se comporte comme celle de A_i^{VA} pour les valeurs supérieures à B_i et elle se comporte comme celle de A_i^{VA} après une translation de valeur W_i pour les valeurs inférieures à B_i.

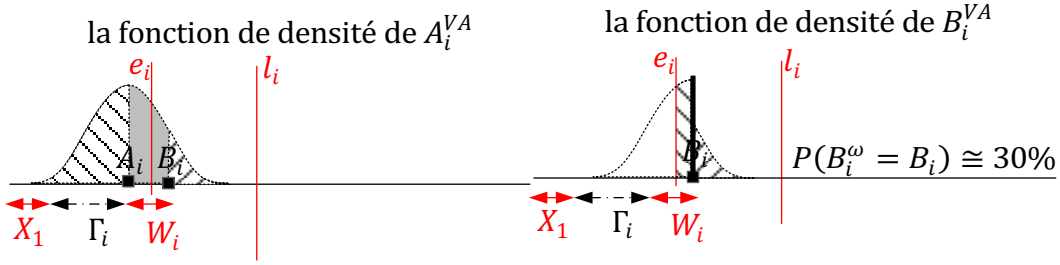


Figure 4.39 : Représentation de la loi de A_i^{VA} et B_i^{VA} sous H_3

3 – Une fois la densité de la loi suivie par B_i^{VA} connue, il est possible de calculer l'intégrale de cette fonction entre les différentes valeurs associées aux contraintes, illustrée sur la figure 4.40. Plusieurs méthodes de calcul d'intégrale peuvent être utilisées comme par exemple celle de Simpson améliorée par Romberg. Ce calcul permet d'obtenir $P(E_i)$.

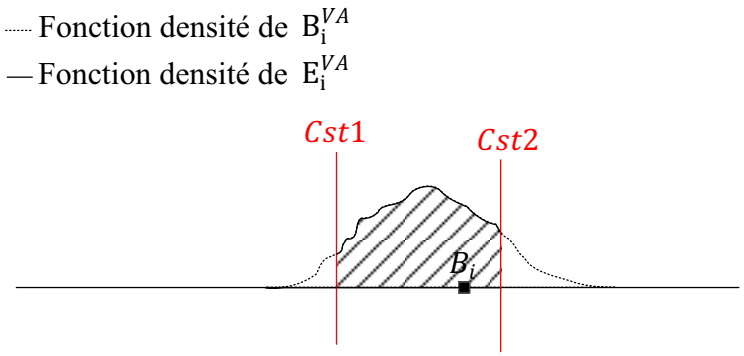


Figure 4.40 : Représentation de la loi de A_i^{VA} et B_i^{VA} sous H3

Malheureusement, pour obtenir une précision suffisante, cette méthode demande un temps de calcul trop important pour être utilisé dans les méthodes de résolution. Elle permet néanmoins d'évaluer la différence entre l'estimation obtenue par le calcul analytique et la valeur obtenue par simulation.

4.5.3. Déterminer la probabilité de réalisabilité par simulation

Dans la simulation, toutes les hypothèses peuvent être envisagées, mais la principale difficulté réside dans le nombre de réplications à effectuer pour obtenir un résultat significatif en termes de précision. Une nouvelle notation est introduite :

- $\overline{\mathrm{F_{Hk}}(t,\mathrm{n})}$: valeur moyenne de réalisabilité d'une tournée, obtenue après n réplications

avec Hk $\in \{H_1; H_2; H_3\}$. Comme les tournées sont considérées indépendantes, la valeur moyenne de réalisabilité d'une solution après n réplications peut être calculée et est notée :

$$\overline{\mathrm{F_{Hk}}(\mathrm{S},\mathrm{n})} = \prod_{i=1}^{m} \overline{\mathrm{F_{Hk}}(t_i,\mathrm{n})}$$

, avec t_i l'ensemble des tournées qui composent S.

Le schéma algorithmique est donné dans l'algorithme 3. Il traite le cas d'une simulation sous l'hypothèse 3 avec des sommets qui ne sont pas considérés indépendants. Les deux parties encadrées correspondant aux hypothèses considérées : le premier bloc correspond à l'indépendance entre les sommets et le second correspond à la politique du chauffeur. Une simulation peut donc être construite pour toutes les hypothèses envisagées. La date de départ de dépôt est initialisée à B_1 en ligne 1. L'algorithme réalise en ligne 3 pour chaque sommet un tirage d'une loi normale. Ceci qui correspond au temps de trajet entre le sommet $i-1$ et le sommet i. Il calcule alors la date d'arrivée A_i^{ω}. Puis, suivant le comportement défini pour le chauffeur, l'algorithme détermine la date B_i^{ω}. De la ligne 13 à la ligne 25, les différentes contraintes du problème sont testées avec la valeur B_i^{ω}. Si une contrainte est violée, le programme s'arrête. Sinon il passe au sommet suivant dans la tournée.

Même si un nombre important d'exécutions est nécessaire, la complexité de l'algorithme est en $O(n)$, où n est le nombre de sommets dans la tournée. Ceci permet de conserver un temps de calcul raisonnable.

Simulation sous l'hypothèse H3 sans l'hypothèse des sommets indépendants

Algorithme 3 $simulation$

Input parameters:

λ : **a node list**

Global parameters:

t_{ij} : starting time on the node i

e_i, l_i : time windows on node S_i

p[i] : position of the pickup node associated with the delivery node i in λ

L : maximal riding time for clients

T_k : maximal riding time for the vehicles

Variables:

A_i^ω : simulated arriving time on the node i

B_i^ω : simulated starting time on the node i

Output parameters:

faisability : true → $trip\ is\ faisable$ | false → $trip\ is\ unfaisable$

Begin

$B_1^\omega := B_1$

For i in {2; ...; nk} **do**

$T_{ij}(\omega) :=$ loi_normale(t_{i-1i} , $t_{i-1i}/10$) // Bloc for dependent nodes

$A_i^\omega := B_{i-1}^\omega + d_{s_i} + T_{ij}(\omega)$

if $(A_i^\omega < A_i)$ **then** // Bloc for the hypothesis H3

  $B_i^\omega := A^\omega{}_i + W_i$

else if $(A_i^\omega \leq B_i)$ **then**

  $B_i^\omega := B_i$

else if $(A_i^\omega > B_i)$ **then**

  $B_i^\omega := A_i^\omega$

endif

endif

if $(B_i^\omega \in [e_i;\ l_i] = false)$ **then**

  faisability = false

  return faisability

endif

if ($(is\_delivery\_node(i))\ AND\ (B_i^\omega - B_{p(i)}^\omega \geq L)$) **then**

  faisability = false

  return faisability

endif

if (($i = nk$) AND $(B_i^\omega - B_1^\omega \geq T_k)$) **then**

  faisability = false

  return faisability

endif

end do

faisability := $true$;

return faisability

End

4.6. Génération de solutions robustes

4.6.1. Etude de la robustesse des meilleurs solutions du DDARP

Les instances utilisées sont les instances classiques de la littérature sur le transport à la demande : elles ont été présentées dans l'article de (Cordeau and Laporte, 2003). Les solutions optimales de ces instances ne sont pas connues. Par contre, de nombreuses solutions ont été publiées au cours de ces dernières années. Parmi ces publications, on peut citer les travaux de (Cordeau and Laporte, 2003), (Parragh and Schmid, 2013) et (Braekers et al., 2014).

Le tableau 4.5 regroupe les résultats pour le calcul de la robustesse des meilleures solutions connues (BKS – *Best Known Solution*) dans la littérature. Pour chaque instance, le nom (Nom), le nombre de véhicule (m), le nombre de clients (n) et la valeur de la BKS ($c(BKS)$) sont indiqués. Pour chaque hypothèse $\mathrm{H}_k \in \{H_1; H_2; H_3\}$, le critère de robustesse $\rho^*_{H_k}$ de la BKS et l'estimation par simulation de la réalisabilité de la BKS avec n réplications $\overline{\mathrm{F}_{H_k}(\mathrm{BKS}, \mathrm{n})}$ sont présentés. Ici, $n = 100000$ réplications sont utilisées. On remarque que, même si les valeurs du critère $\rho^*_{H_k}$ et l'estimation de la probabilité associée $\overline{\mathrm{F}_{H_k}(\mathrm{BKS}, \mathrm{n})}$ sont différentes, les variations sur la figure 4.41 des courbes quelles que soit l'hypothèse sont similaires. Par exemple pour H_1, les instances pr01, pr02, pr11 et pr18 ont un critère $\rho^*_{H_1}$ élevé. Ces mêmes instances ont une probabilité $\overline{\mathrm{F}_{\mathrm{H}_1}(\mathrm{BKS}, \mathrm{n})} \geq 70\%$. Les instances pr03, pr06 et pr10 ont des valeurs de $\rho^*_{H_1}$ proches de 0. Elles ont aussi une valeur $\overline{\mathrm{F}_{\mathrm{H}_1}(\mathrm{BKS}, \mathrm{n})}$ proche de 3%. Ces mêmes remarques tiennent pour les deux autres hypothèses. Ainsi, le critère ρ^* semble être un critère représentatif de la robustesse de la solution.

Tableau 4.5:
Résultats de l'optimisation sur la meilleure solution connue du DDARP

Nom	m	n	c(BKS)	$\rho^*_{H_1}$	$\overline{\mathrm{F}_{H_1}(\mathrm{BKS}, \mathrm{n})}$	$\rho^*_{H_2} \equiv \rho^*_{H_3}$	$\overline{\mathrm{F}_{H_2}(\mathrm{BKS}, \mathrm{n})}$	$\overline{F_{\mathrm{H}_3}(\mathrm{BKS}, \mathrm{n})}$
pr01	3	24	190.02	74.8%	86.4%	66.8%	25.90%	50.00%
pr02	5	48	301.34	76.8%	85.4%	48.6%	28.40%	72.60%
pr03	7	72	532.00	4.8%	0.1%	3.1%	0.00%	0.00%
pr04	9	96	570.25	14.3%	0.3%	9.4%	0.00%	0.10%
pr05	11	120	626.93	13.1%	0.7%	5.0%	0.00%	0.70%
pr06	13	144	785.26	3.9%	0.0%	3.0%	0.00%	0.00%
pr07	4	36	291.71	21.9%	14.7%	14.7%	0.30%	7.90%
pr08	6	72	487.84	8.9%	0.2%	5.9%	0.00%	0.20%
pr09	8	108	658.31	9.4%	0.4%	4.3%	0.00%	0.30%
pr10	10	144	851.82	1.3%	0.0%	0.6%	0.00%	0.00%
pr11	3	24	164.46	63.2%	61.0%	57.6%	36.10%	69.00%
pr12	5	48	295.66	32.0%	5.8%	25.0%	0.20%	3.70%
pr13	7	72	484.83	32.5%	16.5%	19.3%	0.80%	6.20%
pr14	9	96	529.33	59.3%	64.6%	35.6%	9.20%	27.80%
pr15	11	120	577.29	37.9%	72.0%	17.9%	1.90%	25.10%
pr16	13	144	730.67	16.2%	11.1%	7.5%	0.20%	3.40%
pr17	4	36	248.21	26.8%	1.6%	23.8%	0.30%	0.90%
pr18	6	72	458.73	62.2%	72.9%	43.8%	17.00%	54.40%
pr19	8	108	593.49	5.8%	0.0%	3.5%	0.00%	0.10%
pr20	10	144	785.68	3.8%	0.0%	2.8%	0.00%	0.10%
avg.					24.7%		6.02%	16.13%

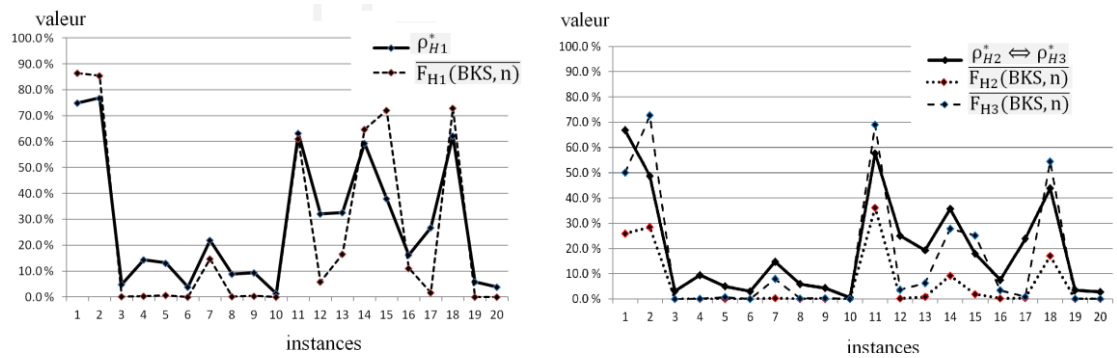


Figure 4.41 : Corrélation entre ρ et la robustesse obtenue par simulation*

4.6.2. La méthode de résolution avec hiérarchisation des critères

L'ELS proposée dans le chapitre 2 pour résoudre le DARP a été modifiée pour obtenir des solutions robustes. La fonction coût a été définie selon la proposition de (Cordeau and Laporte, 2003), c'est-à-dire $f(s) = c(s) + \alpha(1 - \rho^*)$. Ce coût prend en compte deux critères : le coût $c(s)$ de la solution, identique ou coût de la solution dans le cas déterministe (distance parcourue par la flotte) et le critère de robustesse ρ^* défini précédemment.

La robustesse est le critère principal de la fonction coût. Elle est pondérée dans $f(x)$ par un coefficient α défini de sorte que $\min(\alpha\ (1 - \rho^*)) > \max(c(s))$. La méthode de résolution est appliquée séparément pour les hypothèses H_1 puis pour les hypothèses H_2 et H_3 car les critères $\rho^*_{H_k}$ sont calculés différemment dans ces deux cas. La méthode de résolution commence par une phase d'optimisation qui retourne une solution avec un indice $\rho^*_{H_k}$ élevé et un coût faible. Dans un second temps, les tournées associées à la solution obtenue sont testées à l'aide de simulations afin d'obtenir des informations supplémentaires, notamment la probabilité de réalisabilité de la tournée. La valeur $\overline{(F(s,n))}$ associée à la solution peut alors être calculée.

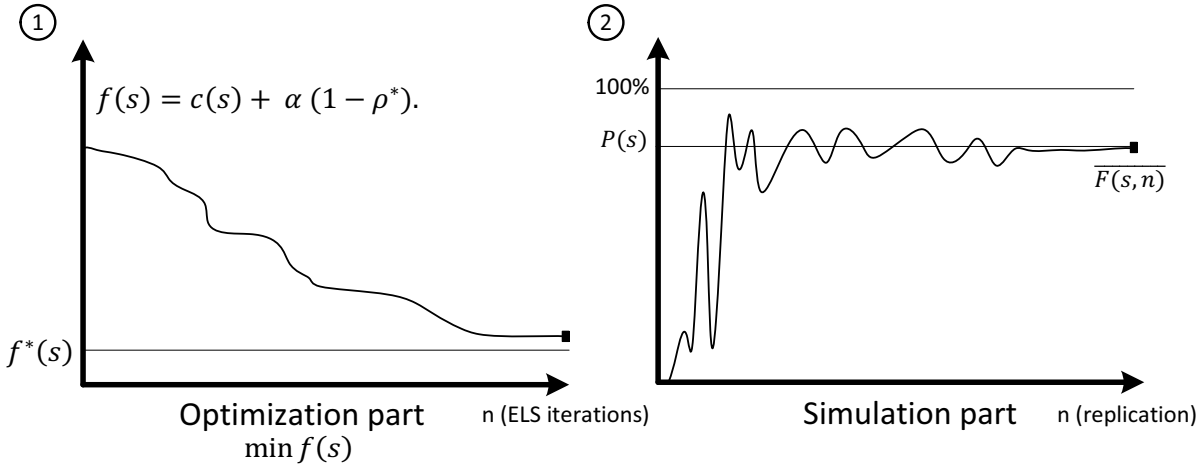


Figure 4.42 : Schéma de résolution

4.6.3. Les résultats obtenus avec l'approche ELS

Le nombre de réplications utilisées pour évaluer la robustesse d'une solution par simulation est $n = 10000$. Les simulations sont réalisées dans le cas "sans indépendance entre les sommets". La résolution a été effectuée pour les trois hypothèses sur la politique de gestion des chauffeurs en cas de retard ou d'avance. Les solutions proposées sous les hypothèses H_2 et H_3 sont les mêmes puisque la méthode de résolution utilise la même fonction d'évaluation et retourne les mêmes valeurs du critère de robustesse et du coût.

Les solutions obtenues pour chaque instance avec les différentes hypothèses sont représentées dans le tableau 4.6. Pour chaque instance et sous chaque hypothèse (H_1 ou H_2 / H_3), le coût c(s) de la meilleure solution trouvée s et l'écart relatif (gap) par rapport au coût de la BKS sont indiqués. Puis la valeur du critère de robustesse $\rho^*_{H_k}$ et l'estimation par simulation de la réalisabilité $\overline{F_{H_k}(BKS, n)}$ de s avec n réplications sont présentées. Le temps en minutes pour obtenir s est également fourni.

Toutes les solutions obtenues sont robustes sous hypothèse H_1. La réalisabilité pour chacune est supérieure à 99.9% dans 19 des 20 instances testées. Concernant les hypothèses H_2 et H_3, les résultats sont également supérieurs à la robustesse obtenue pour les BKS, avec un critère

de robustesse moyen de 97.8% et un estimateur de la probabilité d'être réalisable à 96,4% pour H_2 et 98,7% pour H_3. Sous les hypothèses H_2 et H_3, des solutions de moins bonne qualité sont obtenues pour l'instance pr10 avec un $\rho^* = 78{,}7\%$. Elle est associée à un estimateur de probabilité d'être réalisable de seulement 63,3 % pour H_2 et 96,2% pour H_3. Ceci correspond néanmoins à une grande amélioration par rapport au 0% associé à la BKS bien que cela reste quand-même assez faible pour H_2 (plus d'une chance sur 3 que la de la solution ne soit pas réalisable en pratique).

L'augmentation de la robustesse de la solution se fait logiquement au détriment du coût. Les solutions générées ont une augmentation du coût de 18% en moyenne par rapport à la meilleure solution connue du DDARP pour les hypothèses H_2 et H_3 et d'environ 13% pour l'hypothèse H_1. Les temps de résolution sont les même que ceux utilisés lors de la résolution du DDARP.

Il est intéressant de noter que les hypothèses H_2 et H_3 où le véhicule est autorisé à prendre de l'avance sur les sommets ont au final des probabilités plus faibles d'être valides que les solutions obtenues avec l'hypothèse H_1. Cette différence est probablement due aux contraintes de temps de trajet des clients. Si le client est embarqué plus tôt, alors il doit aussi être déposé plus tôt.

Tableau 4.6:
Résultats de l'optimisation sur la meilleure solution obtenue avec l'approche ELS

	stochastique H_1					stochastique H_2 et H_3					
nom	c(S)	gap	$\rho^*_{H_1}$	$\overline{F_{H_1}(s,n)}$	temps	c(S)	gap	$\rho^*_{H_2}/\rho^*_{H_3}$	$\overline{F_{H_2}(s,n)}$	$\overline{F_{H_3}(s,n)}$	temps
pr01	200.48	5.50%	100.0%	100.0%	0.25	202.98	6.82%	100.0%	100.0%	100.0%	0.25
pr02	307.11	1.92%	100.0%	100.0%	1.25	314.59	4.4%	100.0%	100.0%	100.0%	1.25
pr03	587.96	10.52%	100.0%	100.0%	2.30	643.53	20.96%	97.9%	96.1%	99.9%	2.30
pr04	643.75	12.89%	100.0%	100.0%	7.37	665.65	16.73%	99.5%	99.2%	100.0%	7.37
pr05	744.92	18.82%	100.0%	100.0%	12.07	767.45	22.41%	98.3%	97.3%	99.6%	12.07
pr06	979.78	24.77%	100.0%	100.0%	21.92	1109.04	41.23%	94.5%	91.2%	98.8%	21.92
pr07	306.69	5.13%	100.0%	100.0%	0.47	328.96	12.77%	100.0%	100.0%	100.0%	0.47
pr08	573.65	17.59%	100.0%	100.0%	2.68	592.72	21.5%	95.9%	93.4%	99.7%	2.68
pr09	774.53	17.65%	99.7%	100.0%	11.25	801.65	21.77%	95.4%	94.8%	99.4%	11.25
pr10	1049.23	23.18%	99.4%	99.5%	21.33	1085.67	27.45%	78.7%	63.3%	96.2%	21.33
pr11	168.81	2.64%	100.0%	100.0%	0.28	171.43	4.24%	100.0%	100.0%	100.0%	0.28
pr12	310.66	5.07%	100.0%	100.0%	1.37	324.75	9.84%	100.0%	100.0%	100.0%	1.37
pr13	527.10	8.72%	100.0%	100.0%	3.70	544.54	12.31%	100.0%	100.0%	100.0%	3.70
pr14	634.16	19.80%	100.0%	100.0%	10.20	630.64	19.14%	99.9%	99.9%	100.0%	10.20
pr15	634.08	9.84%	100.0%	100.0%	19.93	702.38	21.67%	100.0%	100.0%	100.0%	19.93
pr16	891.86	22.06%	100.0%	100.0%	32.32	882.28	20.75%	99.4%	99.4%	100.0%	32.32
pr17	266.83	7.50%	100.0%	100.0%	0.58	271.55	9.4%	100.0%	100.0%	100.0%	0.58
pr18	494.54	7.81%	100.0%	100.0%	4.32	560.83	22.26%	100.0%	100.0%	100.0%	4.32
pr19	699.70	17.89%	100.0%	100.0%	12.43	738.65	24.46%	98.5%	97.4%	99.6%	12.43
pr20	978.61	24.56%	100.0%	100.0%	31.45	989	25.88%	97.1%	95.5%	99.8%	31.45
avg.		**13.19**	**100.0%**	**100.0%**	**9.87**		**18.30**	**97.8%**	**96.4%**	**99.7%**	**9.87**

4.7. L'optimisation bi-objectif

La dernière partie de ce chapitre consiste à mettre en place une méthode bi-objectif de résolution du SDARP. Les deux critères étudiés sont les mêmes que précédemment, *i.e.* le coût et la robustesse. Ils correspondent respectivement à la distance parcourue par la flotte et à la probabilité $F_{Hx}(s)$ que la tournée soit réalisable pour l'hypothèse Hx avec $x\epsilon\{1;2;3\}$. Dans la partie précédente, il a été montré que quelle que soit l'hypothèse considérée, cette

probabilité est difficilement calculable analytiquement et qu'obtenir une estimation par simulation est aussi coûteux en temps de calcul. C'est pourquoi l'optimisation est réalisée en utilisant le critère ρ^*, qui est nommé ρ pour simplifier les notations dans la partie 4. Comme expliqué dans la présentation du problème, le critère ρ dépend de l'hypothèse considérée. Le critère ρ correspond à ρ^*_{H1} pour l'hypothèse H_1 dans lequel les véhicules ne sont pas autorisés à prendre de l'avance. Et ρ vaut $\rho^*_{H2/H3}$, associé à la fois aux hypothèses H_2 et H_3 dans lesquelles le véhicule peut prendre de l'avance.

La méthode d'optimisation utilisée est un algorithme à population. Le résultat d'une exécution de la méthode est un front de Pareto sur les deux critères (le coût et ρ). L'algorithme utilisé est une version modifiée de NSGA-2 dans laquelle la phase de reproduction est remplacée par une phase de mutation.

A la fin de la méthode, une fois le front de Pareto obtenu, la robustesse des solutions du front est vérifiée par une étape de simulation. Comme expliqué dans le paragraphe précédent, la simulation permet d'obtenir un estimateur $\overline{F_{Hx}(s,n)}$ de la robustesse, associé à l'hypothèse Hx avec $x\epsilon\{1;2;3\}$ et avec n simulations.

4.7.1. Littérature de l'optimisation multi-objectif

Concernant les problèmes de tournées de véhicules, le DARP est un problème dans lequel la qualité de service est un critère important puisque les éléments transportés sont non seulement des personnes mais en plus le plus souvent les émetteurs des requêtes de transport. C'est pourquoi il existe quelques articles de la littérature qui proposent des optimisations multi-objectifs afin de prendre en compte à la fois le coût et la qualité de la solution.

Par exemple, en 2012, (Zidi et al., 2012) propose un algorithme de recuit simulé multi-objectif (MOSA : Multi-Objective Simulated Annealing) sur le problème du DARP. Les critères étudiés sont le coût et la qualité de service, composée de deux critères. Ces deux critères sont : le temps de trajet des clients et le nombre de sommets visités par le véhicule pendant le trajet d'un client. Ces critères sont traités sans agrégation en utilisant une règle de dominance.

Plus récemment, (Guerriero et al., 2014) proposent une méthode bi-objectif pour le DARP qui a pour objectif d'optimiser simultanément le temps de trajet maximal et le temps d'attente total de la solution. Leur approche se compose de deux étapes. La première détermine un ensemble de tournées réalisables avec un algorithme génétique. La deuxième est une approche bi-objectif qui résout un problème de partitionnement avec une méthode ε-contrainte.

4.7.2. Choix des critères d'optimisation

Les deux critères sont optimisés dans cette partie sont le coût et ρ. Le premier critère est un critère qu'il faut minimiser et le second un critère (compris entre 0 et 1) qu'il faut maximiser. Le front de Pareto correspondant est donc orienté différemment de celui utilisé le chapitre 3 pour le problème du TDVRP. La représentation graphique du front est une courbe qui va d'en bas à gauche vers le haut à droite, comme illustré sur la figure 4.43. Plus un front se rapproche du sommet qui correspond à la solution de coût 0 et de ρ 1 et plus il est considéré comme de bonne qualité.

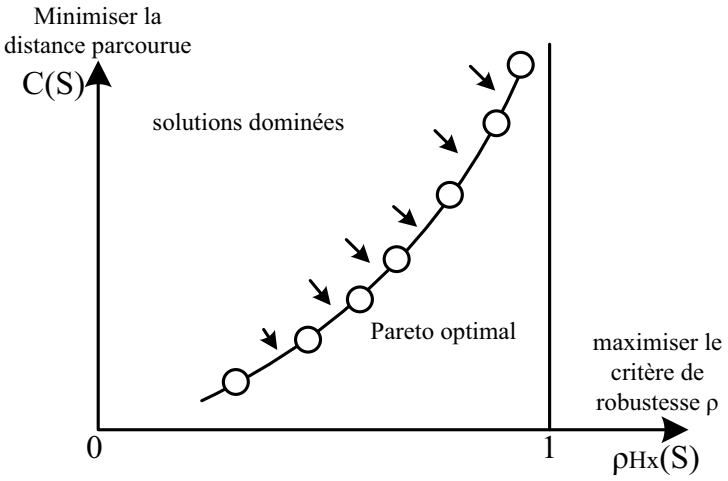


Figure 4.43 : Stratégie de résolution multicritère

Comme expliqué dans le chapitre 3 sur le TD VRP, il est possible de comparer les fronts de Pareto en utilisant un indice nommé hypervolume (HV). Cette mesure consiste à calculer l'aire qui correspond aux solutions non-dominées par le front, comme représenté sur la figure 4.44. Comme $\rho(S) \in [0,1]$ cette aire est bornée par 1 pour ρ(S). Pour le coût, il faut déterminer une borne supérieure et l'utiliser systématiquement pour comparer deux fronts. Pour les expérimentations réalisées dans ce chapitre, la valeur utilisée est 2000. Plus la valeur de HV est faible et plus le front est de bonne qualité. L'objectif de la méthode d'optimisation est donc d'obtenir un ensemble de solutions non-dominées dont HV est le plus proche de l'indice HV associé au front de Pareto optimal.

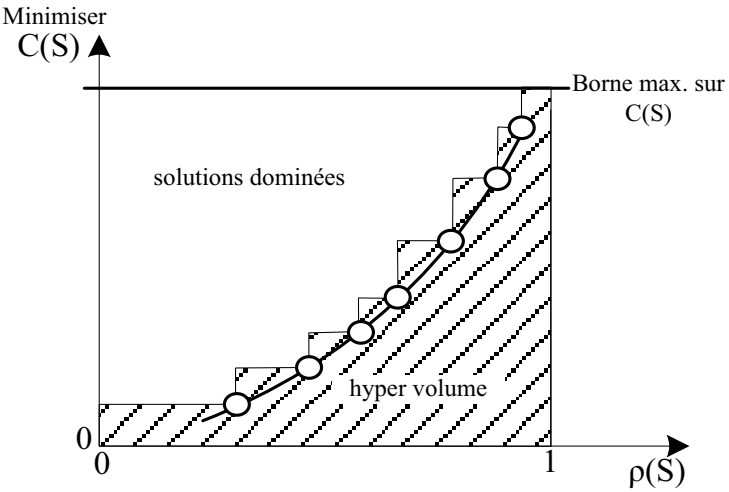


Figure 4.44 : Méthode de résolution multicritère

Le problème réside dans le fait que le front Pareto optimal n'est pas connu sur les instances utilisées. Il est donc intéressant de trouver une méthode qui permet d'estimer la qualité du front obtenu. Une idée est proposée dans cette partie. Un front de Pareto est construit en utilisant la solution obtenue lors de la résolution du DDARP et celle obtenue pour le SDARP avec la méthode monocritère. De ce front de Pareto, composé uniquement de deux solutions, trois valeurs sont calculées B1, B2, B3. Sur la figure 4.45, la meilleure solution connue pour le DDARP est notée BKS et la meilleure solution trouvée dans la partie 4.6.3 est notée (BFS). Les valeurs B1, B2, B3 sont alors utilisées pour déterminer la qualité d'un front obtenu à la fin de la méthode d'optimisation.

La première valeur $B1$ correspond à l'hypervolume (HV) associé au front composé uniquement des deux solutions BKS et BFS. Il est représenté par l'aire hachurée sur la figure 4.45. Pour le calculer la formule suivante est utilisée :

$$B1 = c(BKS)\rho(BFS) + \big(1 - \rho(BFS)\big)Bm + [c(BFS) - c(BKS)\rho(BKS) - \rho(BFS)]$$

, avec Bm borne maximale utilisée pour le calcul de HV.

Une seconde valeur $B2$, plus faible, est associée à des fronts de qualité supérieure. Elle correspond à $B1$ de laquelle est retirée l'aire représentée en bleu dans la figure 4.45.

$$B2 = c(BKS)\rho(BFS) + \big(1 - \rho(BFS)\big)Bm + [c(BFS) - c(BKS)\rho(BKS) - \rho(BFS)]/2$$

Enfin une dernière valeur $B3$ est associée aux fronts de très bonne qualité. Elle correspond à $B2$ moins l'aire en rouge dans figure 4.45.

$$B3 = c(BKS) * \rho(BFS) + (1 - \rho(BFS)) * Bm$$

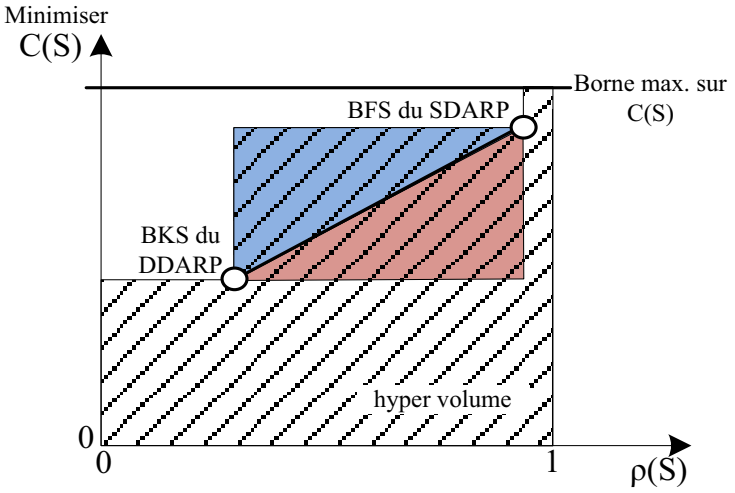


Figure 4.45 : Borne supérieure à l'hypervolume

Pour chaque indice HV associé à un front, correspond une qualité $\in \{0; 1; 2; 3\}$. La qualité 0 est associée aux fronts tels que $HV \geq B1$; la qualité 1 est associée aux fronts tels que $B1 > HV \geq B2$; la qualité 2 est associée aux fronts tels que $B2 > HV \geq B3$ et la qualité 3 est associée aux fronts tels que $B3 < HV$.

La qualité 0 est la qualité la plus faible et la qualité 3 est la plus grande. On peut souligner le fait que tous les fronts qui comportent plus de solutions, sans améliorer le front, sont de qualité 1. Un exemple est donné sur la figure 4.46 (a). Un front qui appartient à la catégorie 3 possède au moins une solution qui améliore une des deux solutions (BFS) ou (BKS) comme illustré figure 4.46 (b).

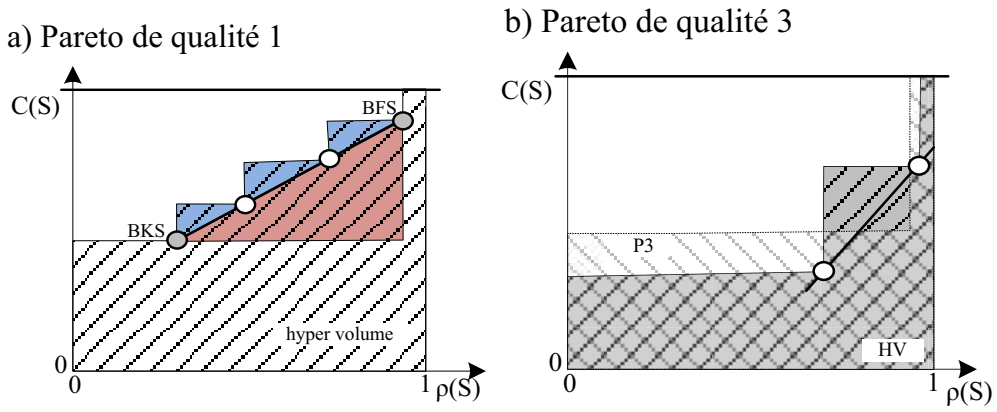


Figure 4.46 : Exemple de Pareto de différentes qualités

Les valeurs de ces différents critères ont été calculées pour les instances proposées par (Cordeau and Laporte, 2003) qui servent d'instances de test dans ce chapitre. Les résultats sont reportés dans le tableau 4.7. Pour des raisons de lisibilité, comme deux critères ρ sont utilisés, un pour H_1 et un pour H_2 et H_3, le tableau est séparé en deux.

Tableau 4.7:
Valeurs des différentes bornes utilisées pour déterminer la qualité des fronts de Pareto.

	stochastique H_1			stochastique H_2 et H_3		
	B1	*B2*	B3	B1	*B2*	B3
pr01	192.66	191.34	190.02	193.29	191.65	190.02
pr02	302.68	302.01	301.34	304.41	302.88	301.34
pr03	585.27	558.64	532.00	666.66	614.75	562.83
pr04	633.24	601.74	570.25	658.68	618.04	577.40
pr05	729.46	678.20	626.93	770.00	710.13	650.27
pr06	972.19	878.73	785.26	1145.42	998.74	852.07
pr07	303.41	297.56	291.71	320.80	306.26	291.71
pr08	566.01	526.93	487.84	641.08	595.46	549.84
pr09	767.28	714.81	662.34	843.30	781.66	720.03
pr10	1052.37	955.54	858.71	1277.38	1186.88	1096.38
pr11	166.06	165.26	164.46	167.02	165.74	164.46
pr12	305.86	300.76	295.66	315.44	305.55	295.66
pr13	513.36	499.10	484.83	525.13	504.98	484.83
pr14	572.00	550.66	529.33	571.93	551.37	530.80
pr15	612.56	594.92	577.29	654.97	616.13	577.29
pr16	865.75	798.21	730.67	864.43	801.36	738.29
pr17	261.84	255.02	248.21	265.29	256.75	248.21
pr18	472.27	465.50	458.73	497.32	478.03	458.73
pr19	693.54	643.51	593.49	749.15	681.87	614.59
pr20	971.28	878.48	785.68	1010.59	915.74	820.90

4.7.3. La méthode de résolution

a) Les algorithmes à population

Les algorithmes à population améliorent successivement un ensemble de solutions. Contrairement aux méthodes par agrégation, ils permettent d'obtenir un estimateur du front de Pareto optimal en une unique exécution.

Les points clés de ce type de méthodes sont :

- la sélection des parents selon la règle suivante : plus les individus sont de bonne qualité, plus ils ont de chance d'être sélectionnés ;

- les opérateurs de croisement ;
- le remplacement qui consiste à sélectionner une sous-population afin qu'elle devienne la nouvelle génération.

Le SDARP est un problème très contraint et aucun opérateur suffisamment efficace n'a pu être développé afin d'obtenir des solutions valides construites à partir de deux solutions parent. C'est pourquoi le choix a été fait de remplacer la phase de reproduction par une phase de mutation. L'algorithme utilisé est basé sur un algorithme de type NSGA-2 dans lequel la phase de reproduction est modifiée. Le pseudo-code de cet algorithme est présenté dans l'algorithme 4. Il consiste à générer une population initiale de solutions, avec ns le nombre de solutions désirées. C'est un paramètre d'entrée de l'algorithme. Une fois cette initialisation réalisée, la méthode répète trois étapes successivement jusqu'à ce que le critère d'arrêt soit atteint. Ces trois étapes sont :

- l'ajout de ns solutions à la population à l'aide de la procédure $add\_children()$, le nombre de solutions total à la fin de cette étape est donc de $2ns$;
- le tri par non-domination de la population avec la procédure $non\_dominated\_sort()$;
- la sélection les ns meilleures solutions avec la fonction $select\_best\_solutions()$. Ces solutions forment la nouvelle population.

Différents critère d'arrêt peuvent être utilisés, les plus simples sont le nombre d'itérations ou le temps d'exécution. Pour l'expérimentation, le temps d'exécution est utilisé.

Algorithme 4 $NSGA-2*$

Input parameters
ns : nb of solutions in the initial population

Output parameters
$POP[]$: a population of solution

Begin
1 **call** initial_$population(POP, ns)$
2 **repeat**
3 | **call** $add\_children(POP, ns)$
4 | **call** $non\_dominated\_sort(POP, ns)$
5 | $POP :=$ **call** $select\_best\_solutions(POP, ns)$
6 **until** (!stopping_criterion)
7 **return** POP
End

b) La population initiale

Une population initiale est générée en utilisant la fonction d'initialisation présentée dans le chapitre 2. Cette fonction génère des solutions pour le problème du DDARP en deux parties. Un ordre sur les sommets d'origine des clients est d'abord généré, puis à l'aide d'une fonction d'insertion basée uniquement sur le coût, une solution est construite. Ce procédé est répété jusqu'à ce que le nombre de solutions dans la population soit suffisant.

Un exemple de la population initiale est illustré sur la figure 4.47. Elle représente une population de 100 solutions avec en rouge les solutions non-dominées.

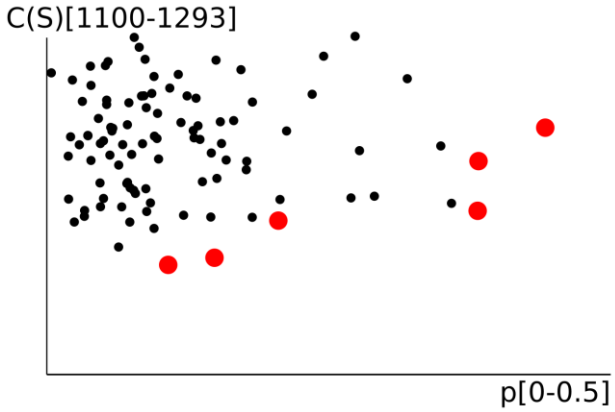


Figure 4.47 : Phase d'initialisation pour l'instance PR06 sous l'hypothèse H_1

c) Le tri par non domination

Le tri par non-domination permet d'obtenir un classement des solutions dans la population. Il partitionne un ensemble E de solutions en plusieurs ensembles E_i composés de solutions non-dominées. Le principe consiste à calculer les solutions non-dominées de l'ensemble E. Ces solutions sont placées dans E_1 avant d'être supprimées de E. Les nouvelles solutions non-dominées de E sont ensuite calculées puis placées dans E_2 et supprimées de E. Ce processus est répété jusqu'à ce que $E = \emptyset$. On dit d'une solution appartenant à E_i que c'est une solution de rang i.

Le résultat obtenu après un tri par non domination sur une population du SDARP est illustré par la figure 4.48.

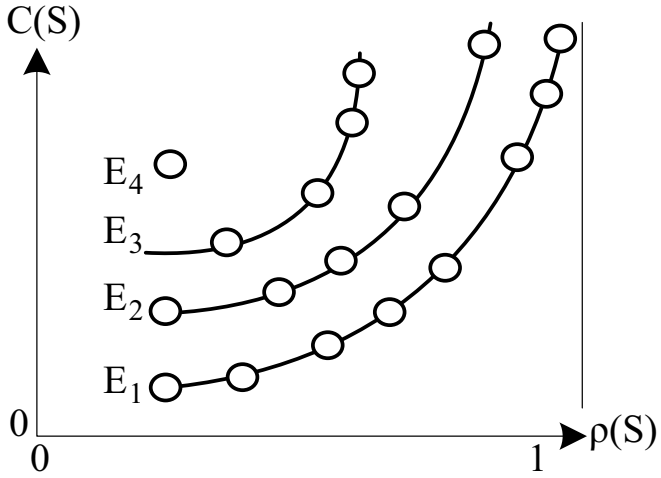


Figure 4.48 : Tri des solutions par non-domination

Chacun des ensembles E_i est ensuite trié par ordre de coût croissant. Ceci est équivalant à les trier par ordre de robustesse décroissante puisque les solutions qui composent E_i sont non-dominées. Le résultat pour les 5 premiers ensembles de la population initiale de la figure 4.47 est illustré dans la figure 4.49.

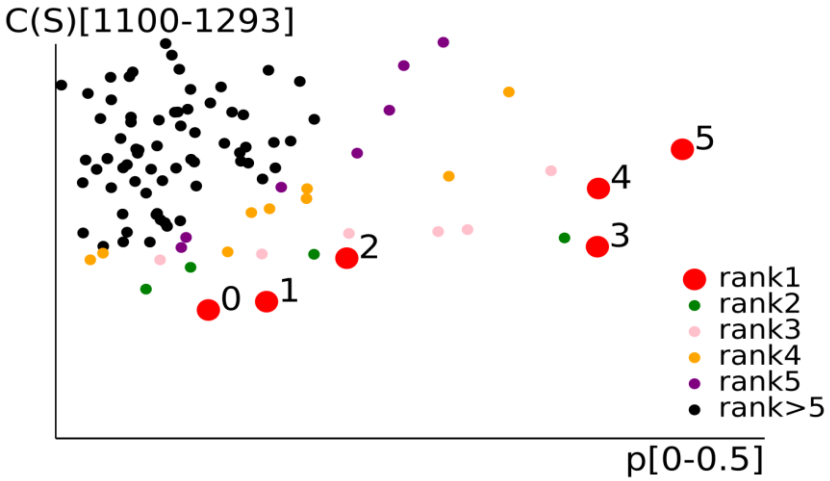


Figure 4.49 : Le tri par non-domination sur la population initiale de l'instance PR06 sous H1

d) Génération de nouvelles solutions

La génération de nouvelles solutions se compose de 3 étapes :

- une solution de la population initiale est sélectionnée, elle devient la solution courante $S_{père}$;
- une mutation est appliquée à $S_{père}$;
- une recherche locale est appliquée à la solution courante. Les solutions rencontrées durant cette phases sont considérées comme des solutions filles, notées S_{fils}. Une partie de ces solutions est ajoutée à la population en fonction de la valeur des critères et de la position de la solution $S_{père}$ dans la population.

L'algorithme 5 définit la procédure $\_children()$. Elle prend en entrée une population POP ainsi qu'un nombre nc de solutions à générer.

La fonction $select()$ à la ligne 3 sélectionne la solution $S_{père}$ dans la population. Cette sélection est aléatoire mais pas uniforme entre toutes les solutions. Les solutions de rang plus élevé ont une probabilité plus grande d'être choisies.

Une mutation est alors appliquée la ligne 4 à cette solution $S_{père}$. Entre les deux mutations présentes dans la résolution du DDARP au chapitre 2, le choix a été fait de n'utiliser que le premier mouvement. Il consiste à choisir aléatoirement un client parmi tous les clients, à le retirer de la tournée à laquelle il appartient, avant de le réinsérer dans une autre tournée, sans tenir compte ni du coût ni de la robustesse.

De la ligne 6 à la ligne 15, une phase de recherche locale est appliquée sur la solution S_{fils}. Les paramètres mi et $miwi$ correspondent respectivement au nombre d'itérations maximales et au nombre d'itérations maximales sans ajouter de fils. Ils sont définis comme des paramètres globaux.

Les mêmes mouvements que ceux utilisés pour le DDARP sont utilisés pour le SDARP, mais au lieu de s'intéresser uniquement aux mouvements améliorant le coût, toute solution qui n'est pas dominée par la solution courante est considérée comme améliorante et donc génère une nouvelle solution fille, qui devient la nouvelle solution courante à l'étape suivante.

Algorithme 5 $add\_children$

Input/output parameters

POP : population of solution
nc : nb of children needed

Variable parameters

POP_C : population of children solutions
$S_{père}, S_{Fils}$: solutions

Global parameters

mi : max. iterations
$miwi$: max. iterations without improvement

Begin

1 $POP\_C \coloneqq \emptyset$
2 **repeat**
3 | S_{Pere}:= **call** $select$ (POP)
4 | S_{Fils}:= **call** $mutations$ (S_{Pere})
5 | $i \coloneqq 0$ and $j \coloneqq 0$
6 | **while** ((!stopping_criterion) **and** (i < mi) **and** (j < miwi)) **then**
7 | | $i \coloneqq i + 1$
8 | | S_{Fils} := **call** $local\_search$ (S_{Fils})
9 | | **if** (criteria are interesting (Sol))
10 | | | **call** add_to_pop (POP, Sol)
11 | | | $j \coloneqq 0$
12 | | **else**
13 | | | $j \coloneqq j + 1$
14 | | **end if**
15 | **end while**
16 **until** (!stopping_criterion) **and** ($|POP_C| \leq ns$))
17 $POP \coloneqq POP + POP\_C$
18 **return** POP

End

Parmi ces solutions filles, seules celles dont les critères appartiennent à la zone d'acceptation sont ajoutées à la population finale. Cette zone est définie pour chaque solution $S_{père}$,. Elle est déterminée en fonction des critères de la solution $S_{père}$ ainsi que des critères des solutions de même rang qui la précédente et qui la suivent, comme illustré dans la figure 4.50.

Comme les solutions de la population sont triées lors du tri par non-domination et que chaque rang est ordonné, cette zone est obtenue en $O(1)$. Savoir si S_{Fils} appartient à la zone est donc plus rapide que de calculer si S_{Fils} fait partie des solutions non-dominées par la totalité des solutions de même rang que $S_{père}$. Cela permet de diriger le front vers les zones les plus intéressantes.

Un avantage évident est que plus les solutions qui entourent S_{pere} sont éloignées sur le front, plus la zone d'acceptation est vaste et donc une quantité plus importante de solutions est introduite.

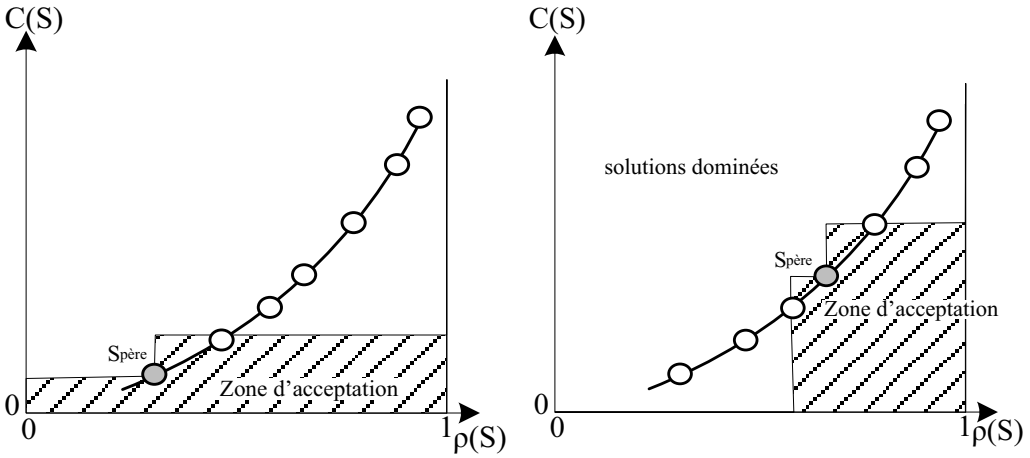


Figure 4.50 : Les zones d'acceptations

La figure 4.51 illustre l'exécution de l'algorithme la génération de solutions filles pour une solution $S_{père}$. Dans cet exemple les trois premières solutions S_{fils} ne sont pas ajoutées à la population car elles n'appartiennent pas à la zone d'acceptation de $S_{père}$. Par contre les 4èmes et 5èmes solutions filles sont ajoutées.

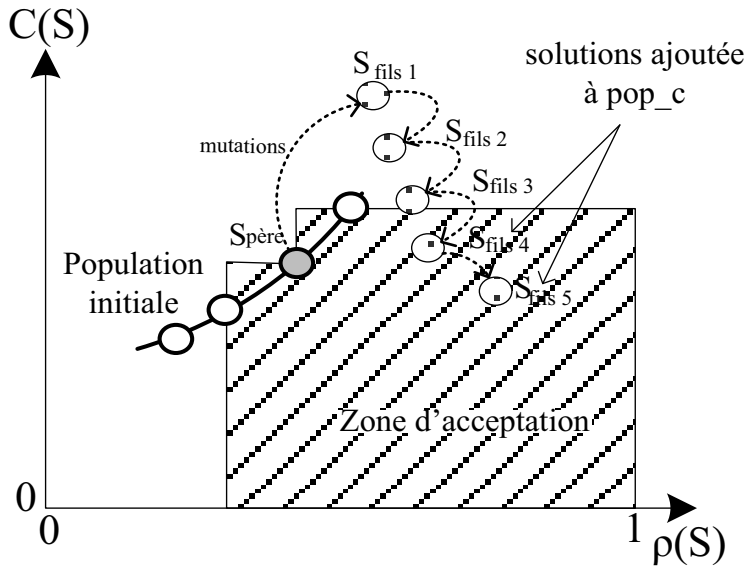


Figure 4.51 : Les solutions fils ajoutées à la population

e) Diversification des solutions

La génération de la population initiale est combinée avec une table de hachage afin de ne pas générer des solutions identiques. Le même procédé que pour le DDARP est repris. A une solution est associée une valeur de hachage $Hash$. Pour plus d'information, se reporter au paragraphe 2.6.3 sur la diversification pour la méthode ELS mise en place au chapitre 2.

Si la valeur de hachage est rencontrée un nombre de fois supérieur à une constante prédéfinie, la solution ne peut pas être ajoutée à la population.

4.7.4. Les résultats

Les résultats sont obtenus sur les instances proposées par (Cordeau and Laporte, 2003) dans lesquelles les temps de trajet ne sont plus supposées déterministes.

Les résultats sont obtenus après 5 exécutions. L'ordinateur utilisé est le même que celui utilisé pour les autres exécutions. Le processeur est un Intel® Core™ i7-3770 CPU @ 3.40GHz. La méthode a été codée en C++ sous Windows 7. Le nombre de Mflops est estimé à 2529. Pour plus de détails, il est possible de se reporter aux tableaux de la session 1.7.1.

Concernant le paramétrage de la méthode, la population initiale est constituée de 100 solutions et la constante associée au nombre de fois qu'une clé de hachage peut être rencontrée est de 2.

Le critère d'arrêt est le temps d'exécution. Chaque instance est exécutée pendant un temps équivalant à que celui utilisé pour résoudre le DDARP et donc il est aussi identique au temps pour résoudre de SDARP avec l'approche ELS. Les critères d'arrêts de la recherche locale ELS sont soit un nombre d'itération supérieur à 100, soit 10 exécutions successives sans ajouter de solutions à la population.

Comme expliqué précédemment, deux optimisations différentes ont été effectuée. Une pour trouver les solutions les plus robustes pour les hypothèses dans lesquelles les véhicules ne sont pas autorisés à prendre de l'avance (hypothèse H_1). Une autre pour les hypothèses H_2 et H_3. Les résultats sont détaillés dans le tableau 4.8 pour le premier critère ρ et dans le tableau 4.9 pour le second.

Les tableaux ont un format similaire, ils se composent d'une première colonne avec le nom de l'instance traitée, suivie de colonnes reparties en 4 groupes :

- $front\ de\ Pareto$ contient des informations sur le front de Pareto formé par les solutions finales de rang 1 dans la population ;
- $solution\ de\ coût\ minimal$ contient des informations sur la solution de rang 1 qui minimise le front ;
- $solution\ de\ robustesse\ maximale$ contient des informations sur la solution de rang 1 qui maximise le critère de robustesse dans le front ;
- $temps$ représente le temps d'exécution de la méthode. Le temps maximal autorisé est le même que celui utilisé pour résoudre le SDARP avec la méthode monocritère.

Chacun de ces groupes se compose de plusieurs colonnes. Si on suppose que X est un critère étudié, les colonnes $\bar{X}$ représentent les résultats moyens sur les 5 exécutions et les résultats X^* sont les valeurs pour la meilleures solution obtenue sur les 5 exécutions.

Le groupe de colonnes 1, nommé front de Pareto, contient les critères suivants : nb le nombre de solutions sur le front de Pareto et HV la mesure de l'hypervolume associé.

Les groupes de colonnes 2 et 3 contiennent les critères suivants : gap l'écart entre le coût de la solution considérée dans le front et le coût de la BKS (dont les valeurs sont reportées dans le tableau 4.5) ; ρ représente le critère de robustesse ; $P_{Hi}(s)$ est un estimateur de la robustesse sous l'hypothèse Hi obtenue comme expliqué précédemment par simulation. Il correspond donc à $\overline{F_{Hx}(s,n)}$ pour n = 100000.

Le dernier groupe contient uniquement le temps d'exécution moyen de la méthode exprimé en minutes.

a) Résultats sous l'hypothèse H1

Les résultats obtenus sous l'hypothèse H_1 sont détaillés dans le tableau 4.8.

Concernant les résultats sur l'hypervolume du front $\overline{HV}$ et HV^*, ils sont tous un indice de qualité supérieure à 0, qualité définie dans le paragraphe 4.7.2. Ceux mis en gras dans le tableau sont de qualité 2 et il n'existe pas de fronts de qualité 3 sous cette hypothèse.

Concernant la solution qui minimise le coût de la solution, l'écart avec la BKS est en moyenne de 5,46%. Le critère de robustesse est en moyenne de 49,02%, ce qui correspond après la phase de simulation à une probabilité d'être réalisable de 42.21%. Donc en moyenne, sur toutes les instances, pour plus d'une réalisation sur deux, la solution proposée viole une contrainte. Même si cette probabilité n'est pas très élevée, la robustesse obtenue sur les BKS de la littérature, reportée dans le tableau 4.5, est seulement de 24% en moyenne. Ce qui indique que la méthode n'explore probablement pas assez les solutions extrêmes et se dirige plutôt vers les solutions de compromis entre les critères, la BKS et les solutions du front restant incomparables, comme illustré dans la figure 4.52.

Concernant la solution qui a le coût le plus faible obtenu pour les 5 exécutions, le gap est de 3.9% en moyenne sur toutes les instances avec une valeur de 0% pour une seule instance, pr01 qui est aussi la plus petite. Dans le tableau, il peut être surprenant de constater que le critère de robustesse et la probabilité d'être réalisable associés à la solution de coût minimal sont tous les deux supérieurs à leur valeur moyenne correspondante, avec 53.32% contre 49.02% pour ρ et 45.2% contre 42.21%. Ces résultats indiquent simplement que les solutions de coût minimal de certaines exécutions dominent les solutions des autres exécutions.

Enfin concernant la solution dont le critère de robustesse est le plus important dans le front final de chaque instance, on constate que ρ, ainsi que la probabilité d'être réalisable, sont en moyenne très proche de 100% avec une valeur minimale pour pr10 de 95.6%. Cela reste quand même légèrement inférieur aux solutions obtenues par la méthode monocritère représentée dans le tableau 4.6 qui est de 100%. L'écart moyen obtenu est quant à lui bien inférieur avec 10.79% contre 13.19% dans le tableau 4.6.

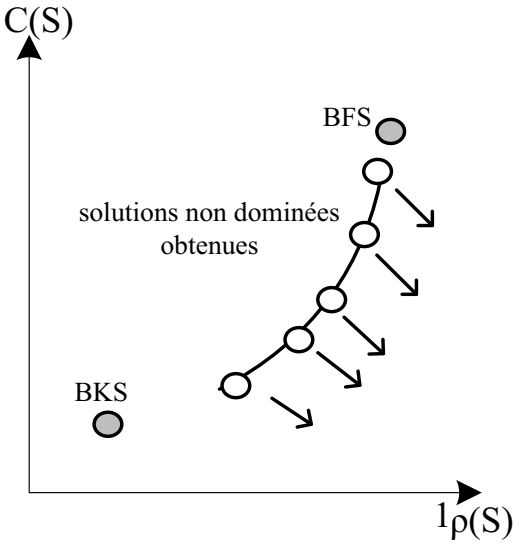


Figure 4.52 : Positionnement des fronts par rapport à la BKS et à la BFS pour H1

Tableau 4.8:
résultats sur les instances de (Cordeau & Laporte, 2003) sous l'hypothèse H1

nom	**front de Pareto**			**solution de coût minimal**						**solution de robustesse maximale**						**temps**
	$\overline{nb}$	$\overline{HV}$	HV^*	$\overline{gap}$	gap^*	$\overline{\rho}$	ρ^*	$\overline{P_{H1}(s)}$	$P_{H1}(s)^*$	$\overline{gap}$	gap^*	$\overline{\rho^*_{\square}}$	$\rho^*_{\square}$	$\overline{P_{H1}(s)}$	$P_{H1}(s)^*$	**min**
pr01	14.6	191.49	**191.12**	0.08%	0.00%	75.8%	74.8%	87.3%	85.4%	3.87%	4.31%	100%	100%	100%	100%	0.25
pr02	20.6	308.49	304.74	2.18%	0.86%	85.2%	88.7%	79.2%	90.3%	4.17%	5.51%	100%	100%	100%	100%	1.25
pr03	42.2	**556.71**	**548.83**	4.13%	2.84%	55.3%	66.0%	55.2%	86.8%	12.40%	11.03%	100%	100%	100%	100%	2.30
pr04	43	605.04	**589.91**	5.85%	3.18%	59.3%	39.6%	55.4%	32.9%	9.95%	12.38%	100%	100%	100%	100%	7.37
pr05	49.4	**673.04**	**661.44**	6.88%	5.16%	37.7%	54.8%	31.4%	38.6%	10.61%	12.21%	100%	100%	100%	100%	12.07
pr06	73.4	**840.56**	**826.05**	6.23%	4.24%	21.5%	16.9%	12.6%	0.3%	10.84%	10.08%	99.9%	100%	100%	100%	21.92
pr07	13.4	298.60	**295.80**	2.11%	1.21%	66.7%	83.7%	63.6%	81.5%	5.12%	5.79%	100%	100%	100%	100%	0.47
pr08	49.6	**517.77**	**510.35**	5.35%	3.61%	39.2%	33.5%	31.5%	37.3%	18.87%	20.09%	100%	100%	100%	100%	2.68
pr09	84	721.65	**702.09**	8.06%	5.90%	26.1%	22.9%	20.6%	5.8%	18.01%	13.18%	99.8%	100%	100%	100%	11.25
pr10	89	979.21	**930.37**	8.17%	7.02%	13.3%	11.1%	2.5%	2.2%	17.51%	20.80%	95.6%	100%	95.5%	100%	21.33
pr11	7.6	168.58	165.80	2.19%	0.25%	77.3%	78.5%	78.3%	75.1%	3.71%	4.52%	100%	100%	100%	100%	0.28
pr12	20.4	307.75	305.25	3.77%	2.72%	63.3%	46.4%	57.8%	26.3%	5.98%	5.89%	100%	100%	100%	100%	1.37
pr13	38	522.72	516.31	7.39%	6.22%	60.2%	52.6%	64.3%	43.3%	13.20%	12.23%	100%	100%	100%	100%	3.70
pr14	29.8	569.96	565.88	7.37%	6.66%	55.0%	79.6%	52.1%	95.3%	11.58%	12.92%	100%	100%	100%	100%	10.20
pr15	30	612.57	**590.08**	5.77%	2.09%	51.5%	96.7%	23.6%	48.4%	8.28%	9.33%	100%	100%	96.4%	100%	19.93
pr16	72.8	**789.88**	**784.44**	7.48%	6.47%	28.4%	28.4%	22.5%	38.7%	11.37%	9.94%	100%	100%	100%	100%	32.32
pr17	19.2	259.43	**254.45**	4.11%	2.04%	46.5%	46.7%	31.0%	5.7%	8.86%	10.80%	100%	100%	100%	100%	0.58
pr18	29	484.80	474.71	5.25%	3.11%	58.8%	80.4%	47.3%	78.1%	11.10%	12.17%	100%	100%	100%	100%	4.32
pr19	61	646.43	**639.25**	8.16%	6.80%	29.9%	23.0%	15.7%	1.2%	15.92%	22.34%	100%	100%	100%	100%	12.43
pr20	69.6	**862.86**	**847.40**	8.73%	7.59%	29.4%	42.0%	12.3%	27.1%	14.47%	19.04%	99.6%	100%	99.9%	100%	31.45
avg.	**42.8**	**545.88**	**535.21**	**5.46%**	**3.90%**	**49.02%**	**53.32%**	**42.21%**	**45.02%**	**10.79%**	**11.73%**	**99.73%**	**100%**	**99.58%**	**100%**	**9.87**

b) Résultats sous les hypothèses H_2 et H_3

Les résultats obtenus sous l'hypothèse H_2 et H_3 sont détaillés dans le tableau 4.9. Sous ces hypothèses, le véhicule peut gagner du temps sur les dates de début de service. C'est pourquoi un second critère de robustesse ρ est utilisé. Il correspond au critère $\rho^*_{H_2/H_3}$ présenté dans la partie précédente.

Comme précédemment, tous les indices HV du front de Pareto ont une qualité de 1 selon la définition donnée dans le paragraphe 4.7.2 et 11 instances ont un HV moyen de qualité 2. Contrairement au cas H_1, 3 instances (pr03, pr06, pr08) ont une valeur HV^*qui correspond à une qualité 3. Dans le tableau, elles sont représentées par des valeurs en gras soulignées et comme expliqué, de telles valeurs impliquent que le front comporte au moins une solution qui a un coût inférieur au coût de la BKS ou qui a un ρ supérieur à BFS. Pour ces trois instances c'est la BFS qui a été améliorée.

Concernant la solution de coût minimal dans les fronts, le gap moyen sur toutes les instances est de 5.8%, ce qui est assez éloigné de la BKS pour le problème du DARP. Comme sous hypothèse H_1, on constate que ρ ainsi que la probabilité de réalisabilité des solutions pour H_2 et H_3 sont supérieurs à celle obtenue pour les BKS. Par exemple, $P_{H_2}(S)$ à une valeur moyenne de 17.6% contre 6.02% pour la BKS, qui est reportée dans le tableau 4.5. Le front est donc probablement encore dans une configuration comme celle illustrée dans la figure 4.52.

Concernant les solutions de robustesse maximale, les résultats obtenus sont en moyenne légèrement de meilleure qualité que celle des solutions obtenues dans le même temps par la méthode monocritère. La valeur de ρ moyenne étant de 98.4% contre 97.8% pour le cas monocritère. De plus les écarts de ces solutions avec les BKS sont aussi meilleurs avec 14.3% contre 18.30% pour le cas monocritère, tableau 4.6. Pour la plupart des instances, les solutions générées par la méthode bi-objectif dominent les solutions générées par la méthode monocritère comme illustré sur la figure 4.53. Même si nous n'en avons pas la preuve, il est probable que la diversité de la population permette à l'algorithme de mieux parcourir l'espace des solutions concernant le second critère.

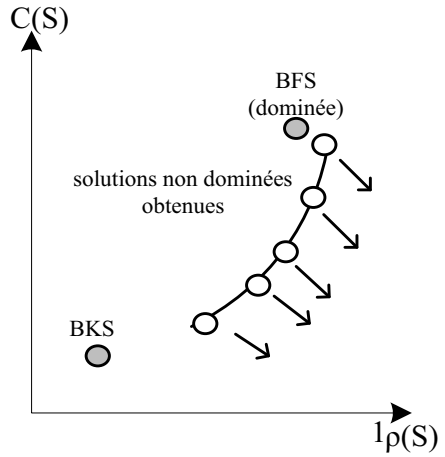


Figure 4.53 : Positionnement des fronts par rapport à la BKS et à la BFS pour H2 et H3

Tableau 4.9:
résultats sur les instances de (Cordeau & Laporte, 2003) sous les hypothèses H2 et H3

nom	front de Pareto			solution de coût minimal								solution de robustesse maximale								temps
	$\overline{nb}$	$\overline{HV}$	HV^*	$\overline{gap}$	gap^*	$\overline{\rho}$	ρ^*	$\overline{P_{H2}(s)}$	$P_{H2}(s)^*$	$\overline{P_{H3}(s)}$	$P_{H3}(s)^*$	$\overline{gap}$	gap^*	$\overline{\rho^*_{\square}}$	$\rho^*_{\square}$	$\overline{P_{H2}(s)}$	$P_{H2}(s)^*$	$\overline{P_{H3}(s)}$	$P_{H3}(s)^*$	min
pr01	24.2	191.91	**191.64**	0.0%	0.00%	66.8%	67.0%	25.9%	24.0%	49.6%	48.0%	5.7%	6.1%	100%	100.0%	100%	100%	100%	100%	0.25
pr02	35.2	308.10	304.93	1.6%	0.00%	55.9%	49.0%	37.2%	27.0%	60.5%	75.0%	6.0%	8.5%	100%	100.0%	100%	100%	100%	100%	1.25
pr03	73.4	**565.36**	**551.14**	3.8%	2.02%	30.3%	52.0%	10.4%	26.0%	37.5%	64.0%	21.1%	22.9%	99.6%	100.0%	99.3%	100%	100%	100%	2.30
pr04	71	**612.05**	**605.14**	5.9%	4.38%	33.2%	14.0%	10.7%	0.0%	22.5%	0.0%	13.6%	18.1%	99.8%	99.9%	99.5%	99.9%	100%	100%	7.37
pr05	84.2	**672.92**	**665.36**	6.3%	4.42%	31.0%	13.0%	12.7%	2.0%	29.3%	30.0%	16.4%	21.3%	100%	100.0%	99.9%	100%	100%	100%	12.07
pr06	86.2	**869.53**	**832.96**	8.1%	5.54%	20.6%	42.0%	6.7%	21.0%	25.9%	65.0%	15.0%	17.3%	98.7%	100.0%	98.0%	100%	99.9%	100%	21.92
pr07	28	**299.52**	**297.36**	2.0%	1.36%	62.8%	57.0%	44.7%	29.0%	73.2%	67.0%	12.6%	15.6%	100%	100.0%	100%	100%	100%	100%	0.47
pr08	64	**532.67**	**528.96**	5.8%	4.15%	24.9%	16.0%	5.5%	1.0%	22.0%	8.0%	22.6%	18.4%	99.5%	99.9%	99.2%	99.6%	100%	100%	2.68
pr09	80.4	**762.97**	**729.66**	8.6%	6.72%	15.6%	23.0%	3.3%	10.0%	21.7%	59.0%	20.6%	20.2%	97.4%	98.9%	96.0%	98.0%	100%	100%	11.25
pr10	92.8	1214.00	**1121.69**	8.5%	7.32%	13.8%	3.0%	2.7%	0.0%	21.4%	0.0%	15.0%	19.5%	74.3%	83.3%	60.0%	77.3%	87.4%	97.7%	21.33
pr11	10.6	167.93	166.38	1.5%	0.00%	68.5%	58.0%	53.3%	35.0%	75.0%	67.0%	4.7%	5.4%	100%	100.0%	100%	100%	100%	100%	0.28
pr12	33.6	306.84	305.79	3.3%	3.03%	62.9%	45.0%	37.9%	0.0%	57.7%	3.0%	9.9%	12.6%	100%	100.0%	100%	100%	100%	100%	1.37
pr13	31	526.34	509.73	8.0%	4.71%	32.2%	35.0%	9.8%	3.0%	26.2%	26.0%	15.5%	22.7%	100%	100.0%	100%	100%	100%	100%	3.70
pr14	60.4	**574.40**	**566.27**	7.6%	6.25%	35.9%	27.0%	14.5%	14.0%	34.4%	42.0%	14.6%	15.4%	100%	100.0%	100%	100%	100%	100%	10.20
pr15	56.6	614.85	**603.74**	5.9%	3.90%	30.7%	14.0%	16.3%	1.0%	25.5%	4.0%	10.9%	10.6%	100%	100.0%	100%	100%	100%	100%	19.93
pr16	71.6	803.60	**790.57**	9.2%	5.99%	30.5%	34.0%	13.9%	21.0%	38.7%	49.0%	14.9%	17.9%	99.8%	100.0%	99.5%	100%	100%	100%	32.32
pr17	25.6	260.86	256.30	4.4%	2.09%	41.4%	42.0%	10.0%	15.0%	27.8%	43.0%	11.4%	13.1%	100%	100.0%	100%	100%	100%	100%	0.58
pr18	34.4	**498.83**	**494.02**	8.1%	6.39%	55.9%	28.0%	32.4%	3.0%	53.9%	10.0%	17.1%	20.4%	100%	100.0%	100%	100%	100%	100%	4.32
pr19	87.2	**648.55**	**638.63**	7.5%	5.90%	21.3%	13.0%	3.8%	0.0%	14.2%	1.0%	18.7%	21.5%	99.8%	100.0%	99.9%	100%	100%	100%	12.43
pr20	92.2	**880.34**	**875.96**	10.3%	8.96%	9.9%	4.0%	0.3%	0.0%	2.5%	0.0%	19.1%	17.3%	99.8%	99.9%	99.6%	100%	100%	100%	31.45
avg.	57.1	**565.58**	**551.81**	**5.8%**	**4.2%**	**37.2%**	**31.8%**	**17.6%**	**11.6%**	**36.0%**	**33.0%**	**14.3%**	**16.2%**	**98.4%**	**99.1%**	**97.5%**	**98.7%**	**99.4%**	**99.9%**	**9.87**

4.8. Conclusion

Dans ce chapitre, le problème du DARP stochastique est traité. Dans ce problème, les temps de trajet sont soumis à des variations qui ne sont pas connues durant la phase d'optimisation. Sur chaque arc, ces variations sont supposées suivre une loi normale. L'objectif est de générer des solutions qui respectent les contraintes du problème dans le cas déterministe, mais qui sont en plus robustes aux variations sur les temps de transport. Deux critères sont alors associés à une solution : la distance parcourue par la flotte et la probabilité que la solution soit réalisable dans le cas stochastique.

Il a été mis en évidence qu'en plus des variations sur les temps de trajet, le comportement des chauffeurs sur les sommets a une influence sur la robustesse des solutions. Beaucoup de comportements peuvent être envisagés et chacun représente une hypothèse de gestion. Nous nous sommes intéressé dans ce chapitre à trois hypothèses principales. Pour chacune de ces hypothèses, en utilisant les lois de probabilités associées au temps de trajet, une nouvelle fonction d'évaluation a été mise en place ainsi qu'un critère de robustesse corrélé avec la probabilité que la solution soit robuste. Le calcul analytique de la robustesse d'une solution est possible, mais il reste très coûteux. C'est pourquoi la méthode de résolution proposée utilise une évaluation de la robustesse par simulation.

Le problème est traité selon deux approches différentes. La première approche est une approche monocritère dans laquelle les critères sont pondérés de manière à favoriser la robustesse. Le résultat obtenu montre qu'il est possible de générer des solutions robustes qui ont des probabilités d'être réalisables supérieures à 90% à l'exception d'une des plus grandes instances dont la probabilité est de 63% seulement.

La seconde approche, quant à elle, est une méthode bi-objectif qui utilise le tri par non domination. Les éléments sont choisis aléatoirement en fonction de leur position dans la population. L'élément choisi subit une phase de mutation puis une phase de recherche locale. Durant la recherche locale des solutions filles sont générées. Si ces solutions filles ont des valeurs de critère jugées intéressantes, alors elles sont ajoutées dans la population. Les critères intéressants sont déterminés par la position du sommet initialement sélectionné dans la population. Une fois le nombre maximal de solutions dans la population atteint, la méthode utilise le tri par non domination pour conserver uniquement, à chaque itération, un sous-ensemble de solutions. La méthode de résolution, pour un même temps d'exécution que la méthode par agrégation des critères, permet d'obtenir un ensemble de solutions non-dominées. Conserver une forte diversité dans la population de solutions permet à la méthode de mieux parcourir l'espace des solutions et, pour certaines instances, elle permet d'obtenir des solutions meilleures que celles obtenues avec la méthode monocritère.

Plusieurs perspectives peuvent être envisagées. Premièrement, le modèle proposé peut être étendu à d'autres lois de probabilités. Cette perspective est envisageable car les critères de robustesse, le calcul analytique et les simulations développés dans cette thèse pour la loi normale, peuvent être adaptés facilement à d'autres lois. Deuxièmement, la variance des lois utilisées est proportionnelle à la longueur de l'arc considéré. Il serait intéressant de construire des instances dans lesquelles cette variance dépendrait à la fois de la longueur mais aussi du type d'arc, avec par exemple certains arcs dont l'espérance mathématique de la durée est courte mais la variance est grande. Le véhicule aurait alors à choisir entre les arcs de variabilité faible mais de durée importante ou des arcs de plus faible durée mais à forte variabilité. Ainsi des tournées de faible coût mais aussi de faible robustesse seraient une alternative à des tournées de coût plus important mais aussi de robustesse plus élevée. Troisièmement, ce type de modélisation a été appliqué uniquement sur le DARP dans le cadre

de la thèse. Cette modélisation pourrait être étendue aux autres problèmes de tournées de véhicules qui prennent en compte les évènements aléatoires.

Un autre point intéressant serait d'inclure dans le modèle des phénomènes de congestion de proche en proche du réseau comme cela ce produit en pratique lors d'un accident dans un réseau urbain. Un tel événement induit une congestion sur un arc mais aussi sur les arcs voisins. Cela remettrait en cause l'hypothèse d'indépendance entre les lois qui modélisent les temps de trajet sur les arcs.

Pour finir, une perspective importante qui pourrait être envisagée serait de prendre en compte la période de la journée dans la loi associée à chaque arc, car les probabilités d'occurrence de congestion ne sont pas les mêmes en fonction de l'heure. De même, la variance peut être considérée comme dépendante de l'heure. Ainsi, comme pour le TDVRP présenté dans le chapitre 3, le modèle prendrait en considération à la fois les événements prévisibles, et aussi les événements non prévisibles qui impactent la durée de parcours. Cela constituerait une amélioration sensible de la précision du modèle.

Références

Braekers, K., Caris, A., Janssens, G.K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transp. Res. Part B Methodol. 67, 166–186. doi:10.1016/j.trb.2014.05.007

Cordeau, J.-F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp. Res. Part B Methodol. 37, 579–594. doi:10.1016/S0191-2615(02)00045-0

Firat, M., Woeginger, G.J., 2011. Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. Oper. Res. Lett. 39, 32–35. doi:10.1016/j.orl.2010.11.004

Fleury, G., Lacomme, P., Prins, C., Sevaux, M., 2005. A memetic algorithm for a bi-objective and stochastic CARP. Proceeding MIC 22–26.

Gauvin, C., Desaulniers, G., Gendreau, M., 2014. A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. Comput. Oper. Res. 50, 141–153. doi:10.1016/j.cor.2014.03.028

Guerriero, F., Pezzella, F., Pisacane, O., Trollini, L., 2014. Multi-objective Optimization in Dial-a-ride Public Transportation. Transp. Res. Procedia, 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain 3, 299–308. doi:10.1016/j.trpro.2014.10.009

Parragh, S.N., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. Comput. Oper. Res. 40, 490–497. doi:10.1016/j.cor.2012.08.004Schilde, M., Doerner, K.F., Hartl, R.F., 2014. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. Eur. J. Oper. Res. 238, 18–30. doi:10.1016/j.ejor.2014.03.005

Zidi, I., Mesghouni, K., Zidi, K., Ghedira, K., 2012. A multi-objective simulated annealing for the multi-criteria dial a ride problem. Eng. Appl. Artif. Intell. 25, 1121–1131. doi:10.1016/j.engappai.2012.03.012

Conclusion générale

Dans le cadre de cette thèse, nous nous sommes intéressé aux problèmes de transport à la demande (DARP). Ces problèmes consistent à déterminer des tournées pour satisfaire les demandes de clients souhaitant être transportés depuis des lieux d'origine vers des lieux de destination. Une des difficultés de ce problème est de prendre en compte différentes contraintes, comme les fenêtres de temps et les temps de transport des clients. Après avoir réalisé une classification des différents problèmes appartenant à la famille des tournées de véhicules et des méthodes de résolution, nous nous sommes intéressé à trois problèmes particuliers. Nous proposons pour chaque problème une méthode de résolution approchée.

Dans un premier temps, nous nous sommes intéressé au problème de transport à la demande comme il est définit dans la littérature. C'est un problème statique qui appartient à la famille des problèmes $One\text{-}to\text{-}One$. Nous avons proposé une méthode de type métaheuristique basée sur un schéma de recherche locale évolutive (ELS : Evolutionary Local Search). Cette méthode comprend différentes originalités :

- une fonction d'évaluation efficace qui permet d'obtenir les dates au plus tôt et au plus tard ;
- une recherche locale qui adapte ses paramètres durant l'exécution afin de s'adapter à l'instance considérée ;
- une fonction de génération de solutions initiales.

Dans un deuxième temps, nous avons étudié un problème moins contraint que le DARP qui est le VRP. Le VRP fait partie de la famille des $One\text{-}to\text{-}Many\text{-}to\text{-}One$, et contrairement au DARP, il n'y a pas de fenêtre de temps ou de temps de trajet associé à un client. A ce problème classique s'ajoutent dans le modèle :

- les temps de transport sur les axes routiers qui varient en fonction de l'heure de départ du véhicule ;
- les plus courts chemins entre les clients ne sont pas précalculés, car ils évoluent au cours du temps ;
- les véhicules sont autorisés à attendre sur les nœuds du graphe pour éviter les phénomènes de bouchons.

Le problème est traité de manière bi-critères, avec d'une part le temps de conduite et d'autre part le temps total de la tournée (incluant les pauses). La méthode de résolution proposée est une approche en deux phases : premièrement le TDVRP est résolu sans autoriser l'attente avec une métaheuristique de type GRASP×ELS ; puis, à l'aide des meilleures solutions obtenues durant la première phase, un second schéma métaheuristique de type Path Relinking (PR) est utilisé pour résoudre le problème dans lequel l'attente est autorisée. Un nouvel ensemble de solutions est proposé.

Enfin, nous avons traité une extension du DARP dans laquelle les temps de trajet entre les clients ne sont plus modélisés par des constantes mais par des variables aléatoires. Ces variables aléatoires suivent des lois de probabilité connues. Dans ce modèle, à une solution est associé un coût ainsi qu'un critère de robustesse qui est corrélé avec la probabilité que la tournée soit réalisable.

Deux méthodes de résolution ont été proposées. La première pondère les deux critères afin de résoudre le problème de manière mono-objectif en recherchant en priorité les solutions les

plus robustes. Pour cette méthode de résolution l'approche ELS, présentée dans la 1ère partie a été adaptée. La deuxième méthode de résolution est bi-critères et à la fin d'une exécution, un ensemble de solutions non dominées est obtenu. L'approche utilisée est basée sur un algorithme à population NSGA-2 (*non-dominated Sorting Genetic Algorithm*). Ces deux méthodes ont demandé la mise en place d'une fonction spécifique d'évaluation d'une tournée, qui soit capable de déterminer les dates de passages sur les sommets afin de générer la solution la plus robuste possible. Les instances classiques de la littérature ont été adaptées et réutilisées pour tester l'efficacité de nos méthodes.

Dans cette thèse, nous nous sommes intéressé essentiellement à des problèmes de tournées de véhicules liés à des transports de clients, avec des méthodes de résolution approchées. Ces travaux offrent plusieurs perspectives.

A court terme il est possible de s'intéresser à la résolution du DARP avec des durées de transport qui varient au cours de la journée, en généralisant l'approche introduite dans le chapitre 3. D'autre part, il serait possible de définir une approche unifiant l'approche stochastique et celle déterministe basée sur le découpage de la journée en fenêtre horaire, en associant à chaque période un ensemble de variables aléatoires spécifiques.

Sur le long terme, des axes de recherche nouveaux sont à définir dans la continuité des travaux actuels en se basant essentiellement sur les enjeux liés aux problèmes de tournées. Dans cette optique, 3 axes prioritaires de recherches nous sembleraient intéressants à développer car ils correspondent à des enjeux sociétaux majeurs.

De notre point de vue, il y aurait un intérêt certain à inclure l'aspect économique dans la résolution des problèmes de tournées, non pas comme une nouvelle contrainte à inclure, mais comme un objectif à minimiser ou comme une situation d'équilibre à trouver. On peut penser par exemple à un système de tarification variable dans lequel le prix payé par le client dépendrait du niveau de qualité de service qu'il souhaite recevoir. On ferait alors le lien entre d'une part, le fournisseur d'un service et d'autre part les clients vus comme un acteur économique acceptant ou non une prestation par rapport à un prix et à une qualité négociée. Ce type d'enjeu est déjà présent dans de nombreux secteurs économiques (par exemple pour les forfaits téléphoniques). Il est très important et devrait être intégré à la résolution des problèmes car il offre un intérêt économique certain. Les services de transport privé de type Uber proposent un autre mode de transport avec un prix différent et une qualité différente par rapport aux clients. Remarquons alors, que la définition de la qualité reste à préciser. La relation entre la qualité et le prix intervient aussi dans les transports publics ou assimilés puisque, sur les moyennes distances et avec la libéralisation du marché interurbain, on a parfois le choix entre le bus, le train et l'avion à des prix bien sûr différents et aussi des qualités de service différentes. Ainsi, inclure une notion de "juste prix" par rapport à une qualité de service pourrait conduire à redéfinir certains problèmes de transport et même de transport multimodaux.

Le second axe prospectif à développer pourrait s'organiser autour de la notion d'équité, souvent absente dans les problèmes de transport. La plupart du temps, la qualité du transport est définie avec un indicateur de retard moyen, oubliant alors l'indicateur propre à chaque client. Or, du point de vue du client, ce dernier est de fait le seul indicateur intéressant. Ainsi promettre à chaque client de rejoindre sa destination en moins de 3h en moyenne est-il plus intéressant du point de vue de chaque client que de garantir que tous les clients seront à destination en 3h30 au pire. On peut généraliser la réflexion, en pensant à la manière dont les usagers perçoivent la qualité d'un service public. Bien souvent l'usager est sensible à la qualité de service dont il bénéficie mais il peut aussi être agacé de constater que d'autres usagers ont eu un meilleur service. Les scènes de querelles dans un bureau de poste, une gare ou un

aéroport, liées à la perception d'une qualité de service mesurée en temps d'attente, en sont autant d'exemples significatifs. Pour le DARP et tous les problèmes de transport où la qualité de service intervient, on pourrait se préoccuper de créer des solutions équitables entre les usagers, c'est-à-dire des solutions proches en terme de qualité de service. Dans le cas du DARP, cette notion d'équité pourrait reposer sur le *Riding Time* de chaque client.

Le troisième axe de recherche à développer est un axe, de notre point de vue, qui devrait s'intéresser à la notion de tournée tout en garantissant à la fois la sécurité des données des usagers et des chauffeurs. Parce que ces réflexions n'ont pas été poussées jusqu'au bout, certains chauffeurs de la société Uber ont été agressés physiquement. En effet, leur position GPS était publique et mise à jour en temps réel sur le réseau routier. Ceci a permis à leurs agresseurs de choisir les chauffeurs et d'organiser des guet-apens. Pensons aussi aux clients dont les demandes de transport sont mises à disposition de tous les chauffeurs. La facilité d'accès à ces données soulève la question de l'usage potentiellement délictueux qui pourrait en être fait. On pourrait se poser à la question de ne pas diffuser d'information précise sur la position des clients, de ne pas révéler exactement la position des chauffeurs. En créant des informations "brouillées" on permettrait aux systèmes de planification de fonctionner sans pour autant exposer (sauf au dernier moment) des informations facilement exploitables pour organiser des actions dangereuses ou délictueuses.

Ainsi, l'ensemble de ces propositions permettrait de renforcer les liens entre la problématique de transport à la demande et les préoccupations industrielles et sociétales. Ceci ne se traduit pas nécessairement par l'ajout de nouvelles contraintes au sens mathématique. La définition de nouveaux critères d'évaluation, la prise en compte de la qualité de service ou la question de l'accessibilité des données en sont quelques exemples. Dans un contexte de transport par nature individualisé, d'autres problématiques peuvent aussi émerger telles que l'articulation avec les autres modes de transport, la capacité à répondre en temps réel à de nouvelles requêtes en modifiant les tournées existantes sont encore d'autres voies à explorer.